PETER FLACH

Machine Learning

The Art and Science of Algorithms that Make Sense of Data

CAMBRIDGE

CAMBRIDGE

more information- www.camebridge.com/9781107096394

MACHINE LEARNING

The Art and Science of Algorithms that Make Sense of Data As one of the most comprehensive machine learning texts around, this book does justice to the field's incredible richness, but without losing sight of the unifying principles.

Peter Flach's clear, example-based approach begins by discussing how a spam filter works, which gives an immediate introduction to machine learning in action, with a minimum of technical fuss. He covers a wide range of logical, geometric and statistical models, and state-of-the-art topics such as matrix factorisation and ROC analysis. Particular attention is paid to the central role played by features.

Machine Learning will set a new standard as an introductory textbook:

- The Prologue and Chapter 1 are freely available on-line, providing an accessible first step into machine learning.
- The use of established terminology is balanced with the introduction of new and useful concepts.
- Well-chosen examples and illustrations form an integral part of the text.
- Boxes summarise relevant background material and provide pointers for revision.
- Each chapter concludes with a summary and suggestions for further reading.
- A list of 'Important points to remember' is included at the back of the book together with an extensive index to help readers navigate through the material.

MACHINE LEARNING

The Art and Science of Algorithms that Make Sense of Data

PETER FLACH

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São Paulo, Delhi, Mexico City

Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK

Published in the United States of America by Cambridge University Press, New York
www.cambridge.org
Information on this title: www.cambridge.org/9781107096394

© Peter Flach 2012

This publication is in copyright. Subject to statutory exception and to the provisions of relevant collective licensing agreements, no reproduction of any part may take place without the written permission of Cambridge University Press.

First published 2012

Printed and bound in the United Kingdom by the MPG Books Group

A catalogue record for this publication is available from the British Library ISBN 978-1-10709639-4 Hardback
ISBN 978-1-107-42222-3 Paperback Additional resources for this publication at
www.cs.bris.ac.uk/home/flach/mlbook

Cambridge University Press has no responsibility for the persistence or accuracy of URLs for external or third-party internet websites referred to in this publication, and does not guarantee that any content on such websites is, or will remain, accurate or appropriate.

To Hessel Flach (1923–2006)

Brief Contents

Preface

Prologue: A machine learning sampler

1 The ingredients of machine learning

2 Binary classification and related tasks

2 Binary classification and related tasks

3 Beyond binary classification

4 Concept learning

5 Tree models

6 Rule models

7 Linear models

8 Distance-based models

9 Probabilistic models

10 Features

11 Model ensembles

12 Machine learning experiments

Epilogue: Where to go from here

Important points to remember

References

Index

Contents

Preface

Prologue: A machine learning sampler

1 The ingredients of machine learning

1.1 Tasks: the problems that can be solved with machine learning
Looking for structure
Evaluating performance on a task
1.2 Models: the output of machine learning
Geometric models
Probabilistic models
Logical models
Grouping and grading
1.3 Features: the workhorses of machine learning
Two uses of features
Feature construction and transformation
Interaction between features
1.4 Summary and outlook
What you'll find in the rest of the book

2 Binary classification and related tasks

2.1 Classification
Assessing classification performance
Visualising classification performance
2.2 Scoring and ranking
Assessing and visualising ranking performance
Turning rankers into classifiers

2.3 Class probability estimation
Assessing class probability estimates
Turning rankers into class probability estimators
2.4 Binary classification and related tasks: Summary and further reading

3 Beyond binary classification

3.1 Handling more than two classes
Multi-class classification
Multi-class scores and probabilities
3.2 Regression
3.3 Unsupervised and descriptive learning
Predictive and descriptive clustering
Other descriptive models
3.4 Beyond binary classification: Summary and further reading

4 Concept learning

4.1 The hypothesis space
Least general generalisation
Internal disjunction
4.2 Paths through the hypothesis space
Most general consistent hypotheses
Closed concepts
4.3 Beyond conjunctive concepts
Using first-order logic
4.4 Learnability
4.5 Concept learning: Summary and further reading

5 Tree models

5.1 Decision trees
5.2 Ranking and probability estimation trees
Sensitivity to skewed class distributions
5.3 Tree learning as variance reduction
Regression trees
Clustering trees
5.4 Tree models: Summary and further reading

6 Rule models

6.1 Learning ordered rule lists
Rule lists for ranking and probability estimation
6.2 Learning unordered rule sets
Rule sets for ranking and probability estimation
A closer look at rule overlap
6.3 Descriptive rule learning
Rule learning for subgroup discovery
Association rule mining
6.4 First-orderrule learning
6.5 Rule models: Summary and further reading

7 Linear models

7.1 The least-squares method
Multivariate linear regression
Regularised regression
Using least-squares regression for classification
7.2 The perceptron
7.3 Support vector machines
Soft margin SVM
7.4 Obtaining probabilities from linear classifiers
7.5 Going beyond linearity with kernel methods
7.6 Linear models: Summary and further reading

8 Distance-based models

8.1 So many roads
8.2 Neighbours and exemplars
8.3 Nearest-neighbour classification
8.4 Distance-based clustering
K-means algorithm
Clustering around medoids
Silhouettes
8.5 Hierarchical clustering
8.6 From kernels to distances
8.7 Distance-based models: Summary and further reading

9 Probabilistic models

9.1 The normal distribution and its geometric interpretations
9.2 Probabilistic models for categorical data

Using a naive Bayes model for classification
Training a naive Bayes model
9.3 Discriminative learning by optimising conditional likelihood
9.4 Probabilistic models with hidden variables
Expectation-Maximisation
Gaussian mixture models
9.5 Compression-based models
9.6 Probabilistic models: Summary and further reading

10 Features

10.1 Kinds of feature
Calculations on features
Categorical, ordinal and quantitative features
Structured features
10.2 Feature transformations
Thresholding and discretisation
Normalisation and calibration
Incomplete features
10.3 Feature construction and selection
Matrix transformations and decompositions
10.4 Features: Summary and further reading

11 Model ensembles

11.1 Bagging and random forests
11.2 Boosting
Boosted rule learning
11.3 Mapping the ensemble landscape
Bias, variance and margins
Other ensemble methods
Meta-learning
11.4 Model ensembles: Summary and further reading

12 Machine learning experiments

12.1 What to measure
12.2 How to measure it
12.3 How to interpret it
Interpretation of results over multiple data sets
12.4 Machine learning experiments: Summary and further reading

Epilogue: Where to go from here

Important points to remember

References

Index

Preface

This book started life in the Summer of 2008, when my employer, the University of Bristol, awarded me a one-year research fellowship. I decided to embark on writing a general introduction to machine learning, for two reasons. One was that there was scope for such a book, to complement the many more specialist texts that are available; the other was that through writing I would learn new things – after all, the best way to learn is to teach.

The challenge facing anyone attempting to write an introductory machine learning text is to do justice to the incredible richness of the machine learning field without losing sight of its unifying principles. Put too much emphasis on the diversity of the discipline and you risk ending up with a 'cookbook' without much coherence; stress your favourite paradigm too much and you may leave out too much of the other interesting stuff. Partly through a process of trial and error, I arrived at the approach embodied in the book, which is is to emphasise both unity and diversity: unity by separate treatment of *tasks* and *features*, both of which are common across any machine learning approach but are often taken for granted; and diversity through coverage of a wide range of logical, geometric and probabilistic models.

Clearly, one cannot hope to cover all of machine learning to any reasonable depth within the confines of 400 pages. In the Epilogue I list some important areas for further study which I decided not to include. In my view, machine learning is a marriage of statistics and knowledge representation, and the subject matter of the book was chosen to reinforce that view. Thus, ample space has been reserved for tree and rule learning, before moving on to the more statistically-oriented material. Throughout the book I have placed particular emphasis on intuitions, hopefully amplified by a generous use of examples and graphical illustrations, many of which derive from my work on the use of ROC analysis in machine learning.

How to read the book

The printed book is a linear medium and the material has therefore been organised in such a way that it can be read from cover to cover. However, this is not to say that one couldn't pick and mix, as I have tried to organise things in a modular fashion.

For example, someone who wants to read about his or her first learning

algorithm as soon as possible could start with Section 2.1, which explains binary classification, and then fast-forward to Chapter 5 and read about learning decision trees without serious continuity problems. After reading Section 5.1 that same person could skip to the first two sections of Chapter 6 to learn about rule-based classifiers.

Alternatively, someone who is interested in linear models could proceed to Section 3.2 on regression tasks after Section 2.1, and then skip to Chapter 7 which starts with linear regression. There is a certain logic in the order of Chapters 4–9 on logical, geometric and probabilistic models, but they can mostly be read independently; similar for the material in Chapters 10–12 on features, model ensembles and machine learning experiments.

I should also mention that the Prologue and Chapter 1 are introductory and reasonably self-contained: the Prologue does contain some technical detail but should be understandable even at pre-University level, while Chapter 1 gives a condensed, highlevel overview of most of the material covered in the book. Both chapters are freely available for download from the book's web site at `www.cs.bris.ac.uk/~flach/mlbook`; over time, other material will be added, such as lecture slides. As a book of this scope will inevitably contain small errors, the web site also has a form for letting me know of any errors you spotted and a list of errata.

Acknowledgements

Writing a single-authoredbook is always going to be a solitary business, but I have been fortunate to receive help and encouragement from many colleagues and friends. Tim Kovacs in Bristol, Luc De Raedt in Leuven and Carla Brodley in Boston organised reading groups which produced very useful feedback. I also received helpful comments from Hendrik Blockeel, Nathalie Japkowicz, Nicolas Lachiche, Martijn van Otterlo, Fabrizio Riguzzi and Mohak Shah. Many other people have provided input in one way or another: thank you.

José Hernández-Orallo went well beyond the call of duty by carefully reading my manuscript and providing an extensive critique with many excellent suggestions for improvement, which I have incorporated so far as time allowed. José: I will buy you a free lunch one day.

Many thanks to my Bristol colleagues and collaborators Tarek Abudawood, Rafal Bogacz, Tilo Burghardt, Nello Cristianini, Tijl De Bie, Bruno Golénia, Simon Price, Oliver Ray and Sebastian Spiegler for joint work and enlightening discussions. Many thanks also to my international collaborators Johannes

Fürnkranz, Cesar Ferri, Thomas Gärtner, José Hernández-Orallo, Nicolas Lachiche, John Lloyd, Edson Matsubara and Ronaldo Prati, as some of our joint work has found its way into the book, or otherwise inspired bits of it. At times when the project needed a push forward my disappearance to a quiet place was kindly facilitated by Kerry, Paul and David, Renée, and Trijntje.

David Tranah from Cambridge University Press was instrumental in getting the process off the ground, and suggested the pointillistic metaphor for 'making sense of data' that gave rise to the cover design (which, according to David, is 'just a canonical silhouette' not depicting anyone in particular – in case you were wondering…). Mairi Sutherland provided careful copy-editing.

I dedicate this book to my late father, who would certainly have opened a bottle of champagne on learning that 'the book' was finally finished. His version of the problem of induction was thought-provoking if somewhat morbid: the same hand that feeds the chicken every day eventually wrings its neck (with apologies to my vegetarian readers). I am grateful to both my parents for providing me with everything I needed to find my own way in life.

Finally, more gratitude than words can convey is due to my wife Lisa. I started writing this book soon after we got married – little did we both know that it would take me nearly four years to finish it. Hindsight is a wonderful thing: for example, it allows one to establish beyond reasonable doubt that trying to finish a book while organising an international conference and overseeing a major house refurbishment is really not a good idea. It is testament to Lisa's support, encouragement and quiet suffering that all three things are nevertheless now coming to full fruition. Dank je wel, meisje!

Peter Flach, Bristol

Prologue: A machine learning sampler

YOU MAY NOT be aware of it, but chances are that you are already a regular user of machine learning technology. Most current e-mail clients incorporate algorithms to identify and filter out spam e-mail, also known as junk e-mail or unsolicited bulk e-mail. Early spam filters relied on hand-coded pattern matching techniques such as regular expressions, but it soon became apparent that this is hard to maintain and offers insufficient flexibility – after all, one person's spam is another person's ham![1] Additional adaptivity and flexibility is achieved by employing machine learning techniques.

SpamAssassin is a widely used open-source spam filter. It calculates a score for an incoming e-mail, based on a number of built-in rules or 'tests' in SpamAssassin's terminology, and adds a 'junk' flag and a summary report to the e-mail's headers if the score is 5 or more. Here is an example report for an e-mail I received:

```
-0.1 RCVD_IN_MXRATE_WL       RBL: MXRate recommends allowing
                             [123.45.6.789 listed in sub.mxrate.net]
0.6 HTML_IMAGE_RATIO_02      BODY: HTML has a low ratio of text to image area
1.2 TVD_FW_GRAPHIC_NAME_MID  BODY: TVD_FW_GRAPHIC_NAME_MID
0.0 HTML_MESSAGE             BODY: HTML included in message
0.6 HTML_FONx_FACE_BAD       BODY: HTML font face is not a word
1.4 SARE_GIF_ATTACH          FULL: Email has a inline gif
0.1 BOUNCE_MESSAGE           MTA bounce message
0.1 ANY_BOUNCE_MESSAGE       Message is some kind of bounce message
1.4 AWL                      AWL: From: address is in the auto white-list
```

From left to right you see the score attached to a particular test, the test identifier, and a short description including a reference to the relevant part of the e-mail. As you see, scores for individual tests can be negative (indicating evidence suggesting the e-mail is ham rather than spam) as well as positive. The overall score of 5.3 suggests the email might be spam. As it happens, this particular e-mail was a notification from an intermediate server that another message – which had a whopping score of 14.6 – was rejected as spam. This 'bounce' message included the original message and therefore inherited some of

its characteristics, such as a low text-to-image ratio, which pushed the score over the threshold of 5.

Here is another example, this time of an important e-mail I had been expecting for some time, only for it to be found languishing in my spam folder:

```
2.5 URI_NOVOWEL URI:        URI hostname has long non-vowel sequence
3.1 FROM_DOMAIN_NOVOWEL     From: domain has series of non-vowel letters
```

The e-mail in question concerned a paper that one of the members of my group and I had submitted to the European Conference on Machine Learning (ECML) and the European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), which have been jointly organised since 2001. The 2008 instalment of these conferences used the internet domain `www.ecmlpkdd2008.org` – a perfectly respectable one, as machine learning researchers know, but also one with eleven 'non-vowels' in succession – enough to raise SpamAssassin's suspicion! The example demonstrates that the importance of a SpamAssassin test can be different for different users. Machine learning is an excellent way of creating software that adapts to the user.

How does SpamAssassin determine the scores or 'weights' for each of the dozens of tests it applies? This is where machine learning comes in. Suppose we have a large 'training set' of e-mails which have been hand-labelled spam or ham, and we know the results of all the tests for each of these e-mails. The goal is now to come up with a weight for every test, such that all spam e-mails receive a score above 5, and all ham e-mails get less than 5. As we will discuss later in the book, there are a number of machine learning techniques that solve exactly this problem. For the moment, a simple example will illustrate the main idea.

Example 1 (Linear classification). Suppose we have only two tests and four training e-mails, one of which is spam (see Table 1). Both tests succeed for the spam e-mail; for one ham e-mail neither test succeeds, for another the first test succeeds and the second doesn't, and for the third ham e-mail the first test fails and the second succeeds. It is easy to see that assigning both tests a weight of 4 correctly 'classifies' these four e-mails into spam and ham. In the mathematical notation introduced in Background 1 we

could describe this classifier as $4x_1 + 4x_2 > 5$ or $(4,4) \cdot (x_1, x_2) > 5$. In fact, any weight between 2.5 and 5 will ensure that the threshold of 5 is only exceeded when both tests succeed. We could even consider assigning different weights to the tests – as long as each weight is less than 5 and their sum exceeds 5 – although it is hard to see how this could be justified by the training data.

E-mail	x_1	x_2	Spam?	$4x_1 + 4x_2$
1	1	1	1	8
2	0	0	0	0
3	1	0	0	4
4	0	1	0	4

Table 1. A small training set for SpamAssassin. The columns marked x_1 and x_2 indicate the results of two tests on four different e-mails. The fourth column indicates which of the e-mails are spam. The right-most column demonstrates that by thresholding the function $4x_1+4x_2$ at 5, we can separate spam from ham.

But what does this have to do with learning, I hear you ask? It is just a mathematical problem, after all. That may be true, but it does not appear unreasonable to say that SpamAssassin learns to recognise spam e-mail from examples and counter-examples. Moreover, the more training data is made available, the better SpamAssassin will become at this task. The notion of performance improving with experience is central to most, if not all, forms of machine learning. We will use the following general definition: *Machine learning is the systematic study of algorithms and systems that improve their knowledge or performance with experience.* In the case of SpamAssassin, the 'experience' it learns from is some correctly labelled training data, and 'performance' refers to its ability to recognise spam e-mail. A schematic view of how machine learning feeds into the spam e-mail classification task is given in Figure 2. In other machine learning problems experience may take a different form, such as corrections of mistakes, rewards when a certain goal is reached, among many others. Also note that, just as is the case with human learning, machine learning is not always directed at improving performance on a certain task, but may more generally result in improved knowledge.

There are a number of useful ways in which we can express the SpamAssassin classifier in mathematical notation. If we denote the result of the i -th test for a given e-mail as x_i, where $x_i = 1$ if the test succeeds and 0 otherwise, and we denote the weight of the i -th test as w_i, then the total score of an e-mail can be expressed as $\sum_{i=1}^{n} w_i x_i$, making use of the fact that w_i contributes to the sum only if $x_i = 1$, i.e., if the test succeeds for the e-mail. Using t for the threshold above which an e-mail is classified as spam (5 in our example), the 'decision rule' can be written as $\sum_{i=1}^{n} w_i x_i > t$.

Notice that the left-hand side of this inequality is linear in the x_i variables, which essentially means that increasing one of the x_i by a certain amount, say δ, will change the sum by an amount ($w_i\delta$) that is independent of the value of x_i. This wouldn't be true if x_i appeared squared in the sum, or with any exponent other than 1.

The notation can be simplified by means of linear algebra, writing **w** for the vector of weights $(w_1,\ldots,w_n)$ and **x** for the vector of test results $(x_1,\ldots,x_n)$. The above inequality can then be written using a dot product: $\mathbf{w} \cdot \mathbf{x} > t$. Changing the inequality to an equality $\mathbf{w} \cdot \mathbf{x} = t$, we obtain the 'decision boundary', separating spam from ham. The decision boundary is a plane (a 'straight' surface) in the space spanned by the x_i variables because of the linearity of the left-hand side. The vector **w** is perpendicular to this plane and points in the direction of spam. Figure 1 visualises this for two variables.

It is sometimes convenient to simplify notation further by introducing an extra constant 'variable' $x_0 = 1$, the weight of which is fixed to $w_0 = -t$. The extended data point is then $\mathbf{x}^\circ = (1, x_1,\ldots,x_n)$ and the extended weight vector is $\mathbf{w}^\circ = (-t, w_1,\ldots, w_n)$, leading to the decision rule $\mathbf{w}^\circ \cdot \mathbf{x}^\circ > 0$ and the decision boundary $\mathbf{w}^\circ \cdot \mathbf{x}^\circ = 0$. Thanks to these so-called homogeneous coordinates the decision boundary passes through the origin of the extended coordinate system, at the expense of needing an additional dimension (but note that this doesn't really affect the data, as all data points and the 'real' decision boundary live in the plane $x_0 = 1$).

Background 1. SpamAssassin in mathematical notation. In boxes such as these, I will briefly remind you of useful concepts and notation. If some of these are unfamiliar, you will need to spend some time reviewing them – using other books or online resources such as `www.wikipedia.org` or `mathworld.wolfram.com` – to fully appreciate the rest of the book.

Figure 1. An example of linear classification in two dimensions. The straight line separates the positives from the negatives. It is defined by $\mathbf{w} \cdot \mathbf{x}_i = t$, where $\mathbf{w}$ is a vector perpendicular to the decision boundary and pointing in the direction of the positives, t is the decision threshold, and $\mathbf{x}_i$ points to a point on the decision boundary. In particular, $\mathbf{x}_0$ points in the same direction as $\mathbf{w}$, from which it follows that $\mathbf{w} \cdot \mathbf{x}_0 = ||\mathbf{w}||\ ||\mathbf{x}_0|| = t$ ($||\mathbf{x}||$ denotes the length of the vector $\mathbf{x}$). The decision boundary can therefore equivalently be described by $\mathbf{w} \cdot (\mathbf{x}–\mathbf{x}_0) = 0$, which is sometimes more convenient. In particular, this notation makes it clear that it is the orientation but not the length of $\mathbf{w}$ that determines the location of the decision boundary.

Figure 2. At the top we see how SpamAssassin approaches the spam e-mail classification task: the text of each e-mail is converted into a data point by means of SpamAssassin's built-in tests, and a linear classifier is applied to obtain a 'spam or ham' decision. At the bottom (in blue) we see the bit that is done by machine learning.

We have already seen that a machine learning problem may have several solutions, even a problem as simple as the one from Example 1. This raises the question of how we choose among these solutions. One way to think about this is to realise that we don't really care that much about performance on training data – we already know which of those e-mails are spam! What we care about is whether *future* e-mails are going to be classified correctly. While this appears to

lead into a vicious circle – in order to know whether an e-mail is classified correctly I need to know its true class, but as soon as I know its true class I don't need the classifier anymore – it is important to keep in mind that good performance on training data is only a means to an end, not a goal in itself. In fact, trying too hard to achieve good performance on the training data can easily lead to a fascinating but potentially damaging phenomenon called *overfitting*.

Example 2 (Overfitting). Imagine you are preparing for your *Machine Learning 101* exam. Helpfully, Professor Flach has made previous exam papers and their worked answers available online. You begin by trying to answer the questions from previous papers and comparing your answers with the model answers provided. Unfortunately, you get carried away and spend all your time on memorising the model answers to all past questions. Now, if the upcoming exam completely consists of past questions, you are certain to do very well. But if the new exam asks different questions about the same material, you would be ill-prepared and get a much lower mark than with a more traditional preparation. In this case, one could say that you were *overfitting* the past exam papers and that the knowledge gained didn't *generalise* to future exam questions.

Generalisation is probably the most fundamental concept in machine learning. If the knowledge that SpamAssassin has gleaned from its training data carries over – generalises – to your e-mails, you are happy; if not, you start looking for a better spam filter. However, overfitting is not the only possible reason for poor performance on new data. It may just be that the training data used by the SpamAssassin programmers to set its weights is not representative for the kind of e-mails you get. Luckily, this problem does have a solution: use different training data that exhibits the same characteristics, if possible actual spam and ham e-mails that you have personally received. Machine learning is a great technology for adapting the behaviour of software to your own personal circumstances, and many spam e-mail filters allow the use of your own training data.

So, if there are several possible solutions, care must be taken to select one that doesn't overfit the data. We will discuss several ways of doing that in this book.

What about the opposite situation, if there isn't a solution that perfectly classifies the training data? For instance, imagine that e-mail 2 in Example 1, the one for which both tests failed, was spam rather than ham – in that case, there isn't a single straight line separating spam from ham (you may want to convince yourself of this by plotting the four e-mails as points in a grid, with x_1 on one axis and x_2 on the other). There are several possible approaches to this situation. One is to ignore it: that e-mail may be atypical, or it may be mis-labelled (so-called *noise)*. Another possibility is to switch to a more expressive type of classifier. For instance, we may introduce a second decision rule for spam: in addition to $4x_1 + 4x_2 > 5$ we could alternatively have $4x_1 + 4x_2 < 1$. Notice that this involves learning a different threshold, and possibly a different weight vector as well. This is only really an option if there is enough training data available to reliably learn those additional parameters.

Linear classification, SpamAssassin-style, may serve as a useful introduction, but this book would have been a lot shorter if that was the only type of machine learning. What about learning not just the weights for the tests, but also the tests themselves? How do we decide if the text-to-image ratio is a good test? Indeed, how do we come up with such a test in the first place? This is an area where machine learning has a lot to offer.

One thing that may have occurred to you is that the SpamAssassin tests considered so far don't appear to take much notice of the *contents* of the e-mail. Surely words and phrases like 'Viagra', 'free iPod' or 'confirm your account details' are good spam indicators, while others – for instance, a particular nickname that only your friends use – point in the direction of ham. For this reason, many spam e-mail filters employ text classification techniques. Broadly speaking, such techniques maintain a vocabulary of words and phrases that are potential spam or ham indicators. For each of those words and phrases, statistics are collected from a training set. For instance, suppose that the word 'Viagra' occurred in four spam e-mails and in one ham e-mail. If we then encounter a new e-mail that contains the word 'Viagra', we might reason that the odds that this e-mail is spam are 4:1, or the probability of it being spam is 0.80 and the probability of it being ham is 0.20 (see Background 2 for some basic notions of probability theory).

The situation is slightly more subtle than you might realise because we have to take into account the prevalence of spam. Suppose, for the sake of argument, that

I receive on average one spam e-mail for every six ham e-mails (I wish!). This means that I would estimate the odds of the next e-mail coming in being spam as 1:6, i.e., non-negligible but not very high either. If I then learn that the e-mail contains the word 'Viagra', which occurs four times as often in spam as in ham, I somehow need to combine these two odds. As we shall see later, Bayes' rule tells us that we should simply multiply them: 1:6 times 4:1 is 4:6, corresponding to a spam probability of 0.4. In other words, despite the occurrence of the word 'Viagra', the safest bet is still that the e-mail is ham. That doesn't make sense, or does it?

Probabilities involve 'random variables' that describe outcomes of 'events'. These events are often hypothetical and therefore probabilities have to be estimated. For example, consider the statement '42% of the UK population approves of the current Prime Minister'. The only way to know this for certain is to ask everyone in the UK, which is of course unfeasible. Instead, a (hopefully representative) sample is queried, and a more correct statement would then be '42% of a sample drawn from the UK population approves of the current Prime Minister', or 'the proportion of the UK population approving of the current Prime Minister is estimated at 42%'. Notice that these statements are formulated in terms of proportions or 'relative frequencies'; a corresponding statement expressed in terms of probabilities would be 'the probability that a person uniformly drawn from the UK population approves of the current Prime Minister is estimated at 0.42'. The event here is 'this random person approves of the PM'.

The 'conditional probability' $P(A|B)$ is the probability of event A happening given that event B happened. For instance, the approval rate of the Prime Minister may differ for men and women. Writing $P(\text{PM})$ for the probability that a random person approves of the Prime Minister and $P(\text{PM}|\text{woman})$ for the probability that a random woman approves of the Prime Minister, we then have that $P(\text{PM}|\text{woman}) = P(\text{PM, woman})/P(\text{woman})$, where $P(\text{PM, woman})$ is the probability of the 'joint event' that a random person both approves of the PM and is a woman, and $P(\text{woman})$ is the probability that a random person is a woman (i.e., the proportion of women in the UK population).

Other useful equations include $P(A, B) = P(A|B)P(B) = P(B|A)P(A)$ and $P(A|B) = P(B|A)P(A)/P(B)$. The latter is known as 'Bayes' rule' and will play an important role in this book. Notice that many of these equations can be extended to more than two random variables, *e.g.* the 'chain rule of probability': $P(A, B, C, D) = P(A|B, C, D)P(B|C, D)P(C|D)P(D)$.

Two events A and B are independent if $P(A|B) = P(A)$, i.e., if knowing that B happened doesn't change the probability of A happening. An equivalent formulation is $P(A,B) = P(A)P(B)$. In general, multiplying probabilities involves the assumption that the corresponding events are independent.

The 'odds' of an event is the ratio of the probability that the event happens and the probability that it doesn't happen. That is, if the probability of a particular event happening is p, then the corresponding odds are $o = p/(1 - p)$. Conversely, we have that $p = o/(o + 1)$. So, for example, a probability of 0.8 corresponds to odds of 4:1, the opposite odds of 1:4 give probability 0.2, and if the event is as likely to occur as not then the probability is 0.5 and the odds are 1:1. While we will most often use the probability scale, odds are sometimes more convenient because they are expressed on a multiplicative scale.

Background 2. The basics of probability.

The way to make sense of this is to realise that you are combining two independent pieces of evidence, one concerning the prevalence of spam, and the other concerning the occurrence of the word ‘Viagra’. These two pieces of evidence pull in opposite directions, which means that it is important to assess their relative strength. What the numbers tell you is that, in order to overrule the fact that spam is relatively rare, you need odds of at least 6:1. ‘Viagra’ on its own is estimated at 4:1, and therefore doesn’t pull hard enough in the spam direction to warrant the conclusion that the e-mail is in fact spam. What it does do is make the conclusion ‘this e-mail is ham’ a lot less certain, as its probability drops from 6/7 = 0.86 to 6/10 = 0.60.

The nice thing about this ‘Bayesian’ classification scheme is that it can be repeated if you have further evidence. For instance, suppose that the odds in favour of spam associated with the phrase ‘blue pill’ is estimated at 3:1 (i.e., there are three times more spam e-mails containing the phrase than there are ham e-mails), and suppose our email contains both ‘Viagra’ and ‘blue pill’, then the combined odds are 4:1 times 3:1 is 12:1, which is ample to outweigh the 1:6 odds associated with the low prevalence of spam (total odds are 2:1, or a spam probability of 0.67, up from 0.40 without the ‘blue pill’).

The advantage of not having to estimate and manipulate joint probabilities is that we can handle large numbers of variables. Indeed, the vocabulary of a typical Bayesian spam filter or text classifier may contain some 10 000 terms.[2] So, instead of manually crafting a small set of ‘features’ deemed relevant or predictive by an expert, we include a much larger set and let the classifier figure out which features are important, and in what combinations.

It should be noted that by multiplying the odds associated with ‘Viagra’ and ‘blue pill’, we are implicitly assuming that they are independent pieces of information. This is obviously not true: if we know that an e-mail contains the phrase ‘blue pill’, we are not really surprised to find out that it also contains the word ‘Viagra’. In probabilistic terms:

- the probability P(Viagra|blue pill) will be close to 1;
- hence the joint probability P(Viagra, blue pill) will be close to P(blue pill)
- hence the odds of spam associated with the two phrases ‘Viagra’ and ‘blue pill’ will not differ much from the odds associated with ‘blue pill’ on its own.

Put differently, by multiplying the two odds we are counting what is essentially one piece of information twice. The product odds of 12:1 is almost certainly an overestimate, and the real joint odds may be not more than, say, 5:1.

We appear to have painted ourselves into a corner here. In order to avoid overcounting we need to take joint occurrences of phrases into account; but this is only feasible computationally if we define the problem away by assuming them to be independent. What we want seems to be closer to a rule-based model such as the following:

1. if the e-mail contains the word 'Viagra' then estimate the odds of spam as 4:1;

2. otherwise, if it contains the phrase 'blue pill' then estimate the odds of spam as 3:1;

3. otherwise, estimate the odds of spam as 1:6.

The first rule covers all e-mails containing the word 'Viagra', regardless of whether they contain the phrase 'blue pill', so no overcounting occurs. The second rule *only* covers e-mails containing the phrase 'blue pill' but not the word 'Viagra', by virtue of the 'otherwise' clause. The third rule covers all remaining e-mails: those which neither contain neither 'Viagra' nor 'blue pill'.

The essence of such rule-based classifiers is that they don't treat all e-mails in the same way but work on a case-by-case basis. In each case they only invoke the most relevant features. Cases can be defined by several nested features:

1. Does the e-mail contain the word 'Viagra'?
   - (a) If so: Does the e-mail contain the word 'blue pill'?
     - i. If so: estimate the odds of spam as 5:1.
     - ii. If not: estimate the odds of spam as 4:1.
   - (b) If not: Does the e-mail contain the word 'lottery'?
     - i. If so: estimate the odds of spam as 3:1.
     - ii. If not: estimate the odds of spam as 1:6.

These four cases are characterised by logical conditions such as 'the e-mail

contains the word “Viagra” but not the phrase “blue pill” ’. Effective and efficient algorithms exist for identifying the most predictive feature combinations and organise them as rules or trees, as we shall see later.

We have now seen three practical examples of machine learning in spam e-mail recognition. Machine learners call such a task *binary classification*, as it involves assigning objects (e-mails) to one of two classes: spam or ham. This task is achieved by describing each e-mail in terms of a number of variables or features. In the SpamAssassin example these features were handcrafted by an expert in spam filtering, while in the Bayesian text classification example we employed a large vocabulary of words. The question is then how to use the features to distinguish spam from ham. We have to somehow figure out a connection between the features and the class – machine learners call such a connection a *model* – by analysing a *training set* of e-mails already labelled with the correct class.

Figure 3. An overview of how machine learning is used to address a given task. A task (red box) requires an appropriate mapping – a model – from data described by features to outputs. Obtaining such amapping fromtraining data is what constitutes a learning problem (blue box).

In the SpamAssassin example we came up with a linear equation of the form $\sum_{i=1}^{n} w_i x_i > t$, where the x_i denote the 0–1 valued or ‘Boolean’ featur indicating whether the i -th test succeeded for the e-mail, w_i are the feature weights learned from the training set, and t is the threshold above which e

mails are classified as spam.

- In the Bayesian example we used a decision rule that can be written as $\prod_{i=0}^{n} o_i > 1$, where $o_i = P(\text{spam}|x_i)/P(\text{ham}|x_i)$, $1 \leq i \leq n$, are the odds of spa associated with each word x_i in the vocabulary and $o_0 = P(\text{spam})/P(\text{ham})$ are the prior odds, all of which are estimated from the training set.
- In the rule-based example we built logical conditions that identify subsets of the data that are sufficiently similar to be labelled in a particular way.

Here we have, then, the main ingredients of machine learning: tasks, models and features. Figure 3 shows how these ingredients relate. If you compare this figure with Figure 2, you'll see how the model has taken centre stage, rather than merely being a set of parameters of a classifier otherwise defined by the features. We need this flexibility to incorporate the very wide range of models in use in machine learning. It is worth emphasising the distinction between tasks and learning problems: *tasks are addressed by models, whereas learning problems are solved by learning algorithms that produce models*. While the distinction is widely recognised, terminology may vary: for instance, you may find that other authors use the term 'learning task' for what we call a learning problem.

In summary, one could say that *machine learning is concerned with using the right features to build the right models that achieve the right tasks*. I call these 'ingredients' to emphasise that they come in many different forms, and need to be chosen and combined carefully to create a successful 'meal': what machine learners call an *application* (the construction of a model that solves a practical task, by means of machine learning methods, using data from the task domain). Nobody can be a good chef without a thorough understanding of the ingredients at his or her disposal, and the same holds for a machine learning expert. Our main ingredients of tasks, models and features will be investigated in full detail from Chapter 2 onwards; first we will enjoy a little 'taster menu' when I serve up a range of examples in the next chapter to give you some more appreciation of these ingredients.

---

[1] Spam, a contraction of 'spiced ham', is the name of a meat product that achieved notoriety by being ridiculed in a 1970 episode of *Monty Python's Flying Circus*.

[2] In fact, phrases consisting of multiple words are usually decomposed into their constituent words, such that $P(\text{blue pill})$ is estimated as $P(\text{blue})P(\text{pill})$.

CHAPTER *1*

The ingredients of machine learning

MACHINE LEARNING IS ALL ABOUT using the right features to build the right models that achieve the right tasks – this is the slogan, visualised in Figure 3 on p.11, with which we ended the Prologue. In essence, *features* define a 'language' in which we describe the relevant objects in our domain, be they e-mails or complex organic molecules. We should not normally have to go back to the domain objects themselves once we have a suitable feature representation, which is why features play such an important role in machine learning. We will take a closer look at them in Section 1.3. A *task* is an abstract representation of a problem we want to solve regarding those domain objects: the most common form of these is classifying them into two or more classes, but we shall encounter other tasks throughout the book. Many of these tasks can be represented as a mapping from data points to outputs. This mapping or *model* is itself produced as the output of a machine learning algorithm applied to training data; there is a wide variety of models to choose from, as we shall see in Section 1.2.

We start this chapter by discussing tasks, the problems that can be solved with machine learning. No matter what variety of machine learning models you may encounter, you will find that they are designed to solve one of only a small number of tasks and use only a few different types of features. One could say that *models lend the machine learning field diversity, but tasks and features give it unity*.

1.1 Tasks: the problems that can be solved with machine learning

Spam e-mail recognition was described in the Prologue. It constitutes a binary classification task, which is easily the most common task in machine learning which figures heavily throughout the book. One obvious variation is to consider classification problems with more than two classes. For instance, we may want

to distinguish different kinds of ham e-mails, e.g., work-related e-mails and private messages. We could approach this as a combination of two binary classification tasks: the first task is to distinguish between spam and ham, and the second task is, among ham e-mails, to distinguish between work-related and private ones. However, some potentially useful information may get lost this way, as some spam e-mails tend to look like private rather than work-related messages. For this reason, it is often beneficial to view *multi-class classification* as a machine learning task in its own right. This may not seem a big deal: after all, we still need to learn a model to connect the class to the features. However, in this more general setting some concepts will need a bit of rethinking: for instance, the notion of a decision boundary is less obvious when there are more than two classes.

Sometimes it is more natural to abandon the notion of discrete classes altogether and instead predict a real number. Perhaps it might be useful to have an assessment of an incoming e-mail's urgency on a sliding scale. This task is called *regression*, and essentially involves learning a real-valued function from training examples labelled with true function values. For example, I might construct such a training set by randomly selecting a number of e-mails from my inbox and labelling them with an urgency score on a scale of 0 (ignore) to 10 (immediate action required). This typically works by choosing a class of functions (e.g., functions in which the function value depends linearly on some numerical features) and constructing a function which minimises the difference between the predicted and true function values. Notice that this is subtly different from SpamAssassin learning a real-valued spam score, where the training data are labelled with classes rather than 'true' spam scores. This means that SpamAssassin has less information to go on, but it also allows us to interpret SpamAssassin's score as an assessment of how far it thinks an e-mail is removed from the decision boundary, and therefore as a measure of confidence in its own prediction. In a regression task the notion of a decision boundary has no meaning, and so we have to find other ways to express a models's confidence in its real-valued predictions.

Both classification and regression assume the availability of a training set of examples labelled with true classes or function values. Providing the true labels for a data set is often labour-intensive and expensive. Can we learn to distinguish spam from ham, or work e-mails from private messages, without a labelled training set? The answer is: yes, up to a point. The task of grouping data without prior information on the groups is called *clustering*. Learning from unlabelled data is called *unsupervised learning* and is quite distinct from *supervised learning*, which requires labelled training data. A typical clustering algorithm

works by assessing the similarity between instances (the things we're trying to cluster, e.g., e-mails) and putting similar instances in the same cluster and 'dissimilar' instances in different clusters.

Example 1.1 (Measuring similarity). If our e-mails are described by word occurrence features as in the text classification example, the similarity of e-mails would be measured in terms of the words they have in common. For instance, we could take the number of common words in two e-mails and divide it by the number of words occurring in either e-mail (this measure is called the *Jaccard coefficient*). Suppose that one e-mail contains 42 (different) words and another contains 112 words, and the two e-mails have 23 words in common, then their similarity would be $\frac{23}{42+112-23} = \frac{23}{130} = 0.18$. We can then cluster our e-mails into groups, such that the average similarity of an e-mail to the other e-mails in its group is much larger than the average similarity to e-mails from other groups. While it wouldn't be realistic to expect that this would result in two nicely separated clusters corresponding to spam and ham – there's no magic here – the clusters may reveal some interesting and useful structure in the data. It may be possible to identify a particular kind of spam in this way, if that subgroup uses a vocabulary, or language, not found in other messages.

There are many other patterns that can be learned from data in an unsupervised way. *Association rules* are a kind of pattern that are popular in marketing applications, and the result of such patterns can often be found on online shopping web sites. For instance, when I looked up the book *Kernel Methods for Pattern Analysis* by John Shawe-Taylor and Nello Cristianini on `www.amazon.co.uk`, I was told that 'Customers Who Bought This Item Also Bought' – and 34 more suggestions. Such associations are found by data mining algorithms that zoom in on items that frequently occur together. These algorithms typically work by only considering items that occur a minimum number of times (because you wouldn't want your suggestions to be based on a single customer that happened to buy these 39 books together!). More interesting associations could be found by considering multiple items in your shopping basket. There exist many other types of associations that can be learned and exploited, such as correlations

between real-valued variables.

- *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods* by Nello Cristianini and John Shawe-Taylor;
- *Pattern Recognition and Machine Learning* by Christopher Bishop;
- *The Elements of Statistical Learning: Data Mining, Inference and Prediction* by Trevor Hastie, Robert Tibshirani and Jerome Friedman;
- *Pattern Classification* by Richard Duda, Peter Hart and David Stork;

Looking for structure

Like all other machine learning models, patterns are a manifestation of underlying structure in the data. Sometimes this structure takes the form of a single *hidden* or *latent variable*, much like unobservable but nevertheless explanatory quantities in physics, such as energy. Consider the following matrix:

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 2 & 2 \\ 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 2 \\ 1 & 0 & 1 & 1 \\ 0 & 2 & 2 & 3 \end{pmatrix}$$

Imagine these represent ratings by six different people (in rows), on a scale of 0 to 3, of four different films – say *The Shawshank Redemption, The Usual Suspects, The Godfather, The Big Lebowski*, (in columns, from left to right). *The Godfather* seems to be the most popular of the four with an average rating of 1.5, and *The Shawshank Redemption* is the least appreciated with an average rating of 0.5. Can you see any structure in this matrix?

If you are inclined to say no, try to look for columns or rows that are combinations of other columns or rows. For instance, the third column turns out to be the sum of the first and second columns. Similarly, the fourth row is the sum of the first and second rows. What this means is that the fourth person combines the ratings of the first and second person. Similarly, *The Godfather's* ratings are the sum of the ratings of the first two films. This is made more explicit by writing the matrix as the following product:

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 2 & 2 \\ 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 2 \\ 1 & 0 & 1 & 1 \\ 0 & 2 & 2 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

You might think I just made matters worse – instead of one matrix we now have three! However, notice that the first and third matrix on the right-hand side are now Boolean, and the middle one is diagonal (all off-diagonal entries are zero). Moreover, these matrices have a very natural interpretation in terms of film *genres*. The right-most matrix associates films (in columns) with genres (in rows): *The Shawshank Redemption* and *The Usual Suspects* belong to two different genres, say drama and crime, *The Godfather* belongs to both, and *The Big Lebowski* is a crime film and also introduces a new genre (say comedy). The tall, 6-by-3 matrix then expresses people's preferences in terms of genres: the first, fourth and fifth person like drama, the second, fourth and fifth person like crime films, and the third, fifth and sixth person like comedies. Finally, the middle matrix states that the crime genre is twice as important as the other two genres in terms of determining people's preferences.

Methods for discovering hidden variables such as film genres really come into their own when the number of values of the hidden variable (here: the number of genres) is much smaller than the number of rows and columns of the original matrix. For instance, at the time of writing `www.imdb.com` lists about 630 000 rated films with 4 million people voting, but only 27 film categories (including the ones above). While it would be naive to assume that film ratings can be completely broken down by genres – genre boundaries are often diffuse, and someone may only like comedies made by the Coen brothers – this kind of ☞ *matrix decomposition* can often reveal useful hidden structure. It will be further examined in Chapter 10.

This is a good moment to summarise some terminology that we will be using. We have already seen the distinction between supervised learning from labelled data and unsupervised learning from unlabelled data. We can similarly draw a distinction between whether the model output involves the target variable or not: we call it a *predictive model* if it does, and a *descriptive model* if it does not. This leads to the four different machine learning settings summarised in Table 1.1.

☞ The most common setting is supervised learning of predictive models – in fact, this is what people commonly mean when they refer to supervised

learning. Typical tasks are classification and regression.

- It is also possible to use labelled training data to build a descriptive model that is not primarily intended to predict the target variable, but instead identifies, say, subsets of the data that behave differently with respect to th target variable. This example of supervised learning of a descriptive mode is called *subgroup discovery*; we will take a closer look at it in Section 6.3.

- Descriptive models can naturally be learned in an unsupervised setting, an we have just seen a few examples of that (clustering, association rule discovery and matrix decomposition). This is often the implied setting when people talk about unsupervised learning.

- A typical example of unsupervised learning of a predictive model occurs when we cluster data with the intention of using the clusters to assign clas labels to new data. We will call this *predictive clustering* to distinguish it from the previous, *descriptive* form of clustering.

	Predictive model	*Descriptive model*
Supervised learning	classification, regression	subgroup discovery
Unsupervised learning	predictive clustering	descriptive clustering, association rule discovery

Table 1.1. An overview of different machine learning settings. The rows refer to whether the training data is labelled with a target variable, while the columns indicate whether the models learned are used to predict a target variable or rather describe the given data.

Although we will not cover it in this book, it is worth pointing out a fifth setting of *semi-supervised learning* of predictive models. In many problem domains data is cheap, but labelled data is expensive. For example, in web page classification you have the whole world-wide web at your disposal, but constructing a labelled training set is a painstaking process. One possible approach in semi-supervised learning is to use a small labelled training set to build an initial model, which is then refined using the unlabelled data. For example, we could use the initial model to make predictions on the unlabelled data, and use the most confident predictions as new training data, after which we retrain the model on this enlarged training set.

Evaluating performance on a task

An important thing to keep in mind with all these machine learning problems is that they don't have a 'correct' answer. This is different from many other problems in computer science that you might be familiar with. For instance, if you sort the entries in your address book alphabetically on last name, there is only one correct result (unless two people have the same last name, in which case you can use some other field as tie-breaker, such as first name or age). This is not to say that there is only one way of achieving that result – on the contrary, there is a wide range of sorting algorithms available: insertion sort, bubblesort, quicksort, to name but a few. If we were to compare the performance of these algorithms, it would be in terms of how fast they are, and how much data they could handle: e.g., we could test this experimentally on real data, or analyse it using computational complexity theory. However, what we wouldn't do is compare different algorithms with respect to the correctness of the result, because an algorithm that isn't guaranteed to produce a sorted list every time is useless as a sorting algorithm.

Things are different in machine learning (and not just in machine learning: see Background 1.1). We can safely assume that the perfect spam e-mail filter doesn't exist – if it did, spammers would immediately 'reverse engineer' it to find out ways to trick the spam filter into thinking a spam e-mail is actually ham. In many cases the data is 'noisy' – examples may be mislabelled, or features may contain errors – in which case it would be detrimental to try too hard to find a model that correctly classifies the training data, because it would lead to overfitting, and hence wouldn't generalise to new data. In some cases the features used to describe the data only give an indication of what their class might be, but don't contain enough 'signal' to predict the class perfectly. For these and other reasons, machine learners take performance evaluation of learning algorithms very seriously, which is why it will play a prominent role in this book. We need to have some idea of how well an algorithm is expected to perform on new data, not in terms of runtime or memory usage – although this can be an issue too – but in terms of classification performance (if our task is a classification task).

Suppose we want to find out how well our newly trained spam filter does. One thing we can do is count the number of correctly classified e-mails, both spam and ham, and divide that by the total number of examples to get a proportion which is called the *accuracy* of the classifier. However, this doesn't indicate whether overfitting is occurring. A better idea would be to use only 90% (say) of

the data for training, and the remaining 10% as a *test set*. If overfitting occurs, the test set performance will be considerably lower than the training set performance. However, even if we select the test instances randomly from the data, every once in a while we may get lucky, if most of the test instances are similar to training instances – or unlucky, if the test instances happen to be very non-typical or noisy. In practice this train–test split is therefore repeated in a process called ☞ *cross-validation*, further discussed in Chapter 12. This works as follows: we randomly divide the data in ten parts of equal size, and use nine parts for training and one part for testing. We do this ten times, using each part once for testing. At the end, we compute the average test set performance (and usually also its standard deviation, which is useful to determine whether small differences in average performance of different learning algorithms are meaningful). Cross-validation can also be applied to other supervised learning problems, but unsupervised learning methods typically need to be evaluated differently.

In Chapters 2 and 3 we will take a much closer look at the various tasks that can be approached using machine learning methods. In each case we will define the task and look at different variants. We will pay particular attention to evaluating performance of models learned to solve those tasks, because this will give us considerable additional insight into the nature of the tasks.

Long before machine learning came into existence, philosophers knew that generalising from particular cases to general rules is not a well-posed problem with well-defined solutions. Such inference by generalisation is called *induction* and is to be contrasted with *deduction*, which is the kind of reasoning that applies to problems with well-defined correct solutions. There are many versions of this so-called *problem of induction*. One version is due to the eighteenth-century Scottish philosopher David Hume, who claimed that the only justification for induction is itself inductive: since it appears to work for certain inductive problems, it is expected to work for all inductive problems. This doesn't just say that induction cannot be deductively justified but that its justification is circular, which is much worse.

A related problem is stated by the *no free lunch theorem*, which states that no learning algorithm can outperform another when evaluated over all possible classification problems, and thus the performance of any learning

algorithm, over the set of all possible learning problems, is no better than random guessing. Consider, for example, the 'guess the next number' questions popular in psychological tests: what comes after 1, 2, 4, 8, …? If all number sequences are equally likely, then there is no hope that we can improve – on average – on random guessing (I personally always answer '42' to such questions). Of course, some sequences are very much more likely than others, at least in the world of psychological tests. Likewise, the distribution of learning problems in the real world is highly non-uniform. The way to escape the curse of the no free lunch theorem is to find out more about this distribution and exploit this knowledge in our choice of learning algorithm.

Background 1.1. Problems of induction and free lunches.

1.2 Models: the output of machine learning

Models form the central concept in machine learning as they are what is being learned from the data, in order to solve a given task. There is a considerable – not to say bewildering – range of machine learning models to choose from. One reason for this is the ubiquity of the tasks that machine learning aims to solve: classification, regression, clustering, association discovery, to name but a few. Examples of each of these tasks can be found in virtually every branch of science and engineering. Mathematicians, engineers, psychologists, computer scientists and many others have discovered – and sometimes rediscovered – ways to solve these tasks. They have all brought their specific background to bear, and consequently the principles underlying these models are also diverse. My personal view is that this diversity is a good thing as it helps to make machine learning the powerful and exciting discipline it is. It doesn't, however, make the task of writing a machine learning book any easier! Luckily, a few common themes can be observed, which allow me to discuss machine learning models in a somewhat more systematic way. I will discuss three groups of models: geometric models, probabilistic models, and logical models. These groupings are not meant to be mutually exclusive, and sometimes a particular kind of model has, for instance, both a geometric and a probabilistic interpretation. Nevertheless, it provides a good starting point for our purposes.

Geometric models

The *instance space* is the set of all possible or describable instances, whether they are present in our data set or not. Usually this set has some geometric structure. For instance, if all features are numerical, then we can use each feature as a coordinate in a Cartesian coordinate system. A *geometric model* is constructed directly in instance space, using geometric concepts such as lines, planes and distances. For instance, the linear classifier depicted in Figure 1 on p.5 is a geometric classifier. One main advantage of geometric classifiers is that they are easy to visualise, as long as we keep to two or three dimensions. It is important to keep in mind, though, that a Cartesian instance space has as many coordinates as there are features, which can be tens, hundreds, thousands, or even more. Such high-dimensional spaces are hard to imagine but are nevertheless very common in machine learning. Geometric concepts that potentially apply to high-dimensional spaces are usually prefixed with 'hyper-': for instance, a decision boundary in an unspecified number of dimensions is called a *hyperplane*.

If there exists a linear decision boundary separating the two classes, we say that the data is *linearly separable.* As we have seen, a linear decision boundary is defined by the equation $\mathbf{w} \cdot \mathbf{x} = t$, where $\mathbf{w}$ is a vector perpendicular to the decision boundary, $\mathbf{x}$ points to an arbitrary point on the decision boundary, and t is the decision threshold. A good way to think of the vector $\mathbf{w}$ is as pointing from the 'centre of mass' of the negative examples, $\mathbf{n}$, to the centre of mass of the positives $\mathbf{p}$. In other words, $\mathbf{w}$ is proportional (or equal) to $\mathbf{p} - \mathbf{n}$. One way to calculate these centres of mass is by averaging. For instance, if P is a set of n positive examples, then we can define $\mathbf{p} = \frac{1}{n}\sum_{x \in P} \mathbf{x}$, and similarly for $\mathbf{n}$. By setting the decision threshold appropriately, we can intersect the line from $\mathbf{n}$ to $\mathbf{p}$ half-way (Figure 1.1). We will call this the *basic linear classifier* in this book.[1] It has the advantage of simplicity, being defined in terms of addition, subtraction and rescaling of examples only (in other words, $\mathbf{w}$ is a *linear combination* of the examples). Indeed, under certain additional assumptions about the data it is the best thing we can hope to do, as we shall see later. However, if those assumptions do not hold, the basic linear classifier can perform poorly – for instance, note that it may not perfectly separate the positives from the negatives, even if the data is linearly separable.

Figure 1.1. The basic linear classifier constructs a decision boundary by half-way intersecting the line between the positive and negative centres of mass. It is described by the equation $\mathbf{w} \cdot \mathbf{x} = t$, with $\mathbf{w} = \mathbf{p} - \mathbf{n}$; the decision threshold can be found by noting that $(\mathbf{p} + \mathbf{n})/2$ is on the decision boundary, and hence $t = (\mathbf{p} - \mathbf{n}) \cdot (\mathbf{p} + \mathbf{n})/2 = (||\mathbf{p}||^2 - ||\mathbf{n}||^2)/2$, where $||\mathbf{x}||$ denotes the length of vector $\mathbf{x}$.

Because data is usually noisy, linear separability doesn't occur very often in practice, unless the data is very sparse, as in text classification. Recall that we used a large vocabulary, say 10 000 words, each word corresponding to a Boolean feature indicating whether or not that word occurs in the document. This means that the instance space has 10 000 dimensions, but for any one document no more than a small percentage of the features will be non-zero. As a result there is much 'empty space' between instances, which increases the possibility of linear separability. However, because linearly separable data doesn't uniquely define a decision boundary, we are now faced with a problem: which of the infinitely many decision boundaries should we choose? One natural option is to prefer large margin classifiers, where the *margin* of a linear classifier is the distance between the decision boundary and the closest instance. ☞ *Support vector machines*, discussed in Chapter 7, are a powerful kind of linear classifier that find a decision boundary whose margin is as large as possible (Figure 1.2).

Geometric concepts, in particular linear transformations, can be very helpful to understand the similarities and differences between machine learning methods (Background 1.2). For instance, we would expect most if not all learning algorithms to be translation-invariant, i.e., insensitive to where we put the origin of our coordinate system. Some algorithms may also be rotation-invariant, e.g., linear classifiers or support vector machines;but many others aren't, including

Bayesian classifiers. Similarly, some algorithms may be sensitive to non-uniform scaling.

Figure 1.2. The decision boundary learned by a support vector machine from the linearly separable data from Figure 1. The decision boundary maximises the margin, which is indicated by the dotted lines. The circled data points are the support vectors.

A very useful geometric concept in machine learning is the notion of *distance*. If the distance between two instances is small then the instances are similar in terms of their feature values, and so nearby instances would be expected to receive the same classification or belong to the same cluster. In a Cartesian coordinate system, distance can be measured by *Euclidean distance*, which is the square root of the sum of the squared distances along each coordinate:[2] $\sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}$. A very simple distance-based classifier works as follows: to classify a new instance, we retrieve from memory the most similar training instance (i.e., the training instance with smallest Euclidean distance from the instance to be classified), and simply assign that training instance's class. This classifier is known as the *nearest-neighbour classifier*. Endless variations on this simple yet powerful theme exist: we can retrieve the k most similar training instances and take a vote (k-nearest neighbour); we can weight each neighbour's vote inversely to its distance; we can apply the same idea to regression tasks by averaging the training instances' function value; and so on. What they all have in common is that predictions are local in the sense that they are based on only a few training instances, rather than being derived from a global model built from the entire data set.

Transformations in d-dimensional Cartesian coordinate systems can be conveniently represented by means of matrix notation. Let $\mathbf{x}$ be a d-vector representing a data point, then $\mathbf{x} + \mathbf{t}$ is the resulting point after *translating* over $\mathbf{t}$ (another d-vector). Translating a set of points over $\mathbf{t}$ can be equivalently understood as translating the origin over $-\mathbf{t}$. Using *homogeneous coordinates* – the addition of an extra dimension set to 1 – translations can be expressed by matrix multiplication: e.g., in two dimensions we have

$$\mathbf{x}^\circ = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix} \qquad \mathbf{T} = \begin{pmatrix} 1 & 0 & 0 \\ t_1 & 1 & 0 \\ t_2 & 0 & 1 \end{pmatrix} \qquad \mathbf{T}\mathbf{x}^\circ = \begin{pmatrix} 1 \\ x_1 + t_1 \\ x_2 + t_2 \end{pmatrix}$$

A *rotation* is defined by any d-by-d matrix $\mathbf{D}$ whose transpose is its inverse (which means it is orthogonal) and whose determinant is 1. In two dimensions a rotation matrix can be written as $\mathbf{R} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$, representing a clockwise rotation over angle θ about the origin. For instance, $\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ is a 90 degrees clockwise rotation.

A *scaling* is defined by a diagonal matrix; in two dimensions $\mathbf{S} = \begin{pmatrix} s_1 & 0 \\ 0 & s_2 \end{pmatrix}$. A *uniform scaling* applies the same scaling factor s in all dimensions and can be written as $s\mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Notice that a uniform scaling with scaling factor – 1 is a rotation (over 180 degrees in the two-dimensional case). A common scenario which utilises all these transformations is the following. Given an n-by-d matrix $\mathbf{X}$ representing n data points in d-dimensional space, we first calculate the centre of mass or mean vector μ by averaging each column. We then zero-centre the data set by subtracting $-\mu$ from each row, which corresponds to a translation. Next, we rotate the data such that as much variance (a measure of the data's 'spread' in a certain direction) as possible is aligned with our coordinate axes;this can be achieved by a matrix transformation known as ☞ *principal component analysis*, about which you will learn more in Chapter 10. Finally, we scale the data to unit variance along each coordinate.

Background 1.2. Linear transformations.

There is a nice relationship between Euclidean distance and the mean of a set of points: there is no other point which has smaller total squared Euclidean distance to the given points (see Theorem 8.1 on p.238 for a proof of this). Consequently, we can use the mean of a set of nearby points as a representative *exemplar* for those points. Suppose we want to cluster our data into K clusters, and we have an initial guess of how the data should be clustered. We then calculate the means of each initial cluster, and reassign each point to the nearest cluster mean. Unless our initial guess was a lucky one, this will have changed some of the clusters, so we repeat these two steps (calculating the cluster means and reassigning points to clusters) until no change occurs. This clustering algorithm, which is called ☞ *K-means* and is further discussed in Chapter 8, is very widely used to solve a range of clustering tasks. It remains to be decided how we construct our initial guess. This is usually done randomly: either by randomly partitioning the data set into K 'clusters' or by randomly guessing K 'cluster centres'. The fact that these initial 'clusters' or 'cluster centres' will bear little resemblance to the actual data is not a problem, as this will quickly be rectified by running the algorithm for a number of iterations.

To summarise, geometric notions such as planes, translations and rotations, and distance are very useful in machine learning as they allow us to understand many key concepts in intuitive ways. Geometric models exploit these intuitions and are simple, powerful and allow many variations with little effort. For instance, instead of using Euclidean distance, which can be sensitive to outliers, other distances can be used such as *Manhattan distance*, which sums the distances along each coordinate: $\sum_{i=1}^{d} |x_i - y_i|$.

Probabilistic models

The second type of models are probabilistic in nature, like the Bayesian classifier we considered earlier. Many of these models are based around the following idea. Let X denote the variables we know about, e.g., our instance's feature values; and let Y denote the *target variables* we're interested in, e.g., the instance's class. The key question in machine learning is how to model the relationship between X and Y. The statistician's approach is to assume that there is some underlying random process that generates the values for these variables,

according to a well-defined but unknown probability distribution. We want to use the data to find out more about this distribution. Before we look into that, let's consider how we could use that distribution once we have learned it.

Since X is known for a particular instance but Y may not be, we are particularly interested in the conditional probabilities $P(Y \mid X)$. For instance, Y could indicate whether the e-mail is spam, and X could indicate whether the e-mail contains the words 'Viagra' and 'lottery'. The probability of interest is then $P(Y \mid \text{Viagra, lottery})$, with Viagra and lottery two Boolean variables which together constitute the feature vector X. For a particular e-mail we know the feature values and so we might write $P(Y \mid \text{Viagra} = 1, \text{lottery} = 0)$ if the e-mail contains the word 'Viagra' but not the word 'lottery'. This is called a *posterior probability* because it is used *after* the features X are observed.

Viagra	lottery	$P(Y = \text{spam} \mid \text{Viagra, lottery})$	$P(Y = \text{ham} \mid \text{Viagra, lottery})$
0	0	0.31	**0.69**
0	1	**0.65**	0.35
1	0	**0.80**	0.20
1	1	0.40	**0.60**

Table 1.2. An example posterior distribution. 'Viagra' and 'lottery' are two Boolean features; Y is the class variable, with values 'spam' and 'ham'. In each row the most likely class is indicated in bold.

Table 1.2 shows an example of how these probabilities might be distributed. From this distribution you can conclude that, if an e-mail doesn't contain the word 'Viagra', then observing the word 'lottery' increases the probability of the e-mail being spam from 0.31 to 0.65;but if the e-mail does contain the word 'Viagra', then observing the word 'lottery' as well decreases the spam probability from 0.80 to 0.40. Even though this example table is small, it will grow unfeasibly large very quickly (with n Boolean variables 2^n cases have to be distinguished). We therefore don't normally have access to the full joint distribution and have to approximate it using additional assumptions, as we will see below.

Assuming that X and Y are the only variables we know and care about, the posterior distribution $P(Y \mid X)$ helps us to answer many questions of interest. For instance, to classify a new e-mail we determine whether the words 'Viagra' and 'lottery' occur in it, lookup the corresponding probability $P(Y = \text{spam} \mid \text{Viagra, lottery})$, and predict spam if this probability exceeds 0.5 and ham otherwise. Such a recipe to predict a value of Y on the basis of the values of X and the posterior distribution $P(Y \mid X)$ is called a *decision rule*. We can do this even

without knowing all the values of X, as the following example shows.

Example 1.2 (Missing values). Suppose we skimmed an e-mail and noticed that it contains the word 'lottery' but we haven't looked closely enough to determine whether it uses the word 'Viagra'. This means that we don't know whether to use the second or the fourth row in Table 1.2 to make a prediction. This is a problem, as we would predict spam if the e-mail contained the word 'Viagra' (second row) and ham if it didn't (fourth row).

The solution is to average these two rows, using the probability of 'Viagra' occurring in any e-mail (spam or not):

$$\begin{aligned} P(Y|\text{lottery}) = & P(Y|\text{Viagra}=0,\text{lottery})P(\text{Viagra}=0) \\ & + P(Y|\text{Viagra}=1,\text{lottery})P(\text{Viagra}=1) \end{aligned}$$

For instance, suppose for the sake of argument that one in ten e-mails contain the word 'Viagra', then $P(\text{Viagra} = 1) = 0.10$ and $P(\text{Viagra} = 0) = 0.90$. Using the above formula, we obtain $P(Y = \text{spam}|\text{lottery} = 1) = 0.65 \cdot 0.90 + 0.40 \cdot 0.10 = 0.625$ and $P(Y = \text{ham}|\text{lottery} = 1) = 0.35 \cdot 0.90 + 0.60 \cdot 0.10 = 0.375$. Because the occurrence of 'Viagra' in any e-mail is relatively rare, the resulting distribution deviates only a little from the second row in Table 1.2.

As a matter of fact, statisticians work very often with different conditional probabilities, given by the *likelihood function* $P(X|Y)$.[3] This seems counter-intuitive at first: why would we be interested in the probability of an event we know has occurred (X), conditioned on something we don't know anything about (Y)? I like to think of these as thought experiments: if somebody were to send me a spam e-mail, how likely would it be that it contains exactly the words of the e-mail I'm looking at? And how likely if it were a ham e-mail instead? 'Not very likely at all in either case', you might think, and you would be right: with so many words to choose from, the probability of any particular combination of words would be very small indeed. What really matters is not the magnitude of these likelihoods, but their ratio: how much more likely is it to observe this combination of words in a spam e-mail than it is in a non-spam e-mail. For

instance, suppose that for a particular e-mail described by X we have $P(X \mid Y = \text{spam}) = 3.5 \cdot 10^{-5}$ and $P(X \mid Y = \text{ham}) = 7.4 \cdot 10^{-6}$, then observing X in a spam e-mail is nearly five times more likely than it is in a ham e-mail. This suggests the following decision rule: predict spam if the likelihood ratio is larger than 1 and ham otherwise.

So which one should we use: posterior probabilities or likelihoods? As it turns out, we can easily transform one into the other using *Bayes'rule*, a simple property of conditional probabilities which states that

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Here, $P(Y)$ is the *prior probability*, which in the case of classification tells me how likely each of the classes is *a priori*, i.e., before I have observed the data X. $P(X)$ is the probability of the data, which is independent of Y and in most cases can be ignored (or inferred in a normalisation step, as it is equal to $\sum_Y P(X \mid Y) P(Y)$). The first decision rule above suggested that we predict the class with maximum posterior probability, which using Bayes' rule can be written in terms of the likelihood function:

$$y_{\text{MAP}} = \arg\max_Y P(Y|X) = \arg\max_Y \frac{P(X|Y)P(Y)}{P(X)} = \arg\max_Y P(X|Y)P(Y)$$

This is usually called the *maximum a posteriori* (*MAP*) decision rule. Now, if we assume a uniform prior distribution (i.e., $P(Y)$ the same for every value of Y) this reduces to the *maximum likelihood* (*ML*) decision rule:

$$y_{\text{ML}} = \arg\max_Y P(X|Y)$$

A useful rule of thumb is: *use likelihoods if you want to ignore the prior distribution or assume it uniform, and posterior probabilities otherwise.*

If we have only two classes it is convenient to work with ratios of posterior probabilities or likelihood ratios. If we want to know how much the data favours one of two classes, we can calculate the *posterior odds*: e.g.,

$$\frac{P(Y = \text{spam}|X)}{P(Y = \text{ham}|X)} = \frac{P(X|Y = \text{spam})}{P(X|Y = \text{ham})} \frac{P(Y = \text{spam})}{P(Y = \text{ham})}$$

In words: the *posterior odds* are the product of the *likelihood ratio* and the *prior odds*. If the odds are larger than 1 we conclude that the class in the enumerator is

the more likely of the two; if it is smaller than 1 we take the class in the denominator instead. In very many cases the prior odds is a simple constant factor that can be manually set, estimated from the data, or optimised to maximise performance on a test set.

Example 1.3 (Posterior odds). Using the data from Table 1.2, and assuming a uniform prior distribution, we arrive at the following posterior odds:

$$\frac{P(Y = \text{spam}|\text{Viagra} = 0, \text{lottery} = 0)}{P(Y = \text{ham}|\text{Viagra} = 0, \text{lottery} = 0)} = \frac{0.31}{0.69} = 0.45$$
$$\frac{P(Y = \text{spam}|\text{Viagra} = 1, \text{lottery} = 1)}{P(Y = \text{ham}|\text{Viagra} = 1, \text{lottery} = 1)} = \frac{0.40}{0.60} = 0.67$$
$$\frac{P(Y = \text{spam}|\text{Viagra} = 0, \text{lottery} = 1)}{P(Y = \text{ham}|\text{Viagra} = 0, \text{lottery} = 1)} = \frac{0.65}{0.35} = 1.9$$
$$\frac{P(Y = \text{spam}|\text{Viagra} = 1, \text{lottery} = 0)}{P(Y = \text{ham}|\text{Viagra} = 1, \text{lottery} = 0)} = \frac{0.80}{0.20} = 4.0$$

Using a MAP decision rule (which in this case is the same as the ML decision rule, since we assumed a uniform prior) we predict ham in the top two cases and spam in the bottom two. Given that the full posterior distribution is all there is to know about the domain in a statistical sense, these predictions are the best we can do: they are *Bayes-optimal*.

Y	$P(\text{Viagra} = 1\|Y)$	$P(\text{Viagra} = 0\|Y)$
spam	0.40	0.60
ham	0.12	0.88

Y	$P(\text{lottery} = 1\|Y)$	$P(\text{lottery} = 0\|Y)$
spam	0.21	0.79
ham	0.13	0.87

Table 1.3. Example marginal likelihoods.

It is clear from the above analysis that the likelihood function plays an important role in statistical machine learning. It establishes what is called a *generative model*: a probabilistic model from which we can sample values of all variables involved. Imagine a box with two buttons labelled 'ham' and 'spam'. Pressing the 'ham' button generates a random e-mail according to $P(X \mid Y = \text{ham})$; pressing the 'spam' button generates a random e-mail accordingto $P(X \mid Y = \text{spam})$. The question now is what we put inside the box. Let's try a model that is so simplistic it's almost laughable. Assuming a vocabulary of 10 000 words, you have two bags with 10 000 coins each, one for each word in the vocabulary. In order to generate a random e-mail, you take the appropriate bag depending on which button was pressed, and toss each of the 10 000 coins in that bag to decide which words should go in the e-mail (say heads is in and tails is out).

In statistical terms, each coin – which isn't necessarily fair – represents a parameter of the model, so we have 20 000 parameters. If 'Viagra' is a word in the vocabulary, then the coin labelled 'Viagra' in the bag labelled 'spam' represents $P(\text{Viagra}|Y = \text{spam})$ and the coin labelled 'Viagra' in the bag labelled 'ham' represents $P(\text{Viagra}|Y = \text{ham})$. Together, these two coins represent the left table in Table 1.3. Notice that by using different coins for each word we have tacitly assumed that likelihoods of individual words are independent within the same class, which – if true – allows us to decompose the joint likelihood into a product of *marginal likelihoods*:

$$P(\text{Viagra}, \text{lottery}|Y) = P(\text{Viagra}|Y)P(\text{lottery}|Y)$$

Effectively, this independence assumption means that knowing whether one word occurs in the e-mail doesn't tell you anything about the likelihood of other words. The probabilities on the right are called marginal likelihoods because they are obtained by 'marginalising' some of the variables in the joint distribution: e.g., $P(\text{Viagra}|Y) = \Sigma_{\text{lottery}} P(\text{Viagra, lottery}|Y)$.

Example 1.4 (Using marginal likelihoods). Assuming these estimates come out as in Table 1.3, we can then calculate likelihood ratios (the previously calculated odds from the full posterior distribution are shown in brackets):

$$\frac{P(\text{Viagra}=0|Y=\text{spam})}{P(\text{Viagra}=0|Y=\text{ham})}\frac{P(\text{lottery}=0|Y=\text{spam})}{P(\text{lottery}=0|Y=\text{ham})}=\frac{0.60}{0.88}\frac{0.79}{0.87}=0.62 \qquad (0.45)$$

$$\frac{P(\text{Viagra}=0|Y=\text{spam})}{P(\text{Viagra}=0|Y=\text{ham})}\frac{P(\text{lottery}=1|Y=\text{spam})}{P(\text{lottery}=1|Y=\text{ham})}=\frac{0.60}{0.88}\frac{0.21}{0.13}=1.1 \qquad (1.9)$$

$$\frac{P(\text{Viagra}=1|Y=\text{spam})}{P(\text{Viagra}=1|Y=\text{ham})}\frac{P(\text{lottery}=0|Y=\text{spam})}{P(\text{lottery}=0|Y=\text{ham})}=\frac{0.40}{0.12}\frac{0.79}{0.87}=3.0 \qquad (4.0)$$

$$\frac{P(\text{Viagra}=1|Y=\text{spam})}{P(\text{Viagra}=1|Y=\text{ham})}\frac{P(\text{lottery}=1|Y=\text{spam})}{P(\text{lottery}=1|Y=\text{ham})}=\frac{0.40}{0.12}\frac{0.21}{0.13}=5.4 \qquad (0.67)$$

We see that, using a maximum likelihood decision rule, our very simple model arrives at the Bayes-optimal prediction in the first three cases, but not in the fourth ('Viagra' and 'lottery' both present), where the marginal likelihoods are actually very misleading. A possible explanation is that these terms are very unlikely to occur together in any e-mail, but slightly more likely in ham than spam – for instance, I might be making exactly this point in an e-mail!

One might call the independence assumption that allows us to decompose joint likelihoods into a product of marginal likelihoods 'naive' – which is exactly what machine learners do when they refer to this simplified Bayesian classifier as *naive Bayes*. This shouldn't be taken as a derogatory term – on the contrary, it illustrates a very important guideline in machine learning: *everything should be made as simple as possible, but not simpler*.[4] In our statistical context, this rule boils down to using the simplest generative model that solves our task. For instance, we may decide to stick to naive Bayes on the grounds that the cases in which the marginal probabilities are misleading are very unlikely to occur in reality and therefore will be difficult to learn from data.

We now have some idea what a probabilistic model looks like, but how do we learn such a model? In many cases this will be a matter of estimating the model parameters from data, which is usually achieved by straightforward counting. For example, in the coin toss model of spam recognition we had two coins for every word w_i in our vocabulary, one of which is to be tossed if we are generating a spam e-mail and the other for ham e-mails. Let's say that the spam coin comes up heads with probability $\theta_i^{\oplus}$ and the ham coin with probability $\theta_i^{\ominus}$, then these parameters characterise all the likelihoods:

$$P(w_i=1|Y=\text{spam})=\theta_i^{\oplus} \qquad P(w_i=0|Y=\text{spam})=1-\theta_i^{\oplus}$$

$$P(w_i=1|Y=\text{ham})=\theta_i^{\ominus} \qquad P(w_i=0|Y=\text{ham})=1-\theta_i^{\ominus}$$

Figure 1.3. (top) Visualisation of two marginal likelihoods as estimated from a small data set. The colours indicate whether the likelihood points to spam or ham. **(bottom)** Combining the two marginal likelihoods gives a pattern not unlike that of a Scottish tartan. The colour of a particular cell is a result of the colours in the corresponding row and column.

In order to estimate the parameters $\theta_i^{\pm}$ we need a training set of e-mails labelled spam or ham. We take the spam e-mails and count how many of them w_i occurs in: dividing by the total number of spam e-mails gives us an estimate of $\theta_i^{\oplus}$. Repeating this for the ham e-mails results in an estimate of $\theta_i^{\ominus}$. And that's all there is to it![5]

Figure 1.4. (left) A feature tree combining two Boolean features. Each internal node or split is labelled with a feature, and each edge emanating from a split is labelled with a feature value. Each leaf therefore corresponds to a unique combination of feature values. Also indicated in each leaf is the class distribution derived from the training set. **(right)** A feature tree partitions the instance space into rectangular regions, one for each leaf. We can clearly see that the majority of ham lives in the lower left-hand corner.

Figure 1.3 visualises this for a variant of the naive Bayes classifier discussed above. In this variant, we record the number of times a particular word occurs in an e-mail, rather than just whether it occurs or not. We thus need a parameter $p_{ij\pm}$ for each likelihood $P(w_i = j|Y = \pm)$, where $j = 0, 1, 2, \ldots$. For example, we see that there are two spam e-mails in which 'lottery' occurs twice, and one ham e-mail in which 'Peter' occurs five times. Combining the two sets of marginal likelihoods, we get the tartan-like pattern of Figure 1.3 (bottom), which is why I like to call naive Bayes the 'Scottish classifier'. This is a visual reminder of the fact that a multivariate naive Bayes model *decomposes* into a bunch of univariate ones. We will return to this issue of decomposition several times in the book.

Logical models

The third type of model we distinguish is more algorithmic in nature, drawing inspiration from computer science and engineering. I call this type 'logical' because models of this type can be easily translated into rules that are understandable by humans, such as ·**if** Viagra = 1 **then** Class = Y = spam·. Such rules are easily organised in a tree structure, such as the one in Figure 1.4, which I will call a *feature tree*. The idea of such a tree is that features are used to iteratively partition the instance space. The leaves of the tree therefore correspond to rectangular areas in the instance space (or hyper-rectangles, more generally) which we will call *instance space segments*, or segments for short.

Depending on the task we are solving, we can then label the leaves with a class, a probability, a real value, and so on. Feature trees whose leaves are labelled with classes are commonly called *decision trees*.

Figure 1.5. (left) A complete feature tree built from two Boolean features. **(right)** The corresponding instance space partition is the finest partition that can be achieved with those two features.

Example 1.5 (Labelling a feature tree). The leaves of the tree in Figure 1.4 could be labelled, from left to right, as ham – spam – spam, employing a simple decision rule called *majority class*. Alternatively, we could label them with the proportion of spam e-mail occurring in each leaf: from left to right, 1/3, 2/3, and 4/5. Or, if our task was a regression task, we could label the leaves with predicted real values or even linear functions of some other, real-valued features.

Feature trees are very versatile and will play a major role in this book. Even models that do not appear tree-based at first sight can be understood as being built on a feature tree. Consider, for instance, the naive Bayes classifier discussed previously. Since it employs marginal likelihoods such as the ones in Table 1.3 on p.29, it partitions the instance space in as many regions as there are combinations of feature values. This means that it can be thought of as employing a *complete* feature tree, which contains all features, one at each level of the tree (Figure 1.5). Incidentally, notice that the right-most leaf is the one where naive Bayes made a wrong prediction. Since this leaf covers only a single example, there is a danger that this tree is overfitting the data and that the previous tree is a better model. Decision tree learners often employ *pruning*

techniques which delete splits such as these.

A *feature list* is a binary feature tree which always branches in the same direction, either left or right. The tree in Figure 1.4 is a left-branching feature list. Such feature lists can be written as nested if-then-else statements that will be familiar to anyone with a bit of programming experience. For instance, if we were to label the leaves in Figure 1.4 by majority class we obtain the following *decision list*:

```
·if Viagra = 1 then Class = Y = spam·
·else if lottery = 1 then Class = Y = spam·
·else Class = Y = ham·
```

Logical models often have different, equivalent formulations. For instance, two alternative formulations for this model are

```
·if Viagra = 1 ∨ lottery = 1 then Class = Y = spam·
·else Class = Y = ham·

·if Viagra = 0 ∧ lottery = 0 then Class = Y = ham·
·else Class = Y = spam·
```

The first of these alternative formulations combines the two rules in the original decision list by means of *disjunction* ('or'), denoted by $\vee$. This selects a single non-rectangular area in instance space. The second model formulates a *conjunctive* condition ('and', denoted by $\wedge$) for the opposite class (ham) and declares everything else as spam.

We can also represent the same model as un-nested rules:

```
·if Viagra = 1 then Class = Y = spam·
·if Viagra = 0 ∧ lottery = 1 then Class = Y = spam·
·if Viagra = 0 ∧ lottery = 0 then Class = Y = ham·
```

Here, every path from root to a leaf is translated into a rule. As a result, although rules from the same sub-tree share conditions (such as Viagra = 0), every pair of rules will have at least some mutually exclusive conditions (such as lottery = 1 in the second rule and lottery = 0 in the third). However, this is not always the case: rules can have a certain overlap.

Example 1.6 (Overlapping rules). Consider the following rules:

·**if** lottery = 1 **then** Class = Y = spam·
·**if** Peter = 1 **then** Class = Y = ham·

As can be seen in Figure 1.6, these rules overlap for lottery = 1 ∧ Peter = 1, for which they make contradictory predictions. Furthermore, they fail to make any predictions for lottery = 0 ∧ Peter = 0.

Figure 1.6. The effect of overlapping rules in instance space. The two rules make contradictory predictions in the top right-hand corner, and no prediction at all in the bottom left-hand corner.

A logician would say that rules such as these are both *inconsistent* and *incomplete*. To address incompleteness, we could add a *default rule* to predict, e.g., the majority class for instances not covered by any rule. There are a number of options to deal with overlapping rules, which will be further considered in Chapter 6.

Tree-learning algorithms typically work in a top–down fashion. The first task is to find a good feature to split on at the top of the tree. The aim here is to find splits that result in improved purity of the nodes on the next level, where the purity of a node refers to the degree in which the training examples belonging to that node are of the same class. Once the algorithm has found such a feature, the training set is partitioned into subsets, one for each node resulting from the split. For each of these subsets, we again find a good feature to split on, and so on. An algorithm that works by repeatedly splitting a problem into small sub-problems is what computer scientists call a divide-and-conquer algorithm. We stop splitting a node when all training examples belonging to that node are of the

same class. Most rule learning algorithms also work in a top–down fashion. We learn a single rule by repeatedly adding conditions to the rule until the rule only covers examples of a single class. We then remove the covered examples of that class, and repeat the process. This is sometimes called a separate-and-conquer approach.

An interesting aspect of logical models, which sets them aside from most geometric and probabilistic models, is that they can, to some extent, provide *explanations* for their predictions. For example, a prediction assigned by a decision tree could be explained by reading off the conditions that led to the prediction from root to leaf. The model itself can also easily be inspected by humans, which is why they are sometimes called *declarative*. Declarative models do not need to be restricted to the simple rules that we have considered so far. The logical rule learning system Progol found the following set of conditions to predict whether a molecular compound is carcinogenic (causes cancer):

1. it tests positive in the Salmonella assay; or
2. it tests positive for sex-linked recessive lethal mutation in Drosophila;or
3. it tests negative for chromosome aberration; or
4. it has a carbon in a six-membered aromatic ring with a partial charge of –0.13; or
5. it has a primary amine group and no secondary or tertiary amines;or
6. it has an aromatic (or resonant) hydrogen with partial charge ≥ 0.168; or
7. it has a hydroxy oxygen with a partial charge ≥ -0.616 and an aromatic (or resonant) hydrogen; or
8. it has a bromine; or
9. it has a tetrahedral carbon with a partial charge ≤ -0.144 and tests positive on Progol's mutagenicity rules.[6]

The first three conditions concerned certain tests that were carried out for all molecules and whose results were recorded in the data as Boolean features. In contrast, the remaining six rules all refer to the structure of the molecule and were constructed entirely by Progol. For instance, rule 4 predicts that a molecule is carcinogenic if it contains a carbon atom with certain properties. This condition is different from the first three in that it is not a pre-recorded feature in the data, but a new feature that is constructed by Progol during the learning process because it helps to explain the data.

Grouping and grading

We have looked at three general types of models: geometric models, probabilistic models and logical models. As I indicated, although there are some underlying principles pertaining to each of these groups of models, the main reason for dividing things up along this dimension is one of convenience. Before I move on to the third main ingredient of machine learning, features, I want to briefly introduce another important but somewhat more abstract dimension that is in some sense orthogonal to the geometric–probabilistic–logical dimension. This is the distinction between *grouping models* and *grading models*. The key difference between these models is the way they handle the instance space.

Grouping models do this by breaking up the instance space into groups or *segments*, the number of which is determined at training time. One could say that grouping models have a fixed and finite 'resolution' and cannot distinguish between individual instances beyond this resolution. What grouping models do at this finest resolution is often something very simple, such as assigning the majority class to all instances that fall into the segment. The main emphasis of training a grouping model is then on determining the right segments so that we can get away with this very simple labelling at the local segment level. Grading models, on the other hand, do not employ such a notion of segment. Rather than applying very simple, local models, they form one global model over the instance space. Consequently, grading models are (usually) able to distinguish between arbitrary instances, no matter how similar they are. Their resolution is, in theory, infinite, particularly when working in a Cartesian instance space.

Figure 1.7. A 'map' of some of the models that will be considered in this book. Models that share

characteristics are plotted closer together: logical models to the right, geometric models on the top left and probabilistic models on the bottom left. The horizontal dimension roughly ranges from grading models on the left to grouping models on the right.

A good example of grouping models are the tree-based models we have just considered. They work by repeatedly splitting the instance space into smaller subsets. Because trees are usually of limited depth and don't contain all the available features, the subsets at the leaves of the tree partition the instance space with some finite resolution. Instances filtered into the same leaf of the tree are treated the same, regardless of any features not in the tree that might be able to distinguish them. Support vector machines and other geometric classifiers are examples of grading models. Because they work in a Cartesian instance space, they are able to represent and exploit the minutest differences between instances. As a consequence, it is always possible to come up with a new test instance that receives a score that has not been given to any previous test instance.

The distinction between grouping and grading models is relative rather than absolute, and some models combine both features. For instance, even though linear classifiers are a prime example of a grading model, it is easy to think of instances that a linear model can't distinguish, namely instances on a line or plane parallel to the decision boundary. The point is not so much that there aren't any segments, but that there are infinitely many. On the other end of the spectrum, regression trees combine grouping and grading features, as we shall see a little later. The overall picture is thus somewhat like what is depicted in Figure 1.7. A taxonomy of eight different models discussed in the book is given in Figure 1.8.[7] These models will be discussed in detail in Chapters 4–9.

Figure 1.8. A taxonomy describing machine learning methods in terms of the extent to which they are grading or grouping models, logical, geometric or a combination, and supervised or unsupervised. The colours indicate the type of model, from left to right: logical (red), probabilistic (orange) and geometric (purple).

1.3 Features: the workhorses of machine learning

Now that we have seen some more examples of machine learning tasks and models, we turn to the third and final main ingredient. Features determine much of the success of a machine learning application, because a model is only as good as its features. A feature can be thought of as a kind of measurement that can be easily performed on any instance. Mathematically, they are functions that map from the instance space to some set of feature values called the *domain* of the feature. Since measurements are often numerical, the most common feature domain is the set of real numbers. Other typical feature domains include the set of integers, for instance when the feature counts something, such as the number of occurrences of a particular word; the Booleans, if our feature is a statement that can be true or false for a particular instance, such as 'this e-mail is addressed to Peter Flach'; and arbitrary finite sets, such as a set of colours, or a set of shapes.

Model	geom	stats	logic	group	grad	disc	real	sup	unsup	multi
Trees	1	0	3	3	0	3	2	3	2	3
Rules	0	0	3	3	1	3	2	3	0	2
naive Bayes	1	3	1	3	1	3	1	3	0	3
kNN	3	1	0	2	2	1	3	3	0	3
Linear Classifier	3	0	0	0	3	1	3	3	0	0
Linear Regression	3	1	0	0	3	0	3	3	0	1
Logistic Regression	3	2	0	0	3	1	3	3	0	0
SVM	2	2	0	0	3	2	3	3	0	0
Kmeans	3	2	0	1	2	1	3	0	3	1
GMM	1	3	0	0	3	1	3	0	3	1
Associations	0	0	3	3	0	3	1	0	3	1

Table 1.4. The MLM data set describing properties of machine learning models. Both Figure 1.7 and Figure 1.8 were generated from this data.

Example 1.7 (The MLM data set). Suppose we have a number of learning models that we want to describe in terms of a number of properties:

- the extent to which the models are geometric, probabilistic or logical;
- whether they are grouping or grading models;
- the extent to which they can handle discrete and/or real-valued features;
- whether they are used in supervised or unsupervised learning; and
- the extent to which they can handle multi-class problems.

The first two properties could be expressed by discrete features with three and two values, respectively; or if the distinctions are more gradual, each aspect could be rated on some numerical scale. A simple approach would be to measure each property on an integer scale from 0 to 3, as in Table 1.4. This table establishes a data set in which each row represents an instance and each column a feature. For example, according to this (highly simplified) data some models are purely grouping models (Trees, Associations) or purely grading models (the Linear models, Logistic Regression and GMM), whereas others are more mixed. We can also see that Trees and Rules have very similar values for most of the features, whereas GMM and Associations have mostly different values.

This small data set will be used in several examples throughout the book. In fact, the taxonomy in Figure 1.8 was adapted by hand from a decision

tree learned from this small data set, using the models as classes. And the plot in Figure 1.7 was constructed using a dimensionality reduction technique which preserves pairwise distances as much as possible.

Two uses of features

It is worth noting that features and models are intimately connected, not just because models are defined in terms of features, but because a single feature can be turned into what is sometimes called a *univariate model*. We can therefore distinguish two uses of features that echo the distinction between grouping and grading models. A very common use of features, particularly in logical models, is to zoom in on a particular area of the instance space. Let f be a feature counting the number of occurrences of the word 'Viagra' in an e-mail, and let x stand for an arbitrary e-mail, then the condition $f(x) = 0$ selects e-mails that don't contain the word 'Viagra', $f(x) \neq 0$ or $f(x) > 0$ selects e-mails that do, $f(x) \geq 2$ selects e-mails that contain the word at least twice, and so on. Such conditions are called *binary splits*, because they divide the instance space into two groups: those that satisfy the condition, and those that don't. Non-binary splits are also possible: for instance, if g is a feature that has the value 'tweet' for e-mails with up to 20 words, 'short' for e-mails with 21 to 50 words, 'medium' for e-mails with 51 to 200 words, and 'long' for e-mails with more than 200 words, then the expression $g(x)$ represents a four-way split of the instance space. As we have already seen, such splits can be combined in a feature tree, from which a model can be built.

A second use of features arises particularly in supervised learning. Recall that a linear classifier employs a decision rule of the form $\sum_{i=1}^{n} w_i x_i > t$, where x_i is a numerical feature.[8] The linearity of this decision rule means that each feature makes an independent contribution to the score of an instance. This contribution depends on the weight w_i: if this is large and positive, a positive x_i increases the score; if $w_i \ll 0$, a positive x_i decreases the score; if $w_i \approx 0$, x_i's influence is negligible. Thus, the feature makes a precise and measurable contribution to the final prediction. Also note that that individual features are not 'thresholded', but their full 'resolution' is used in computing an instance's score. These two uses of features – 'features as splits' and 'features as predictors' – are sometimes combined in a single model.

Figure 1.9. (left) A regression tree combining a one-split feature tree with linear regression models in the leaves. Notice how x is used as both a splitting feature and a regression variable. **(right)** The function $y = \cos \pi x$ on the interval $-1 \leq x \leq 1$, and the piecewise linear approximation achieved by the regression tree.

Example 1.8 (Two uses of features). Suppose we want to approximate $y = \cos \pi x$ on the interval $-1 \leq x \leq 1$. A linear approximation is not much use here, since the best fit would be $y = 0$. However, if we split the x-axis in two intervals $-1 \leq x < 0$ and $0 \leq x \leq 1$, we could find reasonable linear approximations on each interval. We can achieve this by using x both as a splitting feature and as a regression variable (Figure 1.9).

Feature construction and transformation

There is a lot of scope in machine learning for playing around with features. In the spam filter example, and text classification more generally, the messages or documents don't come with built-in features; rather, they need to be constructed by the developer of the machine learning application. This *feature construction* process is absolutely crucial for the success of a machine learning application. Indexing an e-mail by the words that occur in it (called a *bag of Words* representation as it disregards the order of the words in the e-mail) is a carefully engineered representation that manages to amplify the 'signal' and attenuate the 'noise' in spam e-mail filtering and related classification tasks. However, it is easy to conceive of problems where this would be exactly the wrong thing to do: for instance if we aim to train a classifier to distinguish between grammatical and ungrammatical sentences, word order is clearly signal rather than noise, and a different representation is called for.

Figure 1.10. (left) Artificial data depicting a histogram of body weight measurements of people with (blue) and without (red) diabetes, with eleven fixed intervals of 10 kilograms width each. **(right)** By joining the first and second, third and fourth, fifth and sixth, and the eighth, ninth and tenth intervals, we obtain a discretisation such that the proportion of diabetes cases increases from left to right. This discretisation makes the featuremore useful in predicting diabetes.

It is often natural to build a model in terms of the given features. However, we are free to change the features as we see fit, or even to introduce new features. For instance, real-valued features often contain unnecessary detail that can be removed by *discretisation*. Imagine you want to analyse the body weight of a relatively small group of, say, 100 people, by drawing a histogram. If you measure everybody's weight in kilograms with one position after the decimal point (i.e., your precision is 100 grams), then your histogram will be sparse and spiky. It is hard to draw any general conclusions from such a histogram. It would be much more useful to discretise the body weight measurements into intervals of 10 kilograms. If we are in a classification context, say we're trying to relate body weight to diabetes, we could then associate each bar of the histogram with the proportion of people having diabetes among the people whose weight falls in that interval. In fact, as we shall see in Chapter 10, we can even choose the intervals such that this proportion is monotonically increasing (Figure 1.10).

The previous example gives another illustration of how, for a particular task such as classification, we can improve the signal-to-noise ratio of a feature. In more extreme cases of feature construction we transform the entire instance space. Consider Figure 1.11: the data on the left is clearly not linearly separable, but by mapping the instance space into a new 'feature space' consisting of the squares of the original features we see that the data becomes almost linearly separable. In fact, by adding in a third feature we can perform a remarkable trick: we can build this feature space classifier without actually constructing the feature space.

Figure 1.11. (left) A linear classifier would perform poorly on this data. **(right)** By transforming the original (x, y) data into $(x', y') = (x^2, y^2)$, the data becomes more 'linear', and alinear decision boundary $x' + y' = 3$ separates the data fairly well. In the original space this corresponds to a circle with radius $\sqrt{3}$ around the origin.

Example 1.9 (The kernel trick). Let $\mathbf{x}_1 = (x_1, y_1)$ and $\mathbf{x}_2 = (x_2, y_2)$ be two data points, and consider the mapping $(x, y) \mapsto (x^2, y^2, \sqrt{2}xy)$ to a three-dimensional feature space. The points in feature space corresponding to $\mathbf{x}_1$ and $\mathbf{x}_2$ are $\mathbf{x'}_1 = (x_1^2, y_1^2, \sqrt{2}x_1 y_1)$ and $\mathbf{x}_2' = (x_2^2, y_2^2, \sqrt{2}x_2 y_2)$. The dot product of these two feature vectors is

$$\mathbf{x}_1' \cdot \mathbf{x}_2' = x_1^2 x_2^2 + y_1^2 y_2^2 + 2x_1 y_1 x_2 y_2 = (x_1 x_2 + y_1 y_2)^2 = (\mathbf{x}_1 \cdot \mathbf{x}_2)^2$$

That is, by squaring the dot product in the original space we obtain the dot product in the new space *without actually constructing the feature vectors*! A function that calculates the dot product in feature space directly from the vectors in the original space is called a *kernel* – here the kernel is $\kappa(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2)^2$.

We can apply this *kernel trick* to the basic linear classifier if we modify the way the decision boundary is calculated. Recall that the basic linear classifier learns a decision boundary $\mathbf{w} \cdot \mathbf{x} = t$ with $\mathbf{w} = \mathbf{p} - \mathbf{n}$ being the difference between the mean of the positive examples and the mean of the negative examples. As an example, suppose we have $\mathbf{n} = (0, 0)$ and $\mathbf{p} = (0, 1)$, and let's assume for the sake of argument that the positive mean has been obtained from two training examples $\mathbf{p}_1 = (-1, 1)$ and $\mathbf{p}_2 = (1, 1)$. This

means that $\mathbf{p} = \frac{1}{2}(\mathbf{p}_1 + \mathbf{p}_2)$ and we can rewrite the decision boundary as $\frac{1}{2}\mathbf{p}_1 \cdot \mathbf{x} + \frac{1}{2}\mathbf{p}_2 \cdot \mathbf{x} - \mathbf{n} \cdot \mathbf{x} = t$. Applying the kernel trick we obtain the following decision boundary: $\frac{1}{2}\kappa(\mathbf{p}_1, \mathbf{x}) + \frac{1}{2}\kappa(\mathbf{p}_2, \mathbf{x}) - \kappa(\mathbf{n}, \mathbf{x}) = t$. Using the kernel defined earlier we have $\kappa(\mathbf{p}_1, \mathbf{x}) = (-x + y)^2$, $\kappa(\mathbf{p}_2, \mathbf{x}) = (x + y)^2$ and $\kappa(\mathbf{n}, \mathbf{x}) = 0$, from which we derive the decision boundary $\frac{1}{2}(-x + y)^2 + \frac{1}{2}(x + y)^2 = x^2 + y^2 = t$, i.e., a circle around the origin with radius $\sqrt{t}$. Figure 1.11 illustrates this further for a larger data set.

The key point in this 'kernelisation' of the basic linear classifier is that we don't summarise the training data by the positive and negative means – rather, we keep the training data (here: $\mathbf{p_1}$, $\mathbf{p_2}$ and $\mathbf{n}$), so that when classifying a new instance we can evaluate the kernel on it paired with each training example. In return for this more elaborate calculation we get the ability to construct much more flexible decision boundaries.

Interaction between features

One fascinating and multi-faceted aspect of features is that they may interact in various ways. Sometimes such interaction can be exploited, sometimes it can be ignored, and sometimes it poses a challenge. We have already seen an example of feature interaction when we talked about Bayesian spam filtering. Clearly, if we notice the term 'Viagra' in an e-mail, we are not really surprised to find that the e-mail also contains the phrase 'blue pill'. Ignoring this interaction, as the naive Bayes classifier does, means that we are overestimating the amount of information conveyed by observing both phrases in the same e-mail. Whether we can get away with this depends on our task: in spam email classification it turns out not to be a big problem, apart from the fact that we may need to adapt the decision threshold to account for this effect.

We can observe other examples of feature interaction in Table 1.4 on p.39. Consider the features 'grad' and 'real', which assess the extent to which models are of the grading kind, and the extent to which they can handle real-valued features. You may observe that the values of these two features differ by at most 1 for all but one model. Statisticians say that these features are positively correlated (see Background 1.3). Another pair of positively correlated features is 'logic' and 'disc', indicating logical models and the ability to handle discrete

features. We can also see some negatively correlated features, where the value of one goes up when the other goes down: this holds naturally for 'split' and 'grad', indicating whether models are primarily grouping or grading models;and also for 'logic' and 'grad'. Finally, pairs of uncorrelated features are 'unsup' and 'multi', standing for unsupervised models and the ability to handle more than two classes;and 'disc' and 'sup', the latter of which indicates supervised models.

In classification, features may be differently correlated depending on the class. For instance, it is conceivable that for somebody whose last name is Hilton and who works for the Paris city council, e-mails with just the word 'Paris' or just the word 'Hilton' are indicative of ham, whereas e-mails with both terms are indicative of spam. Put differently, within the spam class these features are positively correlated, while within the ham class they are negatively correlated. In such a case, ignoring these interactions will be detrimental for classification performance. In other cases, feature correlations may obscure the true model – we shall see examples of this later in the book. On the other hand, feature correlation sometimes helps us to zoom in on the relevant part of the instance space.

Random variables describe possible outcomes of a random process. They can be either discrete (e.g., the possible outcomes of rolling a die are {1, 2, 3, 4, 5, 6}) or continuous (e.g., the possible outcomes of measuring somebody's weight in kilograms). Random variables do not need to range over integer or real numbers, but it does make the mathematics quite a bit simpler so that is what we assume here.

If X is a discrete random variable with probability distribution $P(X)$ then the *expected value* of X is $\mathbb{E}[X] = \sum_x xP(x)$. For instance, the expected value of tossing a fair die is $1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + \ldots + 6 \cdot \frac{1}{6} = 3.5$. Notice that this is not actually a possible outcome. For a continuous random variable we need to replace the sum with an integral, and the probability distribution with a probability density function: $\mathbb{E}[X] = \int_{-\infty}^{+\infty} xp(x)\,dx$. The idea of this rather abstract concept is that if we take a sample $x_1, \ldots, x_n$ of outcomes of the random process, the expected value is what we expect the *sample mean* $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ to be – this is the celebrated *law of large numbers* first proved by Jacob Bernoulli in 1713. For this reason the expected value is often called the *population mean*, but it is important to realise that the latter is a theoretical value, while the sample mean is an empirical *estimate* of that theoretical value.

The expectation operator can be applied to functions of random variables. For instance, the (population) *variance* of a discrete random variable is defined as $\mathbb{E}\left[(X-\mathbb{E}[X])^2\right] = \sum_x (x-\mathbb{E}[X])^2 P(x)$ – this measures the spread of the distribution around the expected value. Notice that

$$\mathbb{E}\left[(X-\mathbb{E}[X])^2\right] = \sum_x (x-\mathbb{E}[X])^2 P(x) = \mathbb{E}\left[X^2\right] - \mathbb{E}[X]^2$$

We can similarly define the *sample variance* as $\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^2$, which decomposes as $\frac{1}{n}\sum_{i=1}^{n} x_i^2 - \overline{x}^2$. You will sometimes see the sample variance defined as $\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\overline{x})^2$: dividing by $n - 1$ rather than n results in a slightly larger estimate, which compensates for the fact that we are calculating the spread around the sample mean rather than the population mean.

The (population) *covariance* between two discrete random variables X and Y is defined as $\mathbb{E}[(X-\mathbb{E}[X])(Y-\mathbb{E}[Y])] = \mathbb{E}[X \cdot Y] - \mathbb{E}[X] \cdot \mathbb{E}[Y]$ The variance of X is a special case of this, with $Y = X$. Unlike the variance, the covariance can be positive as well as negative. Positive covariance means that both variables tend to increase or decrease together; negative covariance means that if one variable increases, the other tends to decrease. If we have a sample of pairs of values of X and Y, *sample covariance* is defined as $\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})(y_i-\overline{y}) = \frac{1}{n}\sum_{i=1}^{n} x_i y_i - \overline{x}\,\overline{y}$. By dividing the covariance between X and Y by $\sqrt{\sigma_X^2 \sigma_Y^2}$ we obtain the *correlation coefficient*, which is a number between –1 and +1.

Background 1.3. Expectations and estimators.

There are other ways in which features can be related. Consider the following three features that can be true or false of a molecular compound:

1. it has a carbon in a six-membered aromatic ring;
2. it has a carbon with a partial charge of –0.13;
3. it has a carbon in a six-membered aromatic ring with a partial charge of –0.13.

We say that the third feature is more *specific* (or less *general*) than the other two, because if the third feature is true, then so are the first and the second. However,

the converse does not hold: if both first and second feature are true, the third feature may still be false (because the carbon in the six-membered ring may not be the same as the one with a partial charge of –0.13). We can exploit these relationships when searching for features to add to our logical model. For instance, if we find that the third feature is true of a particular negative example that we're trying to exclude, then there is no point in considering the more general first and second features, because they will not help us in excluding the negative either. Similarly, if we find that the first feature is false of a particular positive we're trying to include, there is no point in considering the more specific third feature instead. In other words, these relationships help us to structure our search for predictive features.

1.4 Summary and outlook

My goal in this chapter has been to take you on a tour to admire the machine learning landscape, and to raise your interest sufficiently to want to read the rest of the book. Here is a summary of the things we have been looking at.

- Machine learning is about using the right features to build the right model that achieve the right tasks. These tasks include: binary and multi-class classification, regression, clustering and descriptive modelling. Models fo the first few of these tasks are learned in a supervised fashion requiring labelled training data. For instance, if you want to train a spam filter using machine learning, you need a training set of e-mails labelled spam and ha If you want to know how good the model is you also need labelled test da that is distinct from the training data, as evaluating your model on the data it was trained on will paint too rosy a picture: a test set is needed to expos any overfitting that occurs.

- Unsupervised learning, on the other hand, works with unlabelled data and so there is no test data as such. For instance, to evaluate a particular partition of data into clusters, one can calculate the average distance from the cluster centre. Other forms of unsupervised learning include learning associations (things that tend to occur together) and identifying hidden variables such as film genres. Overfitting is also a concern in unsupervise learning: for instance, assigning each data point its own cluster will reduc

the average distance to the cluster centre to zero, yet is clearly not very useful.

- On the output side we can distinguish between predictive models whose outputs involve the target variable and descriptive models which identify interesting structure in the data. Often, predictive models are learned in a supervised setting while descriptive models are obtained by unsupervised learning methods, but there are also examples of supervised learning of descriptive models (e.g., subgroup discovery which aims at identifying regions with an unusual class distribution) and unsupervised learning of predictive models (e.g., predictive clustering where the identified clusters are interpreted as classes).

- We have loosely divided machine learning models into geometric models, probabilistic models and logical models. Geometric models are constructe in Cartesian instance spaces, using geometric concepts such as planes and distances. The prototypical geometric model is the basic linear classifier, which constructs a decision plane orthogonal to the line connecting the positive and negative centres of mass. Probabilistic models view learning a process of reducing uncertainty using data. For instance, a Bayesian classifier models the posterior distribution $P(Y|X)$ (or its counterpart, the likelihood function $P(X|Y)$) which tells me the class distribution Y after observing the feature values X. Logical models are the most 'declarative' the three, employing if–then rules built from logical conditions to single o homogeneous areas in instance space.

- We have also introduced a distinction between grouping and grading models. Grouping models divide the instance space into segments which are determined at training time, and hence have a finite resolution. On eac segment, grouping models usually fit a very simple kind of model, such as 'always predict this class'. Grading models fit a more global model, grade by the location of an instance in instance space (typically, but not always, Cartesian space). Logical models are typical examples of grouping models while geometric models tend to be grading in nature, although this distinction isn't clear-cut. While this sounds very abstract at the moment, the distinction will become much clearer when we discuss coverage curve in the next chapter.

-

Last but not least, we have discussed the role of features in machine learning. No model can exist without features, and sometimes a single feature is enough to build a model. Data doesn't always come with ready-made features, and often we have to transform or even construct features. Because of this, machine learning is often an iterative process: we only know we have captured the right features after we have constructed the model, and if the model doesn't perform satisfactorily we need to analyse its performance to understand in what way the features need to be improved.

What you'll find in the rest of the book

In the next nine chapters, we will follow the structure laid out above, and look in detail at

- machine learning tasks in Chapters 2 and 3;
- logical models: concept learning in Chapter 4, tree models in Chapter 5 an rule models in Chapter 6;
- geometric models: linear models in Chapter 7 and distance-based models i Chapter 8;
- probabilistic models in Chapter 9; and
- features in Chapter 10.

Chapter 11 is devoted to techniques for training 'ensembles' of models that have certain advantages over single models. In Chapter 12 we will consider a number of methods for what machine learners call 'experiments', which involve training and evaluating models on real data. Finally, in the Epilogue we will wrap up the book and take a look ahead.

---

[1] It is a simplified version of linear discriminants.

[2]This can be expressed in vector notation as $||\mathbf{x}-\mathbf{y}|| = \sqrt{(\mathbf{x}-\mathbf{y})\cdot(\mathbf{x}-\mathbf{y})} = \sqrt{\mathbf{x}\cdot\mathbf{x}-2\mathbf{x}\cdot\mathbf{y}+\mathbf{y}\cdot\mathbf{y}} = \sqrt{||\mathbf{x}||^2-2||\mathbf{x}||||\mathbf{y}||\cos\theta+||\mathbf{y}||^2}$, where θ is the angle between **x** and **y.**

[3]It is called the likelihood function rather than the 'likelihood distribution' because, for fixed X, $P(X|Y)$ is a mapping from Y to probabilities, but these don't sum to 1 and therefore don't establish a probability distribution over Y.

[4]This formulation is often attributed to Einstein, although the source is unclear. Other rules in the same spirit include 'Entities should not be multiplied unnecessarily' (called *Occam's razor*, after William of Ockham); 'We are to admit no more causes of natural things than such as are both true and sufficient to explain their appearances' (Isaac Newton); and 'Scientists must use the simplest means of arriving at their results and exclude everything not perceived by the senses' (Ernst Mach). Whether any of these rules are more than methodological rules of thumbs and point to some fundamental property of nature is heavily debated.

[5]Sometimes we need to slightly adapt the raw counts for very frequent or very infrequent words, as we shall see in Section 2.3.

[6]Mutagenic molecules causeiontations in DNA and are often carcinogenic. This last rule refers to a set of rules that was learned earlier by Progol to predict mutagenicity.

[7]The figures have been generated from data explained in Example 1.7 below.

[8]Notice we employ two different notations for features: sometimes we write $f(x)$ if it is more convenient to view a feature as a function applied to instance x, and sometimes we write x_i if it is more convenient to view an instance as a vector of feature values.

CHAPTER 2

IN THIS CHAPTER Binary classification and related tasks and the next we take a bird's-eye view of the wide range of different tasks that can be solved with machine learning techniques. 'Task' here refers to whatever it is that machine learning is intended to improve performance of (recall the definition of machine learning on p.3), for example, e-mail spam recognition. Since this is a classification task, we need to learn an appropriate classifier from training data. Many different types of classifiers exist: linear classifiers, Bayesian classifiers, distance-based classifiers, to name a few. We will refer to these different types as models; they are the subject of Chapters 4–9. Classification is just one of a range of possible tasks for which we can learn a model: other tasks that will pass the review

in this chapter are class probability estimation and ranking. In the next chapter we will discuss regression, clustering and descriptive modelling. For each of these tasks we will discuss what it is, what variants exist, how performance at the task could be assessed, and how it relates to other tasks. We will start with some general notation that is used in this chapter and throughout the book (see Background 2.1 for the relevant mathematical concepts).

The objects of interest in machine learning are usually referred to as *instances*. The set of all possible instances is called the *instance space*, denoted $\mathscr{X}$ in this book. To illustrate, $\mathscr{X}$ could be the set of all possible e-mails that can be written using the Latin alphabet.[1] We furthermore distinguish between the *label space* $\mathscr{L}$ and the *output space* $\mathscr{Y}$. The label space is used in supervised learning to label the examples. In order to achieve the task under consideration we need a *model*: a mapping from the instance space to the output space. For instance, in classification the output space is a set of classes, while in regression it is the set of real numbers. In order to learn such a model we require a *training set* Tr of *labelled instances* $(x, l(x))$, also called *examples*, where $l : \mathscr{X} \rightarrow \mathscr{L}$ is a labelling function.

Based on this terminology and notation, and concentrating on supervised learning of predictive models for the duration of the chapter, Table 2.1 distinguishes a number of specific scenarios. The most commonly encountered machine learning scenario is where the label space coincides with the output

space. That is, $\mathscr{Y} = \mathscr{L}$ and we are trying to learn an approximation $\hat{l} : \mathscr{X} \to \mathscr{L}$ to the true labelling function l, which is only known through the labels it assigned to the training data. This scenario covers both classification and regression. In cases where the label space and the output space differ, this usually serves the purpose of learning a model that outputs more than just a label – for instance, a score for each possible label. In this case we have $\mathscr{Y} = \mathbb{R}^k$, with $k = |\mathscr{L}|$ the number of labels.

Matters may be complicated by *noise*, which can take the form of *label noise* – instead of $l = l(x)$ we observe some corrupted label l' – or *instance noise* – instead of x we observe an instance x' that is corrupted in some way. One consequence of noisy data is that it is generally not advisable to try to match the training data exactly, as this may lead to overfitting the noise. Some of the labelled data is usually set aside for evaluating or testing a classifier, in which case it is called a *test set* and denoted by Te. We use superscripts to restrict training or test set to a particular class: e.g., $Te^{\oplus} = \{(x, l(x)) | x \in Te, l(x) = \oplus\}$ is the set of positive test examples, and $Te^{\ominus}$ is the set of negative test examples.

The simplest kind of input space arises when instances are described by a fixed number of *features*, also called attributes, predictor variables, explanatory variables or independent variables. Indicating the set of values or *domain* of a feature by $\mathscr{F}_i$, we then have that $\mathscr{X} = \mathscr{F}_1 \times \mathscr{F}_2 \times \ldots \times \mathscr{F}_d$, and thus every instance is a d-vector of feature values. In some domains the features to use readily suggest themselves, whereas in other domains they need to be constructed. For example, in the spam filter example in the Prologue we constructed a large number of features, one for each word in a vocabulary, counting the number of occurrences of that word in the e-mail. Even when features are given explicitly we often want to transform them to maximise their usefulness for the task at hand. We will discuss this in considerable detail in Chapter 10.

We briefly review some important concepts from discrete mathematics. A *set* is a collection of objects, usually of the same kind (e.g., the set of all natural numbers $\mathbb{N}$ or the set of real numbers $\mathbb{R}$). We write $x \in A$ if x is an element of set A, and $A \subseteq B$ if all elements of A are also elements of B (this includes the possibility that A and B are the same set, which is equivalent to $A \subseteq B$ and $B \subseteq A$). The *intersection* and *union* of two sets are defined as $A \cap B = \{x | x \in A \text{ and } x \in B\}$ and $A \cup B = \{x | x \in A \text{ or } x \in B\}$. The *difference* of two sets is defined as $A \setminus B = \{x | x \in A \text{ and } x \notin B\}$. It is customary to fix a *universe of discourse* U such that all sets under consideration are subsets of U. The

complement of a set A is defined as $\overline{A} = U \setminus A$. Two sets are *disjoint* if their intersection is empty: $A \cap B = \emptyset$. The *cardinality* of a set A is its number of elements and is denoted $|A|$. The *powerset* of a set A is the set of all its subsets $2^A = \{B | B \subseteq A\}$; its cardinality is $|2^A| = 2^{|A|}$. The *characteristic function* of a set A is the function $f : U \rightarrow \{\text{true}, \text{false}\}$ such that $f(x) = \text{true}$ if $x \in A$ and $f(x) = \text{false}$ if $x \in U \setminus A$.

If A and B are sets, the *Cartesian product* $A \times B$ is the set of all pairs $\{(x, y) | x \in A \text{ and } y \in B\}$; this generalises to products of more than two sets. A (binary) *relation* is a set of pairs $R \subseteq A \times B$ for some sets A and B; if $A = B$ we say the relation is over A. Instead of $(x, y) \in R$ we also write xRy. A relation over A is *(i) reflexive* if xRx for all $x \in A$; *(ii) symmetric* if xRy implies yRx for all $x, y \in A$; *(iii) antisymmetric* if xRy and yRx implies $x = y$ for all $x, y \in A$; *(iv) transitive* if xRy and yRz implies xRz for all $x, y, z \in A$. *(v) total* if xRy or yRx for all $x, y \in A$.

A *partial order* is a binary relation that is reflexive, antisymmetric and transitive. For instance, the *subset* relation $\subseteq$ is a partial order. A *total order* is a binary relation that is total (hence reflexive), antisymmetric and transitive. The $\leq$ relation on real numbers is a total order. If xRy or yRx we say that x and y are *comparable*; otherwise they are *incomparable*. An *equivalence relation* is a binary relation $\equiv$ that is reflexive, symmetric and transitive. The *equivalence class* of x is $[x] = \{y | x \equiv y\}$. For example, the binary relation 'contains the same number of elements as' over any set is an equivalence relation. Any two equivalence classes are disjoint, and the union of all equivalence classes is the whole set – in other words, the set of all equivalence classes forms a *partition* of the set. If $A_1, \ldots, A_n$ is a partition of a set A, *i.e.* $A_1 \cup \ldots \cup A_n = A$ and $A_i \cap A_j = \emptyset$ for all $i \neq j$,we write $A = A_1 \uplus \ldots \uplus A_n$.

To illustrate this, let T be a feature tree, and define a relation $\sim_T \subseteq \mathscr{X} \times \mathscr{X}$ such that $x \sim T\, x'$ if and only if x and x' are assigned to the same leaf of feature tree T, then $\sim_T$ is an equivalence relation, and its equivalence classes are precisely the instance space segments associated with T.

Background 2.1. Useful concepts from discrete mathematics.

The sections in this chapter are devoted to the first three scenarios in Table 2.1: classification in Section 2.1, scoring and ranking in Section 2.2 and class probability estimation in Section 2.3. To keep things manageable we mostly restrict attention to two-class tasks in this chapter and deal with more than two

classes in Chapter 3. Regression, unsupervised and descriptive learning will also be considered there.

Task	*Label space*	*Output space*	*Learning problem*
Classification	$\mathscr{L} = \mathscr{C}$	$\mathscr{Y} = \mathscr{C}$	learn an approximation $\hat{c} : \mathscr{X} \to \mathscr{C}$ to the true labelling function c
Scoring and ranking	$\mathscr{L} = \mathscr{C}$	$\mathscr{Y} = \mathbb{R}^{\lvert\mathscr{C}\rvert}$	learn a model that outputs a score vector over classes
Probability estimation	$\mathscr{L} = \mathscr{C}$	$\mathscr{Y} = [0,1]^{\lvert\mathscr{C}\rvert}$	learn a model that outputs a probability vector over classes
Regression	$\mathscr{L} = \mathbb{R}$	$\mathscr{Y} = \mathbb{R}$	learn an approximation $\hat{f} : \mathscr{X} \to \mathbb{R}$ to the true labelling function f

Table 2.1. Predictive machine learning scenarios.

Throughout this chapter I will illustrate key concepts by means of examples using simple models of the kind discussed in the Prologue. These models will either be simple tree-based models, representative of grouping models, or linear models, representative of grading models. Sometimes we will even construct models from single features, a setting that could be described as *univariate machine learning*. We will start dealing with the question of how to *learn* such models from Chapter 4 onwards.

2.1 Classification

Classification is the most common task in machine learning. A *classifier* is a mapping $\hat{c} : \mathscr{X} \to \mathscr{C}$, where $\mathscr{C} = \{C_1, C_2, \ldots, C_k\}$ is a finite and usually small set of *class labels*. We will sometimes also use C_i to indicate the set of examples of that class. We use the 'hat' to indicate that $\hat{c}(x)$ is an estimate of the true but unknown function $c(x)$. Examples for a classifier take

the form $(x, c(x))$, where $x \in \mathscr{X}$ is an instance and $c(x)$ is the true class of the instance. Learning a classifier involves constructing the function $\hat{c}$ such that it matches c as closely as possible (and not just on the training set, but ideally on the entire instance space $\mathscr{X}$).

In the simplest case we have only two classes which are usually referred to as *positive* and *negative*, $\oplus$ and $\ominus$, or +1 and –1. Two-class classification is often called *binary classification* (or *concept learning*, if the positive class can be meaningfully called a concept). Spam e-mail filtering is a good example of binary classification, in which spam is conventionally taken as the positive class, and ham as the negative class (clearly, positive here doesn't mean 'good'!). Other examples of binary classification include medical diagnosis (the positive class here is having a particular disease) and credit card fraud detection.

Figure 2.1. (left) A feature tree with training set class distribution in the leaves. **(right)** A decision tree obtained using the majority class decision rule.

The feature tree in Figure 2.1 (left) can be turned into a classifier by labelling each leaf with a class. The simplest way to do this is by assigning the *majority class* in each leaf, resulting in the decision tree in Figure 2.1 (right). The classifier works as follows: if an e-mail contains the word 'Viagra' it is classified as spam (right-most leaf); otherwise, the occurrence of the word 'lottery' decides

whether it gets labelled spam or ham.[2] From the numbers in Figure 2.1 we can get an idea how well this classifier does. The left-most leaf correctly predicts 40 ham e-mails but also mislabels 20 spam e-mails that contain neither 'Viagra' nor 'lottery'. The middle leaf correctly classifies 10 spam e-mails but also erroneously labels 5 ham e-mails as spam. The 'Viagra' test correctly picks out 20 spam e-mails but also 5 ham e-mails. Taken together, this means that 30 out of 50 spam e-mails are classified correctly, and 40 out of 50 ham e-mails.

Assessing classification performance *The performance of such classifiers can be summarised by means of a table known as a contingency table or confusion matrix (Table 2.2 (left)). In this table, each row refers to actual classes as recorded in the test set, and each column to classes as predicted by the classifier. So, for instance, the first row states that the test set contains 50 positives, 30 of which were correctly predicted and 20 incorrectly. The last column and the last row give the marginals (i.e., column and row sums). Marginals are important because they allow us to assess statistical significance. For instance, the contingency table in Table 2.2 (right) has the same marginals, but the classifier clearly makes a random choice as to which predictions are positive and which are negative – as a result the distribution of actual positives and negatives in either predicted class is the same as the overall distribution (uniform in this case).*

	Predicted ⊕	*Predicted* ⊖	
Actual ⊕	30	20	50
Actual ⊖	10	40	50
	40	60	100

	⊕	⊖	
⊕	20	30	50
⊖	20	30	50
	40	60	100

Table 2.2. (left) A two-class contingency table or confusion matrix depicting the performance of the decision tree in Figure 2.1. Numbers on the descending diagonal indicate correct predictions, while the ascending diagonal concerns prediction errors. **(right)** A contingency table with the same marginals but independent rows and columns.

From a contingency table we can calculate a range of performance indicators.

The simplest of these is *accuracy*, which is the proportion of correctly classified test instances. In the notation introduced at the beginning of this chapter, accuracy over a test set *Te* is defined as

$$acc = \frac{1}{|Te|} \sum_{x \in Te} I[\hat{c}(x) = c(x)] \qquad (2.1)$$

Here, the function I [·] denotes the *indicator function*, which is 1 if its argument evaluates to true, and 0 otherwise. In this case it is a convenient way to count the number of test instances that are classified correctly by the classifier (i.e., the estimated class label $\hat{c}(x)$ is equal to the true class label $c(x)$). For example, in Table 2.2 (left) the accuracy of the classifier is 0.70 or 70%, and in Table 2.2 (right) it is 0.50. Alternatively, we can calculate the *error rate* as the proportion of incorrectly classified instances, here 0.30 and 0.50, respectively. Clearly, accuracy and error rate sum to 1.

Test set accuracy can be seen as an *estimate* of the probability that an arbitrary instance $x \in \mathscr{X}$ is classified correctly: more precisely, it estimates the probability $P_{\mathscr{X}}(\hat{c}(x) = c(x))$ (Notice that I write $P_{\mathscr{X}}$ to emphasise that this is a probability distribution over the instance space $\mathscr{X}$; I will often omit subscripts if this is clear from the context.) We typically only have access to the true classes of a small fraction of the instance space and so an estimate is all we can hope to get. It is therefore important that the test set is as representative as possible. This is usually formalised by the assumption that the occurrence of instances in the world – i.e., how likely or typical a particular e-mail is – is governed by an unknown probability distribution on $\mathscr{X}$, and that the test set *Te* is generated according to this distribution.

It is often convenient – not to say necessary – to distinguish performance on the classes. To this end, we need some further terminology. Correctly classified positives and negatives are referred to as *true positives* and *true negatives*, respectively. Incorrectly classified positives are, perhaps somewhat confusingly, called *false negatives*; similarly, misclassified negatives are called *false positives*. A good way to think of this is to remember that positive/negative refers to the classifier's prediction, and true/false refers to whether the prediction is correct or not. So, a false positive is something that was incorrectly predicted as positive, and therefore an actual negative (e.g., a ham email misclassified as spam, or a healthy patient misclassified as having the disease in question). In the previous example (Table 2.2 (left)) we have 30 true positives, 20 false negatives, 40 true negatives and 10 false positives.

The *true positive rate* is the proportion of positives correctly classified, and can be defined mathematically as

$$tpr = \frac{\sum_{x \in Te} I[\hat{c}(x) = c(x) = \oplus]}{\sum_{x \in Te} I[c(x) = \oplus]} \tag{2.2}$$

True positive rate is an estimate of the probability that an arbitrary positive is classified correctly, that is, an estimate of $P_{\mathscr{X}}(\hat{c}(x) = \oplus | c(x) = \oplus)$. Analogously, the *true negative rate* is the proportion of negatives correctly classified (see Table 2.3 on p.57 for the mathematical definition), and estimates $P_{\mathscr{X}}(\hat{c}(x) = \ominus | c(x) = \ominus)$. These rates, which are sometimes called *sensitivity* and *specificity*, can be seen as per-class accuracies. In the contingency table, the true positive and negative rates can be calculated by dividing the number on the descending (good) diagonal by the row total. We can also talk about per-class error rates, which is the *false negative rate* for the positives (i.e., the number of misclassified positives or false negatives as a proportion of the total number of positives) and the *false positive rate* for the negatives (sometimes called the *false alarm rate*). These rates can be found by dividing the number on the ascending (bad) diagonal by the row total.

In Table 2.2 (left) we have a true positive rate of 60%, a true negative rate of 80%, a false negative rate of 40% and a false positive rate of 20%. In Table 2.2 (right) we have a true positive rate of 40%, a true negative rate of 60%, a false negative rate of 60% and a false positive rate of 40%. Notice that the accuracy in both cases is the average of the true positive rate and the true negative rate (and the error rate is the average of the false positive rate and the false negative rate). However, this is true only if the test set contains equal numbers of positives and negatives – in the general case we need to use a *weighted* average, where the weights are the proportions of positives and negatives in the test set.

Example 2.1 (Accuracy as a weighted average). Suppose a classifier's predictions on a test set are as in the following table:

	Predicted $\oplus$	*Predicted* $\ominus$	
Actual $\oplus$	60	15	75
Actual $\ominus$	10	15	25
	70	30	100

From this table, we see that the true positive rate is $tpr = 60/75 = 0.80$ and the true negative rate is $tnr = 15/25 = 0.60$. The overall accuracy is $acc = (60 + 15)/100 = 0.75$, which is no longer the average of true positive and negative rates. However, taking into account the proportion of positives $pos = 0.75$ and the proportion of negatives $neg = 1 - pos = 0.25$, we see that

$$acc = pos \cdot tpr + neg \cdot tnr \tag{2.3}$$

This equation holds in general: if the numbers of positives and negatives are equal, we obtain the unweighted average from the earlier example ($acc = (tpr + tnr)/2$).

Equation 2.3 has a neat intuition: good performance on either class contributes to good classification accuracy, but the more prevalent class contributes more strongly. In order to achieve good accuracy, a classifier should concentrate on the *majority class*, particularly if the class distribution is highly unbalanced. However, it is often the case that the majority class is also the least interesting class. To illustrate, suppose you issue a query to an internet search engine,[3] and suppose that for that particular query there is only one relevant page in every 1 000 web pages. Now consider a 'reluctant' search engine that doesn't return *any* answers – i.e., it classifies every web page as irrelevant to your query. Consequently, it will achieve 0% true positive rate and 100% true negative rate. Because $pos = 1/1000 = 0.1\%$ and $neg = 99.9\%$, the reluctant search engine's accuracy is very high (99.9%). Put differently, if we select a random web page uniformly over all web pages, the probability of selecting a positive is only 0.001, and these are the only pages on which the reluctant engine makes an error. However, we are not normally selecting pages from the web uniformly, and hence accuracy is not a meaningful quantity in this context. To be of any use at all, a search engine should achieve a much better true positive rate, which usually comes at the expense of a worse true negative rate (and hence a drop in accuracy).

Measure	*Definition*	*Equal to*	*Estimates*
number of positives	$Pos = \sum_{x \in Te} I[c(x) = \oplus]$		
number of negatives	$Neg = \sum_{x \in Te} I[c(x) = \ominus]$	$\lvert Te \rvert - Pos$	
number of true positives	$TP = \sum_{x \in Te} I[\hat{c}(x) = c(x) = \oplus]$		
number of true negatives	$TN = \sum_{x \in Te} I[\hat{c}(x) = c(x) = \ominus]$		
number of false positives	$FP = \sum_{x \in Te} I[\hat{c}(x) = \oplus, c(x) = \ominus]$	$Neg - TN$	
number of false negatives	$FN = \sum_{x \in Te} I[\hat{c}(x) = \ominus, c(x) = \oplus]$	$Pos - TP$	
proportion of positives	$pos = \frac{1}{\lvert Te \rvert} \sum_{x \in Te} I[c(x) = \oplus]$	$Pos / \lvert Te \rvert$	$P(c(x) = \oplus)$
proportion of negatives	$neg = \frac{1}{\lvert Te \rvert} \sum_{x \in Te} I[c(x) = \ominus]$	$1 - pos$	$P(c(x) = \ominus)$
class ratio	$clr = pos / neg$	Pos / Neg	
(*) accuracy	$acc = \frac{1}{\lvert Te \rvert} \sum_{x \in Te} I[\hat{c}(x) = c(x)]$		$P(\hat{c}(x) = c(x))$
(*) error rate	$err = \frac{1}{\lvert Te \rvert} \sum_{x \in Te} I[\hat{c}(x) \neq c(x)]$	$1 - acc$	$P(\hat{c}(x) \neq c(x))$
true positive rate, sensitivity, recall	$tpr = \frac{\sum_{x \in Te} I[\hat{c}(x) = c(x) = \oplus]}{\sum_{x \in Te} I[c(x) = \oplus]}$	TP / Pos	$P(\hat{c}(x) = \oplus \mid c(x) = \oplus)$
true negative rate, specificity, negative recall	$tnr = \frac{\sum_{x \in Te} I[\hat{c}(x) = c(x) = \ominus]}{\sum_{x \in Te} I[c(x) = \ominus]}$	TN / Neg	$P(\hat{c}(x) = \ominus \mid c(x) = \ominus)$
false positive rate, false alarm rate	$fpr = \frac{\sum_{x \in Te} I[\hat{c}(x) = \oplus, c(x) = \ominus]}{\sum_{x \in Te} I[c(x) = \ominus]}$	$FP / Neg = 1 - tnr$	$P(\hat{c}(x) = \oplus \mid c(x) = \ominus)$
false negative rate	$fnr = \frac{\sum_{x \in Te} I[\hat{c}(x) = \ominus, c(x) = \oplus]}{\sum_{x \in Te} I[c(x) = \oplus]}$	$FN / Pos = 1 - tpr$	$P(\hat{c}(x) = \ominus \mid c(x) = \oplus)$
precision, confidence	$prec = \frac{\sum_{x \in Te} I[\hat{c}(x) = c(x) = \oplus]}{\sum_{x \in Te} I[\hat{c}(x) = \oplus]}$	$TP / (TP + FP)$	$P(c(x) = \oplus \mid \hat{c}(x) = \oplus)$

Table 2.3. A summary of different quantities and evaluation measures for classifiers on a test set *Te*. Symbols starting with a capital letter denote absolute frequencies (counts), while lower-case symbols denote relative frequencies or ratios. All except those indicated with (*) are defined only for binary classification. The right-most column specifies the instance space probabilities that these relative frequencies are estimating.

We conclude from this example that, if the minority class is the class of interest and very small, accuracy and performance on the majority class are not the right quantities to optimise. For this reason, an alternative to true negative rate called *precision* is usually considered in such cases. Precision is a counterpart to true positive rate in the following sense: while true positive rate is the proportion of predicted positives among the actual positives, precision is the proportion of actual positives among the predicted positives. In Example 2.1 the classifier's precision on the test set is 60/70 = 85.7%. In the reluctant search engine example we have not only 0 true positive rate (which in this context is usually called *recall*) but also 0 precision, which clearly demonstrates the problem with a search engine that doesn't return any answers. Table 2.3 summarises the evaluation measures introduced in this section.

Figure 2.2. (left) A coverage plot depicting the two contingency tables in Table 2.2. The plot is square because the class distribution is uniform. **(right)** Coverage plot for Example 2.1, with a class ratio *clr* = 3.

Visualising classification performance I will now introduce an important tool for visualising the performance of classifiers and other models called a coverage plot. If you look at two-class contingency tables such as the ones depicted in Table 2.2, you realise that, even though the table contains nine numbers, only four of those can be chosen freely. For instance, once you've determined the true/false positives/negatives, the marginals are fixed. Or if you know the true positives, true negatives, total number of positives and size of the test set, you can reconstruct all other numbers. Statisticians say that the table has four degrees of freedom.[4]

Often we are particularly interested in the following four numbers that completely determine the contingency table: the number of positives *Pos*, the number of negatives *Neg*, the number of true positives *TP* and the number of false positives *FP*. A coverage plot visualises these four numbers by means of a rectangular coordinate system and a point. Imagine a rectangle with height *Pos* and width *Neg*. Imagine furthermore that all positives live on the *y*-axis of this rectangle, and all negatives on the *x*-axis. We don't really care how positives and negatives are ordered on their respective axes, as long as *positive predictions*

come before negative predictions. This gives us enough information to depict the whole contingency table as a single point within the rectangle (Figure 2.2).

Figure 2.3. (left) C1 and C3 both dominate C2, but neither dominates the other. The diagonal line indicates that C1 and C3 achieve equal accuracy. **(right)** The same plot with normalised axes. We can interpret this plot as a merger of the two coverage plots in Figure 2.2, employing normalisation to deal with the different class distributions. The diagonal line now indicates that C1 and C3 have the same average recall.

Consider the two classifiers marked C1 and C2 in Figure 2.2 (left). One reason why coverage plots are so useful is that we can immediately see that C1 is better than C2. How do we know that? Well, C1 has both more true positives and fewer false positives than C2, and so is better in both respects. Put differently, C1 achieves better performance than C2 on *both* classes. If one classifier outperforms another classifier on all classes, the first one is said to *dominate* the second.[5] However, things are not always that straightforward. Consider a third classifier C3, better than C1 on the positives but worse on the negatives (Figure 2.3 (left)). Although both C1 and C3 dominate C2, neither of them dominates the other. Which one we prefer depends on whether we put more emphasis on the positives or on the negatives.

We can make this a little bit more precise. Notice that the line segment connecting C1 and C3 has a slope of 1. Imagine travelling up that line: whenever we gain a true positive, we also lose a true negative (or gain a false positive, which is the same thing). This doesn't affect the sum of true positives and true negatives, and hence the accuracy is the same wherever we are on the line. It follows that C1 and C3 have the same accuracy. *In a coverage plot, classifiers with the same accuracy are connected by line segments with slope 1*. If true positives and true negatives are equally important, the choice between C1 and C3 is arbitrary; if true positives are more important we should choose C3, if true negatives are more important we prefer C1.

Now consider Figure 2.3 (right). What I have done here is renormalise the axes

by dividing the x-axis by *Neg* and the y-axis by *Pos*, resulting in a plot in the unit square with true positive rate on the y-axis and false positive rate on the x-axis. In this case the original coverage plot was already square (*Pos* = *Neg*), so the relative position of the classifiers isn't affected by the normalisation. However, since the normalised plot will be square regardless of the shape of the original plot, normalisation is a way to combine differently shaped coverage plots, and thus to combine results on test sets with different class distributions. Suppose you would normalise Figure 2.2 (right): since C3's true and false positive rates are 80% and 40%, respectively (see Example 2.1 on p.56), its position in a normalised plot is exactly the same as the one labelled C3 in Figure 2.3 (right)! In other words, classifiers occupying different points in different coverage spaces (e.g., C3 in Figure 2.2 (right) and C3 in Figure 2.3 (left)) can end up in the same point in a normalised plot.

What is the meaning of the diagonal line connecting C1 and C3 in Figure 2.3 (right)? It can't have the same meaning as in the coverage plot, because in a normalised plot we know the true and false positive rates but not the class distribution, and so we cannot calculate accuracy (refer back to Equation 2.3 on p.56 if you want to remind yourself why). The line is defined by the equation $tpr = fpr + y_0$, where y_0 is the y-intercept (the value of *tpr* where the line intersects the y-axis) . Now consider the average of the true positive rate and the true negative rate, which we will call *average recall*, denoted *avg-rec*.[6] On a line with slope 1 we have $avg\text{-}rec = (tpr + tnr)/2 = (tpr + 1 - fpr)/2 = (1 + y_0)/2$, which is a constant. *In a normalised coverage plot, line segments with slope 1 connect classifiers with the same average recall*. If recall on the positives and the negatives are equally important, the choice between C1 and C3 is arbitrary; if positive recall is more important we should choose C3, if negative recall is more important we prefer C1.

In the literature, normalised coverage plots are referred to as *ROC plots*, and we will follow that convention from now on.[7] ROC plots are much more common than coverage plots, but both have their specific uses. Broadly speaking, you should use a coverage plot if you explicitly want to take the class distribution into account, for instance when you are working with a single data set. An ROC plot is useful if you want to combine results from different data sets with different class distributions. Clearly, there are many connections between the two. Since an ROC plot is always square, lines of constant average recall (so-called average recall *isometrics*) do not only have a slope of 1 but are parallel to the ascending diagonal. The latter property carries over to coverage plots. To illustrate, in the coverage plot in Figure 2.4, C1 and C2 have the same accuracy

(they are connected by a line segment with slope 1), and C1 and C3 have the same average recall (they are connected by a line segment parallel to the diagonal). You can also argue that C2 has both higher accuracy and higher average recall than C3 (why?). In the corresponding ROC plot, the average recall isometric has a slope of 1, and the accuracy isometric's slope is *Neg/Pos* = 1/*clr*.

Figure 2.4. (left) In a coverage plot, accuracy isometrics have a slope of 1, and average recall isometrics are parallel to the ascending diagonal. **(right)** In the corresponding ROC plot, average recall isometrics have a slope of 1; the accuracy isometric here has a slope of 3, corresponding to the ratio of negatives to positives in the data set.

2.2 Scoring and ranking

Many classifiers compute scores on which their class predictions are based. For instance, in the Prologue we saw how SpamAssassin calculates a weighted sum from the rules that 'fire' for a particular e-mail. Such scores contain additional information that can be beneficial in a number of ways, which is why we perceive scoring as a task in its own right. Formally, a ***scoring classifier*** is a mapping $\hat{\mathbf{s}}: \mathscr{X} \rightarrow \mathbb{R}^k$, i.e., a mapping from the instance space to a *k*-vector of real numbers. The boldface notation indicates that a scoring classifier

outputs a vector $\hat{\mathbf{s}}(x) = (\hat{s}_1(x), \ldots, \hat{s}_k(x))$ rather than a single number; $\hat{s}_i(x)$ is the score assigned to class C_i for instance x. This score indicates how likely it is that class label C_i applies. If we only have two classes, it usually suffices to consider the score for only one of the classes; in that case, we use $\hat{s}(x)$ to denote the score of the positive class for instance x.

Figure 2.5 demonstrates how a feature tree can be turned into a scoring tree. In order to obtain a score for each leaf, we first calculate the ratio of spam to ham, which is 1/2 for the left leaf, 2 for the middle leaf and 4 for the right leaf. Because it is often more convenient to work with an additive scale, we obtain scores by taking the logarithm of the class ratio (the base of the logarithm is not really important; here we have taken base-2 logarithms to get nice round numbers). Notice that the majority class decision tree corresponds to thresholding $\hat{s}(x)$ at 0: i.e., predict spam if $\hat{s}(x) > 0$ and ham otherwise.

Figure 2.5. (left) A feature tree with training set class distribution in the leaves. **(right)** A scoring tree using the logarithm of the class ratio as scores; spam is taken as the positive class.

If we take the true class $c(x)$ as +1 for positive examples and –1 for negative examples, then the quantity $z(x) = c(x)\hat{s}(x)$ is positive for correct predictions and

negative for incorrect predictions: this quantity is called the *margin* assigned by the scoring classifier to the example.[8] We would like to reward large positive margins, and penalise large negative values. This is achieved by means of a so-called *loss function* $L: \mathbb{R} \mapsto [0, \infty)$ which maps each example's margin $z(x)$ to an associated loss $L(z(x))$. We will assume that $L(0) = 1$, which is the loss incurred by having an example on the decision boundary. We furthermore have $L(z) \geq 1$ for $z < 0$, and usually also $0 \leq L(z) < 1$ for $z > 0$ (Figure 2.6). The average loss over a test set Te is $\frac{1}{|Te|} \sum_{x \in Te} L(z(x))$.

The simplest loss function is *0–1 loss*, which is defined as $L_{01}(z) = 1$ if $z \leq 0$ and $L(z) = 0$ if $z > 0$. The average 0–1 loss is simply the proportion of misclassified test examples:

$$\frac{1}{|Te|} \sum_{x \in Te} L_{01}(z(x)) = \frac{1}{|Te|} \sum_{x \in Te} I[c(x)\hat{s}(x) \leq 0] = \frac{1}{|Te|} \sum_{x \in Te} I[c(x) \neq \hat{c}(x)] = err$$

where $\hat{c}(x) = +1$ if $\hat{s}(x) > 0$, $\hat{c}(x) = 0$ if $\hat{s}(x) = 0$, and $\hat{c}(x) = -1$ if $\hat{s}(x) < 0$. (It is sometimes more convenient to define the loss of examples on the decision boundary as 1/2). In other words, 0–1 loss ignores the magnitude of the margins of the examples, only taking their sign into account. As a result, 0–1 loss doesn't distinguish between scoring classifiers, as long as their predictions agree. This means that it isn't actually that useful as a search heuristic or objective function when learning scoring classifiers. Figure 2.6 pictures several loss functions that are used in practice. Except for 0–1 loss, they are all *convex*: linear interpolation between any two points on the curve will never result in a point below the curve. Optimising a convex function is computationally more tractable.

Figure 2.6. Loss functions: from bottom-left (*i*) 0–1 loss $L_{01}(z) = 1$ if $z \leq 0$, and $L_{01}(z) = 0$ if $z > 0$; (*ii*) hinge loss $L_h(z) = (1 - z)$ if $z \leq 1$, and $L_h(z) = 0$ if $z > 1$; (*iii*) logistic loss $L_{log}(z) = \log_2(1 + \exp(-z))$; (*iv*) exponential loss $L_{exp}(z) = \exp(-z)$; (*v*) squared loss $L_{sq}(z) = (1 - z)^2$ (this can be set to 0 for $z > 1$, just like

hinge loss).

One loss function that will be of interest later is the *hinge loss*, which is defined as $L_h(z) = (1 - z)$ if $z \leq 1$, and $L_h(z) = 0$ if $z > 1$. The name of this loss function comes from the fact that the loss 'hinges' on whether an example's margin is greater than 1 or not: if so (i.e., the example is on the correct side of the decision boundary with a distance of at least 1) the example incurs zero loss; if not, the loss increases with decreasing margin. In effect, the loss function expresses that it is important to avoid examples having a margin (much) less than 1, but no additional value is placed on achieving large positive margins. This loss function is used when training a ☞*support vector machine* (Section 7.3). We will also encounter *exponential loss* later when we discuss ☞*boosting* in Section 11.2.

Assessing and visualising ranking performance It should be kept in mind that scores are assigned by a classifier, and are not a property inherent to instances. Scores are not estimated from 'true scores' – rather, a scoring classifier has to be learned from examples in the form of instances x labelled with classes c(x), just as a classifier. (The task where we learn a function $\hat{f}$ from examples labelled with true function values (x, f (x)) is called ☞regression and is covered in Section 3.2.) Often it is more convenient to keep the order imposed by scores on a set of instances, but ignore their magnitudes – this has the advantage, for instance, of being much less sensitive to outliers. It also means that we do not have to make any assumptions about the scale on which scores are expressed: in particular, a ranker does not assume a particular score threshold for separating positives from negatives. A ranking is defined as a total order on a set of instances, possibly with ties.[9]

Example 2.2 (Ranking example). The scoring tree in Figure 2.5 produces the following ranking: [20+, 5–] [10+, 5–] [20+, 40–]. Here, 20+ denotes a

sequence of 20 positive examples, and instances in square brackets […] are tied. By selecting a split point in the ranking we can turn the ranking into a classification. In this case there are four possibilities: (A) setting the split point before the first segment, and thus assigning all segments to the negative class; (B) assigning the first segment to the positive class, and the other two to the negative class; (C) assigning the first two segments to the positive class; and (D) assigning all segments to the positive class. In terms of actual scores, this corresponds to (A) choosing any score larger than 2 as the threshold; (B) choosing a threshold between 1 and 2; (C) setting the threshold between –1 and 1; and (D) setting it lower than –1.

Suppose x and x' are two instances such that x receives a lower score: $\hat{s}(x) < \hat{s}(x')$. Since higher scores express a stronger belief that the instance in question is positive, this would be fine except in one case: if x is an actual positive and x' is an actual negative. We will call this a *ranking error*. The total number of ranking errors can then be expressed as $\sum_{x \in Te^{\oplus}, x' \in Te^{\ominus}} I[\hat{s}(x) < \hat{s}(x')]$. Furthermore, for every positive and negative that receive the same score – a *tie* – we count half a ranking error. The maximum number of ranking errors is equal to $|Te^{\oplus}| \cdot |Te^{\ominus}| = Pos \cdot Neg$, and so the *ranking error rate* is defined as

$$rank\text{-}err = \frac{\sum_{x \in Te^{\oplus}, x' \in Te^{\ominus}} I[\hat{s}(x) < \hat{s}(x')] + \frac{1}{2} I[\hat{s}(x) = \hat{s}(x')]}{Pos \cdot Neg} \qquad (2.4)$$

and analogously the *ranking accuracy*

$$rank\text{-}acc = \frac{\sum_{x \in Te^{\oplus}, x' \in Te^{\ominus}} I[\hat{s}(x) > \hat{s}(x')] + \frac{1}{2} I[\hat{s}(x) = \hat{s}(x')]}{Pos \cdot Neg} = 1 - rank\text{-}err \qquad (2.5)$$

Ranking accuracy can be seen as an estimate of the probability that an arbitrary positive–negative pair is ranked correctly.

Example 2.3 (Ranking accuracy). We continue the previous example considering the scoring tree in Figure 2.5, with the left leaf covering 20 spam and 40 ham, the middle leaf 10 spam and 5 ham, and the right leaf 20 spam and 5 ham. The 5 negatives in the right leaf are scored higher than the 10 positives in the middle leaf and the 20 positives in the left leaf, resulting in 50 + 100 = 150 ranking errors. The 5 negatives in the middle leaf are scored higher than the 20 positives in the left leaf, giving a further 100

ranking errors. In addition, the left leaf makes 800 half ranking errors (because 20 positives and 40 negatives get the same score), the middle leaf 50 and the right leaf 100. In total we have 725 ranking errors out of a possible $50 \cdot 50 = 2500$, corresponding to a ranking error rate of 29% or a ranking accuracy of 71%.

The coverage plots and ROC plots introduced in the previous section for visualising classifier performance provide an excellent tool for visualising ranking performance too. If *Pos* positives and *Neg* negatives are plotted on the vertical and horizontal axes, respectively, then each positive–negative pair occupies a unique 'cell' in this plot. If we order the positives and negatives on decreasing score, i.e., examples with higher scores are closer to the origin, then we can clearly distinguish the correctly ranked pairs at the bottom right, the ranking errors at the top left, and the ties in between (Figure 2.7). The number of cells in each area gives us the number of correctly ranked pairs, ranking errors and ties, respectively. The diagonal lines cut the ties area in half, so the area below those lines corresponds to the ranking accuracy multiplied by $Pos \cdot Neg$, and the area above corresponds to the ranking error rate times that same factor.

Concentrating on those diagonal lines gives us the piecewise linear curve shown in Figure 2.7 (right). This curve, which we will call a *coverage curve*, can be understood as follows. Each of the points marked A, B, C and D specifies the classification performance, in terms of true and false positives, achieved by the corresponding ranking split points or score thresholds from Example 2.2. To illustrate, C would be obtained by a score threshold of 0, leading to $TP2 = 20 + 10 = 30$ true positives and $FP2 = 5 + 5 = 10$ false positives. Similarly, B would be obtained by a higher threshold of 1.5, leading to $TP1 = 20$ true positives and $FP1 = 5$ false positives. Point A would result if we set the threshold unattainably high, and D if we set the threshold trivially low.

Why are these points connected by straight lines? How can we interpolate between, say, points C and D? Suppose we set the threshold exactly at –1, which is the score assigned by the left leaf of the tree. The question is now what class we predict for the 20 positives and 40 negatives that filter down to that leaf. It would seem reasonable to decide this by tossing a fair coin, leading to half of the positives receiving a positive prediction (on average) and half of them a negative one, and similar for the negatives. The total number of true positives is then $30 + 20/2 = 40$, and the number of false positives is $10 + 40/2 = 30$. In other words,

we land exactly in the middle of the CD line segment. We can apply the same procedure to achieve performance half-way BC, by setting the threshold at 1 and tossing the same fair coin to obtain uniformly distributed predictions for the 10 positives and 5 negatives in the middle leaf, leading to 20 + 10/2 = 25 true positives and 5 + 5/2 = 7.5 false positives (of course, we cannot achieve a non-integer number of false positives in any trial, but this number represents the expected number of false positives over many trials). And what's more, by biasing the coin towards positive or negative predictions we can achieve expected performance anywhere on the line.

Figure 2.7. (left) Each cell in the grid denotes a unique pair of one positive and one negative example: the green cells indicate pairs that are correctly ranked by the classifier, the red cells represent ranking errors, and the orange cells are half-errors due to ties. **(right)** The coverage curve of a tree-based scoring classifier has one line segment for each leaf of the tree, and one (*FP,TP*) pair for each possible threshold on the score.

More generally, a coverage curve is a piecewise linear curve that rises monotonically from (0,0) to (*Neg,Pos*) – i.e., *TP* and *FP* can never decrease if we decrease the decision threshold. Each segment of the curve corresponds to an equivalence class of the instance space partition induced by the model in question (e.g., the leaves of a feature tree). Notice that the number of segments is never more than the number of test instances. Furthermore, the slope of each segment is equal to the ratio of positive to negative test instances in that equivalence class. For instance, in our example the first segment has a slope of 4, the second segment slope 2, and the third segment slope 1/2 — exactly the scores assigned in each leaf of the tree! This is not true in general, since the coverage curve depends solely on the ranking induced by the scores, not on the scores themselves. However, it is not a coincidence either, as we shall see in the next section on class probability estimation.

Figure 2.8. (left) A coverage curve obtained from a test set with class ratio $clr = 1/2$. **(right)** The corresponding ROC curve is the same as the one corresponding to the coverage curve in Figure 2.7 (right).

An *ROC curve* is obtained from a coverage curve by normalising the axes to [0, 1]. This doesn't make much of a difference in our running example, but in general coverage curves can be rectangular whereas ROC curves always occupy the unit square. One effect this has is that slopes are multiplied by $Neg/Pos = 1/clr$. Furthermore, while in a coverage plot the area under the coverage curve gives the absolute number of correctly ranked pairs, in an ROC plot *the area under the ROC curve is the ranking accuracy* as defined in Equation 2.5 on p.64. For that reason people usually write *AUC* for 'Area Under (ROC) Curve', a convention I will follow.

Example 2.4 (Class imbalance). Suppose we feed the scoring tree in Figure 2.5 on p.62 an extended test set, with an additional batch of 50 negatives. The added negatives happen to be identical to the original ones, so the net effect is that the number of negatives in each leaf doubles. As a result the coverage curve changes (because the class ratio changes), but the ROC curve stays the same (Figure 2.8). Note that the AUC stays the same as well: while the classifier makes twice as many ranking errors, there are also twice as many positive–negative pairs, so the ranking error rate doesn't change.

Let us now consider an example of a coverage curve for a grading classifier. Figure 2.9 (left) shows a linear classifier (the decision boundary is denoted B) applied to a small data set of five positive and five negative examples, achieving an accuracy of 0.80. We can derive a score from this linear classifier by taking

the distance of an example from the decision boundary; if the example is on the negative side we take the negative distance. This means that the examples are ranked in the following order: p1 – p2 – p3 – n1 – p4 – n2 – n3 – p5 – n4 – n5. This ranking incurs four ranking errors: n1 before p4, and n1, n2 and n3 before p5. Figure 2.9 (right) visualises these four ranking errors in the top-left corner. The AUC of this ranking is 21/25 = 0.84.

Figure 2.9. (left) A linear classifier induces a ranking by taking the signed distance to the decision boundary as the score. This ranking only depends on the orientation of the decision boundary: the three lines result in exactly the same ranking. **(right)** The grid of correctly ranked positive–negative pairs (in green) and ranking errors (in red).

From this grid we obtain the coverage curve in Figure 2.10. Because of its stepwise character, this curve looks quite different from the coverage curves for scoring trees that we saw earlier in this section. The main reason is the absence of ties, which means that all segments in the curve are horizontal or vertical, and that there are as many segments as examples. We can generate this stepwise curve from the ranking as follows: starting in the lower left-hand corner, we go up one step if the next example in the ranking is positive, and right one step if the next example is negative. The result is a curve that goes three steps up (for p1–3), one step to the right (for n1), one step up (p4), two steps to the right (n2–3), one step up (p5), and finally two steps to the right (n4–5).

We can actually use the same procedure for grouping models if we handle ties as follows: in case of a tie between *p* positive examples and *n* negative examples, we go *p* steps up and *at the same time n* steps to the right. Looking back at Figure 2.7 on p.66, you will see that this is exactly what happens in the diagonal segments spanning the orange rectangles which arise as a result of the ties in the leaves of the decision tree. Thus, the principles underlying coverage and ROC curves are the same for both grouping and grading models, but the curves themselves look quite different in each case. *Grouping model ROC curves*

have as many line segments as there are instance space segments in the model; grading models have one line segment for each example in the data set. This is a concrete manifestation of something I mentioned in the Prologue: grading models have a much higher 'resolution' than grouping models; this is also called the model's *refinement.*

Notice the three points in Figure 2.10 labelled A, B and C. These points indicate the performance achieved by the decision boundaries with the same label in Figure 2.9. As an illustration, the middle boundary B misclassifies one out of five positives (*tpr* = 0.80) and one out of five negatives (*fpr* = 0.80). Boundary A doesn't misclassify any negatives, and boundary C correctly classifies all positives. In fact, while they should all have the same orientation, their exact location is not important, as long as boundary A is between p3 and n1, boundary B is between p4 and n2, and boundary C is between p5 and n4. There are good reasons why I chose exactly these three boundaries, as we shall see shortly. For the moment, observe what happens if we use all three boundaries to turn the linear model into a grouping model with four segments: the area above A, the region between A and B, the bit between B and C, and the rest below C. The result is that we no longer distinguish between n1 and p4, nor between n2–3 and p5. The ties just introduced change the coverage curve to the dotted segments in Figure 2.10. Notice that this results in a larger AUC of 0.90. Thus, *by decreasing a model's refinement we sometimes achieve better ranking performance*. Training a model is not just about amplifying significant distinctions, but also about diminishing the effect of misleading distinctions.

Turning rankers into classifiers I mentioned previously that the main difference between rankers and scoring classifiers is that a ranker only assumes that a higher score means stronger evidence for the positive class, but otherwise makes no assumptions about the scale on which scores are expressed, or what would be a good score threshold to separate positives from negatives. We will now consider the question how to obtain such a threshold from a coverage curve or ROC curve.

The key concept is that of the accuracy isometric. Recall that in a coverage plot points of equal accuracy are connected by lines with slope 1. All we need to do, therefore, is to draw a line with slope 1 through the top-left point (which is sometimes called *ROC heaven*) and slide it down until we touch the coverage

curve in one or more points. Each of those points achieves the highest accuracy possible with that model. In Figure 2.10 this method would identify points A and B as the points with highest accuracy (0.80). They achieve this in different ways: e.g., model A is more conservative on the positives.

Figure 2.10. The coverage curve of the linear classifier in Figure 2.9. The points labelled A, B and C indicate the classification performance of the corresponding decision boundaries. The dotted lines indicate the improvement that can be obtained by turning the grading classifier into a grouping classifier with four segments.

A similar procedure can be followed with ROC plots, as long as you keep in mind that all slopes have to be multiplied by the reciprocal of the class ratio, $1/clr = Neg/Pos$.

Example 2.5 (Tuning your spam filter). You have carefully trained your Bayesian spam filter, and all that remains is setting the decision threshold. You select a set of six spam and four ham e-mails and collect the scores assigned by the spam filter. Sorted on decreasing score these are 0.89 (spam), 0.80 (spam), 0.74 (ham), 0.71 (spam), 0.63 (spam), 0.49 (ham), 0.42 (spam), 0.32 (spam), 0.24 (ham), and 0.13 (ham). If the class ratio of 3 spam against 2 ham is representative, you can select the optimal point on the ROC curve using an isometric with slope 2/3. As can be seen in Figure 2.11, this leads to putting the decision boundary between the sixth spam e-mail and the third ham e-mail, and we can take the average of their scores as the decision threshold (0.28).

An alternative way of finding the optimal point is to iterate over all possible split points – from before the top ranked e-mail to after the bottom

one – and calculate the number of correctly classified examples at each split: 4 – 5 – 6 – 5 – 6 – 7 – 6 – 7 – 8 – 7 – 6. The maximum is achieved at the same split point, yielding an accuracy of 0.80. A useful trick to find out which accuracy an isometric in an ROC plot represents is to intersect the isometric with the descending diagonal. Since accuracy is a weighted average of the true positive and true negative rates, and since these are the same in a point on the descending diagonal, we can read off the corresponding accuracy value on the *y*-axis.

Figure 2.11. Selecting the optimal point on an ROC curve. The top dotted line is the accuracy isometric, with a slope of 2/3. The lower isometric doubles the value (or prevalence) of negatives, and allows a choice of thresholds. By intersecting the isometrics with the descending diagonal we can read off the achieved accuracy on the *y*-axis.

If the class distribution in the data is *not* representative, we can simply adjust the slope of the isometric. For example, if ham is in fact twice as prevalent, we use an isometric with slope 4/3. In the previous example this leads to three optimal points on the ROC curve.[10] Even if the class ratio in the data is representative, we may have other reasons to assign different weights to the classes. To illustrate, in the spam e-mail situation our spam filter may discard the false positives (ham e-mails misclassified as spam) so we may want to drive the false positive rate down by assigning a higher weight to the negatives (ham). This is often expressed as a *cost ratio* $c = c_{FN}/c_{FP}$ of the cost of false negatives in proportion to the cost of false positives, which in this case would be set to a value smaller than 1. The relevant isometrics then have a slope of $1/c$ in a coverage plot, and $1/(c \cdot clr)$ in an ROC plot. The combination of cost ratio and

class ratio gives a precise context in which the classifier is deployed and is referred to as the *operating condition*.

If the class or cost ratio is highly skewed, this procedure may result in a classifier that assigns the same class to all examples. For instance, if negatives are 1 000 times more prevalent than positives, accuracy isometrics are nearly vertical, leading to an unattainably high decision threshold and a classifier that classifies everything as negative. Conversely, if the profit of one true positive is 1 000 times the cost of a false positive, we would classify everything as positive – in fact, this is the very principle underlying spam e-mail! However, often such one-size-fits-all behaviour is unacceptable, indicating that accuracy is not the right thing to optimise here. In such cases we should use average recall isometrics instead. These run parallel to the ascending diagonal in both coverage and ROC plots, and help to achieve similar performance on both classes.

The procedure just described learns a decision threshold from labelled data by means of the ROC curve and the appropriate accuracy isometric. This procedure is often preferable over fixing a decision threshold in advance, particularly if scores are expressed on an arbitrary scale – for instance, this would provide a way to finetune the SpamAssassin decision threshold to our particular situation and preferences. Even if the scores are probabilities, as in the next section, these may not be sufficiently well estimated to warrant a fixed threshold of 0.5.

2.3 Class probability estimation A *class probability estimator* – or probability estimator in short – is a scoring classifier that outputs probability vectors over classes, i.e., a mapping $\hat{\mathbf{p}}: \mathscr{X} \rightarrow [0,1]^k$. We write $\hat{\mathbf{p}}(x) = (\hat{p}_1(x), \ldots, \hat{p}_k(x))$, where $\hat{p}_i(x)$ is the probability assigned to class C_i for instance x, and $\sum_{i=1}^{k} \hat{p}_i(x) = 1$. If we have only two classes, the probability associated with one class is 1 minus the probability of the other class; in that case, we use $\hat{p}(x)$ to denote the

estimated probability of the positive class for instance *x*. As with scoring classifiers, we usually do not have direct access to the true probabilities $p_i(x)$.

One way to understand the probabilities $\hat{p}_i(x)$ is as estimates of the probability $P_{\mathscr{C}}(c(x') = C_i | x' \sim x)$, where $x' \sim x$ stands for 'x' is similar to x'. In other words, how frequent are instances of this class among instances similar to x? The intuition is that the more (or less) frequent they are, the more (or less) confident we should be in our belief that x belongs to that class as well. What we mean with similarity in this context will depend on the models we are considering – we will illustrate it here by means of a few two-class examples. First, assume a situation in which any two instances are similar to each other. We then have $P_{\mathscr{C}}(c(x') = \oplus | x' \sim x) = P_{\mathscr{C}}(c(x') = \oplus)$ which is simply estimated by the proportion *pos* of positives in our data set (I am going to drop the subscript $\mathscr{C}$ from now on). In other words, in this scenario we predict $\hat{p}(x) = pos$ regardless of whether we know anything about x's true class. At the other extreme, consider a situation in which no two instances are similar unless they are the same, i.e., $x' \sim x$ if $x' = x$, and $x' \neq x$ otherwise. In this case we have $P(c(x') = \oplus | x' \sim x) = P(c(x) = \oplus)$, which – because x is fixed – is 1 if $c(x) = \oplus$ and 0 otherwise. Put differently, we predict $\hat{p}(x) = 1$ for all known positives and $\hat{p}(x) = 0$ for all known negatives, but we can't generalise this to unseen instances.

Figure 2.12. A probability estimation tree derived from the feature tree in Figure 1.4.

A feature tree allows us to strike a balance between these extreme and simplistic scenarios, using the similarity relation $\sim_T$ associated with feature tree

T: $x' \sim_T x$ if, and only if, x and x' are assigned to the same leaf of the tree. In each leaf we then predict the proportion of positives assigned to that leaf. For example, in the right-most leaf in Figure 1.4 on p.32 the proportion of positives is 40/50 = 0.80, and thus we predict $\hat{p}(x)$ = 0.80 for all instances x assigned to that leaf; similarly for the other two leaves (Figure 2.12). If we threshold $\hat{p}(x)$ at 0.5 (i.e., predict spam if the spam probability is 0.5 or more and predict ham otherwise), we get the same classifier as obtained by predicting the majority class in each leaf of the feature tree.

Assessing class probability estimates As with classifiers, we can now ask the question of how good these class probability estimators are. A slight complication here is that, as already remarked, we do not have access to the true probabilities. One trick that is often applied is to define a binary vector $(I[c(x) = C_1], \ldots, I[c(x) = C_k])$, *which has the i-th bit set to 1 if x's true class is* C_i *and all other bits set to 0, and use these as the 'true' probabilities. We can then define the squared error (SE) of the predicted probability vector* $\hat{\mathbf{p}}(x) = (\hat{p}_1(x), \ldots, \hat{p}_k(x))$ *as*

$$\mathrm{SE}(x) = \frac{1}{2}\sum_{i=1}^{k}(\hat{p}_i(x) - I[c(x) = C_i])^2 \qquad (2.6)$$

and the *mean squared error* (*MSE*) as the average squared error over all instances in the test set:

$$\mathrm{MSE}(Te) = \frac{1}{|Te|}\sum_{x \in Te}\mathrm{SE}(x) \qquad (2.7)$$

This definition of error in probability estimates is often used in *forecasting theory* where it is called the *Brier score*. The factor 1/2 in Equation 2.6 ensures that the squared error per example is normalised between 0 and 1: the worst possible situation is that the wrong class is predicted with probability 1, which means two 'bits' are wrong. For two classes this reduces to a single term $(\hat{p}(x) - I[c(x) = \oplus])^2$ only referring to the positive class. Notice that, if a class probability estimator is 'categorical' – i.e., it assigns probability 1 to one class and probability 0 to the rest – it is effectively a classifier, and MSE reduces to accuracy as defined in Section 2.1.

Example 2.6 (Squared error). Suppose one model predicts (0.70, 0.10,

0.20) for a particular example x in a three-class task, while another appears much more certain by predicting (0.99, 0, 0.01). If the first class is the actual class, the second prediction is clearly better than the first: the SE of the first prediction is $((0.70 - 1)^2 + (0.10 - 0)^2 + (0.20 - 0)^2)/2 = 0.07$, while for the second prediction it is $((0.99 - 1)^2 + (0-0)^2 + (0.01 - 0)^2)/2 = 0.0001$. The first model gets punished more because, although mostly right, it isn't quite sure of it.

However, if the third class is the actual class, the situation is reversed: now the SE of the first prediction is $((0.70 - 0)^2 + (0.10 - 0)^2 + (0.20 - 1)^2)/2 = 0.57$, and of the second $((0.99 - 0)^2 + (0 - 0)^2 + (0.01 - 1)^2)/2 = 0.98$. The second model gets punished more for not just being wrong, but being presumptuous.

Returning to the probability estimation tree in Figure 2.12, we calculate the squared error per leaf as follows (left to right):

$$SE_1 = 20(0.33-1)^2 + 40(0.33-0)^2 = 13.33$$
$$SE_2 = 10(0.67-1)^2 + 5(0.67-0)^2 = 3.33$$
$$SE_3 = 20(0.80-1)^2 + 5(0.80-0)^2 = 4.00$$

which leads to a mean squared error of $MSE = \frac{1}{100}(SE_1 + SE_2 + SE_3) = 0.21$. An interesting question is whether we can change the predicted probabilities in each leaf to obtain a lower mean squared error. It turns out that this is not possible: predicting probabilities obtained from the class distributions in each leaf is optimal in the sense of lowest MSE. For instance, changing the predicted probabilities in the left-most leaf to 0.40 for spam and 0.60 for ham, or 0.20 for spam and 0.80 for ham, results in a higher squared error:

$$SE_1' = 20(0.40-1)^2 + 40(0.40-0)^2 = 13.6$$
$$SE_1'' = 20(0.20-1)^2 + 40(0.20-0)^2 = 14.4$$

The reason for this becomes obvious if we rewrite the expression for two-class squared error of a leaf as follows, using the notation $n^{\oplus}$ and $n^{\ominus}$ for the numbers of positive and negative examples in the leaf:

$$n^{\oplus}(\hat{p}-1)^2 + n^{\ominus}\hat{p}^2 = (n^{\oplus}+n^{\ominus})\hat{p}^2 - 2n^{\oplus}\hat{p} + n^{\oplus} = (n^{\oplus}+n^{\ominus})\left[\hat{p}^2 - 2\dot{p}\hat{p} + \dot{p}\right]$$
$$= (n^{\oplus}+n^{\ominus})\left[(\hat{p}-\dot{p})^2 + \dot{p}(1-\dot{p})\right]$$

where $\dot{p} = n^{\oplus}/(n^{\oplus}+n^{\ominus})$ is the relative frequency of the positive class among the examples covered by the leaf, also called the *empirical probability*. As the term $\dot{p}(1-\dot{p})$ does not depend on the predicted probability $\hat{p}$, we see immediately that

we achieve lowest squared error in the leaf if we assign $\hat{p} = \dot{p}$.

Empirical probabilities are important as they allow us to obtain or finetune probability estimates from classifiers or rankers. If we have a set S of labelled examples, and the number of examples in S of class C_i is denoted n_i, then the empirical probability vector associated with S is $\dot{\mathbf{p}}(S) = (n_1/|S|, \ldots, n_k/|S|)$. In practice, it is almost always a good idea to *smooth* these relative frequencies to avoid issues with extreme values (0 or 1). The most common way to do this is to set

$$\dot{p}_l(S) = \frac{n_l + 1}{|S| + k} \qquad (2.8)$$

This is called the *Laplace correction*, after the French mathematician Pierre-Simon Laplace, who introduced it for the case $k = 2$ (also known as Laplace's rule of succession). In effect, we are adding uniformly distributed *pseudo-counts* to each of the k alternatives, reflecting our prior belief that the empirical probabilities will turn out uniform.[11] We can also apply non-uniform smoothing by setting

$$\dot{p}_l(S) = \frac{n_l + m \cdot \pi_l}{|S| + m} \qquad (2.9)$$

This smoothing technique, known as the *m-estimate*, allows the choice of the number of pseudo-counts m as well as the prior probabilities π_i. The Laplace correction is a special case of the m-estimate with $m = k$ and $\pi_i = 1/k$.

If all elements of S receive the same predicted probability vector $\hat{\mathbf{p}}(S)$ – which happens if S is a segment of a grouping model – then a similar derivation to the one above allows us to write the total incurred squared error over S in terms of estimated and empirical probabilities as

$$\begin{aligned}\mathrm{SE}(S) = \sum_{x \in S} \mathrm{SE}(x) &= \sum_{x \in S} \frac{1}{2} \sum_{l=1}^{k} (\hat{p}_l(x) - I[c(x) = C_l])^2 \\ &= \frac{1}{2}|S| \sum_{l=1}^{k} (\hat{p}_l(S) - \dot{p}_l(S))^2 + \frac{1}{2}|S| \sum_{l=1}^{k} (\dot{p}_l(S)(1 - \dot{p}_l(S))\end{aligned}$$

The first term of the final expression is called the *calibration loss*, and measures squared error with respect to the empirical probabilities. It can be reduced to 0 in grouping models where we are free to choose the predicted probabilities for each segment, as in probability estimation trees. Models with low calibration loss are said to be wellcalibrated. The second term is called the *refinement loss*; this depends only on the empirical probabilities, and is smaller if they are less uniform.

This analysis suggests that the best way of obtaining probability estimates is from empirical probabilities, obtained from the training set or from another set of labelled examples specifically set aside for the purpose. However, there are two issues we need to consider here. The first is that with some models we must make sure that the predicted probabilities obey the ranking imposed by the

model. The second is that with grading models we don't have immediate access to empirical probabilities, since each example tends to get assigned an equivalence class of its own. We will now discuss this in a bit more detail.

Turning rankers into class probability estimators Consider again Example 2.5 on p.70, and imagine the scores are not probabilities but on some unknown scale, so that the spam filter is a ranker rather than a class probability estimator. Since each test example receives a different score, the 'empirical probabilities' are either 0 (for negative examples) or 1 (for positive examples), leading to a sequence of $\dot{p}$-values of 1 – 1 – 0 – 1 – 1 – 0 – 1 – 1 – 0 – 0 in order of decreasing scores. The obvious problem is that these $\dot{p}$-values do not obey the order imposed by the scores, and so cannot be used directly to obtain probability estimates. Smoothing the empirical probabilities using Laplace correction doesn't really address this problem, since all it does is replace 0 with 1/3 and 1 with 2/3. We need a different idea.

Looking at Figure 2.11, we see that $\dot{p}$ = 1 corresponds to a vertical segment of the ROC curve, and $\dot{p}$ = 0 to a horizontal segment. The problem we have is caused by having a vertical segment following a horizontal one, or, more generally, a segment with steeper slope following a flatter segment. We will call a sequence of segments with increasing slope a *concavity*, as it forms a 'dent' in the ROC curve. A curve without concavities is a *convex* ROC curve. Our curve has two concavities: one formed by the third, fourth and fifth example, and the other by the sixth, seventh and eighth example. Suppose now that the third to the fifth example all receive the same score, say 0.7; and the sixth to the eight example are also tied, say at 0.4. In that case the ROC curve would have six segments, with empirical probabilities 1 – 1 – 2/3 – 2/3 – 0 – 0. As we see, the $\dot{p}$ -values are now decreasing with the scores; in other words, the concavities have disappeared and the ROC curve has become convex.

Figure 2.13. (left) The solid red line is the convex hull of the dotted ROC curve. **(right)** The corresponding calibration map in red: the plateaus correspond to several examples being mapped to the same segment of the convex hull, and linear interpolation between example scores occurs when we transition from one convex hull segment to the next. A Laplace-corrected calibration map is indicated by the dashed line in blue: Laplace smoothing compresses the range of calibrated probabilities but can sometimes affect the ranking.

More generally speaking, *concavities in ROC curves can be remedied by combining segments through tied scores*. This is achieved by identifying what are sometimes called *adjacent violators*. For instance, in the sequence 1 – 1 – 0 – 1 – 1 – 0 – 1 – 1 – 0 – 0, the third and fourth example are adjacent violators, because they violate the rule that scores should be decreasing from left to right in the sequence (or, mathematically more accurate, they should be non-increasing). This is remedied by assigning them both their average score, leading to the sequence 1 – 1 – [1/2 – 1/2] – 1 – 0 – 1 – 1 – 0 – 0. The newly introduced segment now forms an adjacent violator pair with the fourth example, so we give them all their mean score, leading to the sequence 1 – 1 – [2/3 – 2/3 – 2/3] – 0 – 1 – 1 – 0 – 0.[12] The second 0 – 1 – 1 concavity is treated identically, and the final sequence is 1 – 1 – [2/3 – 2/3 – 2/3] – [2/3 – 2/3 – 2/3] – 0 – 0.

The result is illustrated in Figure 2.13. On the left, we see how the two concavities are replaced with two diagonal line segments with the same slope. These diagonal segments coincide with the accuracy isometric that gives the three 'outermost' points involved in the concavities the same accuracy (Figure 2.11 on p.71). Jointly, the red segments constitute the *convex hull* of the ROC curve, which is the unique convex curve through the outermost points of the original ROC curve. The convex hull has a higher AUC than the original curve, because it replaces (some of) the ranking errors of the original curve with half-errors due to ties. In our example the original ranking incurs 6 out of 24 ranking errors (AUC = 0.75), while the convex hull turns all of these into half-errors (AUC = 0.83).

Once we have determined the convex hull, we can use the empirical probabilities in each segment of the convex hull as calibrated probabilities. Figure 2.13 (right) shows the resulting *calibration map*, which is a piecewise linear, non-decreasing curve mapping original scores on the x-axis to calibrated probabilities on the y-axis. Also shown is an alternative calibration map giving probability estimates after Laplace correction: for the given sequence these are 2/3 – 2/3 – [3/5 – 3/5 – 3/5] – [3/5 – 3/5 – 3/5] – 1/3 – 1/3, giving rise to a much compressed range of probability estimates.

Let's now look at this process from the point of view of mean squared error, calibration and refinement. The original scores had a mean squared error of $\frac{1}{10}[(0.89-1)^2+(0.80-1)^2+(0.74-0)^2+(0.71-1)^2+(0.63-1)^2+(0.49-0)^2+(0.42-1)^2+(0.32-1)^2+(0.24-0)^2+(0.13-0)^2]=0.19$. Notice that this is entirely incurred by the calibration loss, as all empirical probabilities are either 0 or 1 and thus the refinement loss is zero. The calibrated scores have a mean squared error of $\frac{1}{10}[(1-1)^2+(1-1)^2+(0.67-0)^2+(0.67-1)^2+(0.67-1)^2+(0.67-0)^2+(0.67-1)^2+(0.67-1)^2+(0-0)^2+(0-0)^2]=0.13$. Now the entire mean squared error is incurred by refinement loss as the estimated probabilities are equal to the empirical ones in each segment by construction. We have traded an increase in refinement loss for a decrease in calibration loss; since the latter is larger than the former, the overall error decreases. The increase in refinement loss comes from the construction of the convex hull, which introduces diagonal segments. The technical term for this process of obtaining calibrated scores through the convex hull of the ROC curve is *isotonic calibration*, as the mathematical problem underlying it is called isotonic regression. Some caution is in order when applying isotonic calibration, as it is easy to overfit the data in this process. In the calibration map in Figure 2.13 (right), both the horizontal transition points and the vertical levels are directly obtained from the given data, and may not generalise well to unseen data. This is why it is advisable to apply the Laplace correction to the empirical probabilities, even though it will increase the calibration loss on the given data.

2.4 Binary classification and related tasks: Summary and further reading In this

chapter we have looked at binary classification, a ubiquitous task that forms the starting point of a lot of work in machine learning. Although we haven't talked much about learning in this chapter, my philosophy is that you will reach a better understanding of machine learning models and algorithms if you first study the tasks that these models are meant to address.

- In Section 2.1 we defined the binary classification task and introduced an important tool to assess performance at such a task, namely the two-by-tw contingency table. A wide range of performance indicators are derived fro the counts in a contingency table. I introduced the coverage plot, which visualises a contingency table as a rectangle with size *Pos* up and size *Neg* across, and within that rectangle a point with *y*-coordinate *TP* and *x*-coordinate *FP*. We can visualise several models evaluated on the same dat set by several points, and use the fact that accuracy is constant along line segments with slope 1 to visually rank these classifiers on accuracy. Alternatively, we can normalise the rectangle to be a unit square with true and false positive rate on the axes. In this so-called ROC space, line segments with slope 1 (i.e., those parallel to the ascending diagonal) connect points with the same average recall (sometimes also called macro accuracy). The use of these kinds of plot in machine learning was pioneer by Provost and Fawcett (2001). Unnormalised coverage plots were introduced by Fürnkranz and Flach (2003).

- Section 2.2 considered the more general task of calculating a score for eac example (or a vector of scores in the general case of more than two classes While the scale on which scores are expressed is unspecified, it is customary to put the decision threshold at $\hat{s}(x) = 0$ and let the sign of the score stand for the prediction (positive or negative). Multiplying the score with the true class gives us the margin, which is positive for a correct

prediction and negative for an incorrect one. A loss function determines how much negative margins are penalised and positive margins rewarded. The advantage of working with convex and continuously differentiable 'surrogate' loss functions (rather than with 0–1 loss, which is the loss function we ultimately want to optimise) is that this often leads to more tractable optimisation problems.

☞ Alternatively, we can ignore the scale on which scores are measured altogether and only work with their order. Such a ranker is visualised in coverage or ROC space by a piecewise continuous curve. For grouping models the line segments in these curves correspond to instance space segments (e.g., the leaves of a tree model) whereas for grading models there is a segment for each unique score assigned by the model. The area under the ROC curve gives the ranking accuracy (an estimate of the probability that a random positive is ranked before a random negative) anc is known in statistics as the Wilcoxon-Mann-Whitney statistic These curve can be used to find a suitable operating point by translating the operating condition (class and cost distribution) into an isometric in ROC or coverag space. The origins of ROC curves are in signal detection theory (Egan, 1975); accessible introductions can be found in (Fawcett, 2006; Flach, 2010*b*).

☞ In Section 2.3 we looked at scoring models whose scores can be interprete as estimates of the probability that the instance belongs to a particular clas Such models were pioneered in forecasting theory by Brier (1950) and Murphy and Winkler (1984), among others. We can assess the quality of class probability estimates by comparing them to the 'ideal' probabilities (for a positive, 0 for a negative) and taking mean squared error. Since there is no reason why the true probabilities should be categorical this is quite a crude assessment, and decomposing it into calibration loss and refinement loss provides useful additional information. We have also seen a very usef trick for smoothing relative frequency estimates of probabilities by adding pseudo-counts, either uniformly distributed (Laplace correction) or according to a chosen prior (*m*-estimate). Finally, we have seen how we ca use the ROC convex hull to obtain calibrated class probability estimates. The approach has its roots in isotonic regression (Best and Chakravarti, 1990) and was introduced to the machine learning community by Zadrozn and Elkan (2002). Fawcett and Niculescu-Mizil (2007) and Flach and Matsubara (2007) show that the approach is equivalent to calibration by

means of the ROC convex hull. (Note that in this chapter we have seen tw different uses of the term 'convex': one in relation to loss functions, where convexity means that linear interpolation between any two points on the curve depicting the loss function will never result in a point below the curve; and the other in relation to the ROC convex hull, where it refers to the linearly interpolated boundary of a convex set which envelopes all points in the set.)

---

[1]It is perhaps worth emphasising that an instance space like this is an unimaginably vast set (e.g., the set of all possible text messages of 160 characters using only lower-case letters, spaces and full stops is 28^{160},a number too large for most pocket calculators), and that only a minuscule fraction of this set carries enough meaning to be possibly encountered in the real world.

[2]If you are keen to know how such a decision tree can be learned from data, you may want to take a sneak preview at Algorithm 5.1 on p.132.

[3]An internet search engine can be seen as a binary classifier into the classes relevant and irrelevant, or interesting and not interesting, if we fix the query – not very realistic in practice, but a useful analogy for our purposes.

[4]More generally, a k-class contingency table has $(k + 1)^2$ entries and k^2 degrees of freedom.

[5]This terminology comes from the field of *multi-criterion optimisation*. A dominated solution is one that is not on the *Pareto front*.

[6]Remember that recall is just a different name for true positive rate; negative recall is then the same as the true negative rate, and average recall is the average of positive recall (or true positive rate) and negative recall (or true negative rate). It is sometimes called *macro-averaged accuracy*.

[7]ROC stands for *receiver operating characteristic*, a term originating from *signal detection theory*.

[8]Remember that in Chapter 1 we talked about the margin of a classifier as the distance between the decision boundary and the nearest example. Here we use margin in a slightly more general sense: each example has a margin, not just the nearest one. This will be further explained in Section 7.3.

[9]A total order with ties should not be confused with a partial order (see Background 2.1 on p.51). In a total order with ties (which is really a total order on equivalence classes), any two elements are comparable, either in one direction or in both. In a partial order some elements are incomparable.

[10]It seems reasonable to choose the middle of these three points, leading to a threshold of 0.56. An alternative is to treat all e-mails receiving a score in the interval [0.28,0.77] as lying on the decision boundary, and to randomly assign a class to those e-mails.

[11]This can be modelled mathematically by a prior probability distribution known as a *Dirichlet prior*.

[12]These two steps can be combined into one: once a pair of adjacent violators is found, we can scan to the left and right to include examples with the same score as the left and right example in the pair,

respectively.

CHAPTER 3

Beyond binary classification

THE PREVIOUS CHAPTER introduced binary classification and associated tasks such as ranking and class probability estimation. In this chapter we will go beyond these basic tasks in a number of ways. Section 3.1 discusses how to handle more than two classes. In Section 3.2 we consider the case of a real-valued target variable. Section 3.3 is devoted to various forms of learning that are either unsupervised or aimed at learning descriptive models.

3.1 Handling more than two classes

Certain concepts are fundamentally binary. For instance, the notion of a coverage curve does not easily generalise to more than two classes. We will now consider general issues related to having more than two classes in classification, scoring and class probability estimation. The discussion will address two issues: how to evaluate multi-class performance, and how to build multi-class models out of binary models. The latter is necessary for some models, such as linear classifiers, that are primarily designed to separate two classes. Other models, including decision trees, handle any number of classes quite naturally.

Multi-class classification

Classification tasks with more than two classes are very common. For instance, once a patient has been diagnosed as suffering from a rheumatic disease, the doctor will want to classify him or her further into one of several variants. If we have k classes, performance of a classifier can be assessed using a k-by-k contingency table. Assessing performance is easy if we are interested in the classifier's accuracy, which is still the sum of the descending diagonal of the contingency table, divided by the number of test instances. However, as before, this can obscure differences in performance on different classes, and other quantities may be more meaningful.

Example 3.1 (Performance of multi-class classifiers). Consider the following three-class confusion matrix (plus marginals):

	Predicted			
	15	2	3	20
Actual	7	15	8	30
	2	3	45	50
	24	20	56	100

The accuracy of this classifier is (15 + 15 + 45)/100 = 0.75. We can calculate per-class precision and recall: for the first class this is 15/24 = 0.63 and 15/20 = 0.75 respectively, for the second class 15/20 = 0.75 and 15/30 = 0.50, and for the third class 45/56 = 0.80 and 45/50 = 0.90. We could average these numbers to obtain single precision and recall numbers for the whole classifier, or we could take a weighted average taking the proportion of each class into account. For instance, the weighted average precision is 0.20·0.63 + 0.30·0.75 + 0.50·0.80 = 0.75. Notice that we still have that accuracy is weighted average per-class recall, as in the two-class case (see Example 2.1 on p.56).

Another possibility is to perform a more detailed analysis by looking at precision and recall numbers for each pair of classes: for instance, when distinguishing the first class from the third precision is 15/17 = 0.88 and recall is 15/18 = 0.83, while distinguishing the third class from the first these numbers are 45/48 = 0.94 and 45/47 = 0.96 (can you explain why these numbers are much higher in the latter direction?).

Imagine now that we want to construct a multi-class classifier, but we only have the ability to train two-class models – say linear classifiers. There are various ways to combine several of them into a single k-class classifier. The *one-versus-rest* scheme is to train k binary classifiers, the first of which separates class C_1 from C_2, …, C_n, the second of which separates C_2 from all other classes, and so on. When training the i-th classifier we treat all instances of class C_i as positive examples, and the remaining instances as negative examples.

Sometimes the classes are learned in a fixed order, in which case we learn $k - 1$ models, the i-th one separating C_i from $C_{i+1}, \ldots, C_n$ with $l \leq i < n$. An alternative to one-versus-rest is *one-versus-one*. In this scheme, we train $k(k - 1)/2$ binary classifiers, one for each pair of different classes. If a binary classifier treats the classes asymmetrically, as happens with certain models, it makes more sense to train two classifiers for each pair, leading to a total of $k(k - 1)$ classifiers.

A convenient way to describe all these and other schemes to decompose a k-class task into l binary classification tasks is by means of a so-called *output code* matrix. This is a k-by-l matrix whose entries are +1, 0 or –1. The following are output codes describing the two ways to transform a three-class task by means of one-versus-one:

$$\begin{pmatrix} +1 & +1 & 0 \\ -1 & 0 & +1 \\ 0 & -1 & -1 \end{pmatrix} \qquad \begin{pmatrix} +1 & -1 & +1 & -1 & 0 & 0 \\ -1 & +1 & 0 & 0 & +1 & -1 \\ 0 & 0 & -1 & +1 & -1 & +1 \end{pmatrix}$$

Each column of these matrices describes a binary classification task, using the class corresponding to the row with the +1 entry as positive class and the class with the –1 entry as the negative class. So, in the symmetric scheme on the left, we train three classifiers: one to distinguish between C_1 (positive) and C_2 (negative), one to distinguish between C_1 (positive) and C_3 (negative), and the remaining one to distinguish between C_2 (positive) and C_3 (negative). The asymmetric scheme on the right learns three more classifiers with the roles of positives and negatives swapped. The code matrices for the unordered and ordered version of the one-versus-rest scheme are as follows:

$$\begin{pmatrix} +1 & -1 & -1 \\ -1 & +1 & -1 \\ -1 & -1 & +1 \end{pmatrix} \qquad \begin{pmatrix} +1 & 0 \\ -1 & +1 \\ -1 & -1 \end{pmatrix}$$

On the left, we learn one classifier to distinguish C_1 (positive) from C_2 and C_3 (negative), another one to distinguish C_2 (positive) from C_1 and C_3 (negative), and the third one to distinguish C_3 (positive) from C_1 and C_2 (negative). On the right, we have ordered the classes in the order $C_1 - C_2 - C_3$, and thus only two classifiers are needed.

In order to decide the class for a new test instance, we collect predictions from

all binary classifiers which can again be +1 for positive, –1 for negative and 0 for no prediction or *reject* (the latter is possible, for instance, with a rule-based classifier). Together, these predictions form a 'word' that can be looked up in the code matrix, a process also known as *decoding*. Suppose the word is –1 +1 –1 and the scheme is unordered one-versus-rest, then we know the decision should be class C_2. The question is: what should we with words that do not appear in the code matrix? For instance, suppose the word is 0 + 1 0, and the scheme is symmetric one-versus-one (the first of the above four code matrices). In this case we could argue that the nearest code word is the first row in the matrix, and so we should predict C_1. To make this a little bit more precise, we define the distance between a word w and a code word c as $d(w, c) = \sum_i (1 - w_i c_i)/2$, where i ranges over the 'bits' of the words (the columns in the code matrix). That is, bits where the two words agree do not contribute to the distance; each bit where one word has +1 and the other –1 contributes 1; and if one of the bits is 0 the contribution is 1/2, regardless of the other bit.[1] The predicted class for word w is then $\text{argmin}_j\, d(w, c_j)$, where c_j is the j-th row of the code matrix. So, if w = 0 + 1 0 then $d(w, c_1) = 1$ and $d(w, c_2) = d(w, c_3) = 1.5$, which means that we predict C_1.

However, the nearest code word is not always unique. For instance, suppose we use a four-class one-versus-rest scheme, and two of the binary classifiers predict positive and the other two negative, then this word is equidistant to two code words, and so we can't resolve which of the two classes corresponding to the two nearest code words to predict. We can improve the situation by adding more columns to our code matrix:

$$\begin{pmatrix} +1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 \\ -1 & -1 & +1 & -1 \\ -1 & -1 & -1 & +1 \end{pmatrix} \qquad \begin{pmatrix} +1 & -1 & -1 & -1 & +1 & +1 & +1 \\ -1 & +1 & -1 & -1 & +1 & -1 & -1 \\ -1 & -1 & +1 & -1 & -1 & +1 & -1 \\ -1 & -1 & -1 & +1 & -1 & -1 & +1 \end{pmatrix}$$

On the left we see a standard four-class one-versus-rest code matrix, which has been extended with three extra columns (i.e., binary learning problems) on the right. As a result, the distance between any two code words has now increased from 2 to 4, increasing the likelihood that we can decode words that are not contained in the code matrix. The resulting scheme can be seen as a mix between one-versus-rest and one-versus-one classification. However, notice that the additional binary learning problems may be hard. For instance, if our four

classes are spam e-mails, work e-mails, household e-mails (e.g., utility bills or credit card statements) and private e-mails, then each one-versus-rest binary classification task may be much easier than, say, distinguishing between spam and work e-mails on the one hand and household and private e-mails on the other.

The one-versus-rest and one-versus-one schemes are the most commonly used ways to turn binary classifiers into multi-class classifiers. In order to force a decision in the one-versus-rest scenario we can settle on a class ordering prior to or after learning. In the one-versus-one scheme we can use voting to arrive at a decision, which is actually equivalent to distance-based decoding as demonstrated by the following example.

Example 3.2 (One-versus-one voting). A one-versus-one code matrix for $k = 4$ classes is as follows:

$$\begin{pmatrix} +1 & +1 & +1 & 0 & 0 & 0 \\ -1 & 0 & 0 & +1 & +1 & 0 \\ 0 & -1 & 0 & -1 & 0 & +1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{pmatrix}$$

Suppose our six pairwise classifiers predict $w = +1\ -1\ +1\ -1\ +1\ +1$. We can interpret this as votes for $C_1 - C_3 - C_1 - C_3 - C_2 - C_3$; i.e., three votes for C_3, two votes for C_1 and one vote for C_2. More generally, the i -th classifier's vote for the j -th class can be expressed as $(1 + w_i c_{ji})/2$, where C_{ji} is the entry in the j -throw and i -th column of the code matrix. However, this overcounts the 0 entries in the code matrix; since every class participates in $k - 1$ pairwise binary tasks, and there are $l = k\,(k - 1)/2$ tasks, the number of zeros in every row is $k\,(k - 1)/2 - (k - 1) = (k - 1)(k - 2)/2 = l(k - 2)/k$ (3 in our case). For each zero we need to subtract half a vote, so the number of votes for C_j is

$$v_j = \left(\sum_{i=1}^{l} \frac{1 + w_i c_{ji}}{2}\right) - l\frac{k-2}{2k} = \left(\sum_{i=1}^{l} \frac{w_i c_{ji} - 1}{2}\right) + l - l\frac{k-2}{2k}$$
$$= -d_j + l\frac{2k - k + 2}{2k} = \frac{(k-1)(k+2)}{4} - d_j$$

where $d_j = \sum_i (1 - w_i c_{ji})/2$ is the bit-wise distance we used earlier. In other words, the distance and number of votes for each class sum to a constant depending only on the number of classes; with three classes this is 4.5. This can be checked by noting that the distance between w and the first code word is 2.5 (two votes for C_1); with the second code word, 3.5 (one vote for C_2); with the third code word, 1.5 (three votes for C_3); and 4.5 with the fourth code word (no votes).

If our binary classifiers output scores, we can take these into account as follows. As before we assume that the sign of the scores s_i indicates the class. We can then use the appropriate entry in the code matrix c_{ji} to calculate a margin $z_i = s_i c_{ji}$, which we feed into a loss function L (margins and loss functions were discussed in Section 2.2). We thus define the distance between a vector of scores s and the j-th code word c_j as d (s, $d(s, c_j) = \sum_i L(s_i c_{ji})$, and we assign the class which minimises this distance. This way of arriving at a multi-class decision from binary scores is called *loss-based deCoding*.

Example 3.3 (Loss-based decoding). Continuing the previous example, suppose the scores of the six pairwise classifiers are (+5,–0.5, +4,–0.5, +4, +0.5). This leads to the following margins, in matrix form:

$$\begin{pmatrix} +5 & -0.5 & +4 & 0 & 0 & 0 \\ -5 & 0 & 0 & -0.5 & +4 & 0 \\ 0 & +0.5 & 0 & +0.5 & 0 & +0.5 \\ 0 & 0 & -4 & 0 & -4 & -0.5 \end{pmatrix}$$

Using 0–1 loss we ignore the magnitude of the margins and thus predict C_3 as in the voting-based scheme of Example 3.2. Using exponential loss $L(z) = \exp(-z)$, we obtain the distances (4.67,153.08,4.82,113.85). Loss-based decoding would therefore (just) favour C_1, by virtue of its strong wins against C_2 and C_4; in contrast, all three wins of C_3 are with small margin.

It should be noted that loss-based decoding assumes that each binary classifier scores on the same scale.

Multi-class scores and probabilities

If we want to calculate multi-class scores and probabilities from binary classifiers, we have a number of different options.

- We can use the distances obtained by loss-based decoding and turn them into scores by means of some appropriate transformation, just as we turne bit-wise distances into votes in Example 3.2. This method is applicable if the binary classifiers output calibrated scores on a single scale.
- Alternatively, we can use the output of each binary classifier as features (real-valued if we use the scores, binary if we only use the predicted class and train a model that can produce multi-class scores, such as naive Bayes or tree models. This method is generally applicable but requires additional training.
- A simple alternative that is also generally applicable and often produces satisfactory results is to derive scores from *coverage counts*: the number o examples of each class that are classified as positive by the binary classife Example 3.4 illustrates this.

Example 3.4 (Coverage counts as scores). Suppose we have three classes and three binary classifiers which either predict positive or negative (there is no reject option). The first classifier classifies 8 examples of the first class as positive, no examples of the second class, and 2 examples of the third class. For the second classifier these counts are 2, 17 and 1, and for the third they are 4, 2 and 8. Suppose a test instance is predicted as positive by the first and third classifiers. We can add the coverage counts of these two classifiers to obtain a score vector of (12, 2, 10). Likewise, if all three classifiers 'fire' for a particular test instance (i.e., predict positive), the score vector is (14, 19, 11).

We can describe this scheme conveniently using matrix notation:

$$\begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 8 & 0 & 2 \\ 2 & 17 & 1 \\ 4 & 2 & 8 \end{pmatrix} = \begin{pmatrix} 12 & 2 & 10 \\ 14 & 19 & 11 \end{pmatrix} \tag{3.1}$$

The middle matrix contains the class counts (one row for each classifier). The left 2-by-3 matrix contains, for each example, a row indicating which classifiers fire for that example. The right-hand side then gives the combined counts for each example.

With l binary classifiers, this scheme divides the instance space into up to 2^l regions. Each of these regions is assigned its own score vector, so in order to obtain diverse scores l should be reasonably large.

Once we have multi-class scores, we can ask the familiar question of how good these are. As we have seen in Section 2.1, an important performance index of a binary scoring classifier is the area under the ROC curve or AUC, which is the proportion of correctly ranked positive–negative pairs. Unfortunately ranking does not have a direct multi-class analogue, and so the most obvious thing to do is to calculate the average AUC over binary classification tasks, either in a one-versus-rest or one-versus-one fashion. For instance, the one-versus-rest average AUC estimates the probability that, taking a uniformly drawn class as positive, a uniformly drawn example from that class gets a higher score than a uniformly drawn example over all other classes. Notice that the 'negative' is more likely to come from the more prevalent classes;for that reason the positive class is sometimes also drawn from a non-uniform distribution in which each class is weighted with its prevalence in the test set.

Example 3.5 (Multi-class AUC). Assume we have a multi-class scoring classifier that produces a k-vector of scores $\hat{\mathbf{s}}(x) = (\hat{s}_1(x), \ldots, \hat{s}_k(x))$ for each test instance x. By restricting attention to $\hat{s}_i(x)$ we obtain a scoring classifier for class C_i against the other classes, and we can calculate the one-versus-rest AUC for C_i in the normal way.

By way of example, suppose we have three classes, and the one-versus-rest AUCs come out as 1 for the first class, 0.8 for the second class and 0.6 for the third class. Thus, for instance, all instances of class 1 receive a higher first entry in their score vectors than any of the instances of the other

two classes. The average of these three AUCs is 0.8, which reflects the fact that, if we uniformly choose an index i, and we select an instance x uniformly among class C_i and another instance x' uniformly among all instances not from C_i, then the expectation that $\hat{s}_i(x) > \hat{s}_i(x')$ is 0.8.

Suppose now C_1 has 10 instances, C_2 has 20 and C_3 70. The weighted average of the one-versus-rest AUCs is then 0.68: that is, if we uniformly choose x *without reference to the class*, and then choose x' uniformly from among all instances not of the same class as x', the expectation that $\hat{s}_i(x) > \hat{s}_i(x')$ is 0.68. This is lower than before, because it is now more likely that a random x comes from class C_3, whose scores do a worse ranking job.

We can obtain similar averages from one-versus-one AUCs. For instance, we can define AUC_{ij} as the AUC obtained using scores $\hat{s}_i$ to rank instances from classes C_i and C_j. Notice that $\hat{s}_j$ may rank these instances differently, and so $\text{AUC}_{ji} \neq \text{AUC}_{ij}$. Taking an unweighted average over all $i \neq j$ estimates the probability that, for uniformly chosen classes i and $j \neq i$, and uniformly chosen $x \in C_i$ and $x' \in C_j$,we have $\hat{s}_i(x) > \hat{s}_i(x')$. The weighted version of this estimates the probability that the instances are correctly ranked if we don't pre-select the class.

The simplest way to turn multi-class scores into classifications is by assigning the class that achieves the maximum score – that is, if $\hat{\mathbf{s}}(x) = (\hat{s}_1(x), \ldots, \hat{s}_k(x))$ is the score vector assigned to instance x and $m = \text{argmax}_i\ \hat{s}_i\ (x)$, then the class assigned to x is $\hat{c}(x) = C_m$. However, just as in the two-class case such a fixed decision rule can be suboptimal, and instead we may want to learn it from data. What this means is that we want to learn a weight vector $\mathbf{w} = (w_1, \ldots, w_k)$ to adjust the scores and assign $\hat{c}(x) = C_{m'}$ with $m' = \text{argmax}_i\ w_i\hat{s}_i(x)$ instead.[2] Since the weight vector can be multiplied with a constant without affecting m', we can fix one of the degrees of freedom by setting $w_1 = 1$. Unfortunately, finding a globally optimal weight vector is computationally intractable. A heuristic approach that works well in practice is to first learn w_2 to optimally separate C_2 from C_1 as in the two-class case; then learn w_3 to separate C_3 from $C_1 \cup C_2$, and so on.

Figure 3.1. (left) Triples of probabilistic scores represented as points in an equilateral triangle connecting three corners of the unit cube. **(right)** The arrows show how the weights are adjusted from the initial equal weights (dotted lines), first by optimising the separation of C_2 against C_1 (dashed line), then by optimising the separation of C_3 against the other two classes (solid lines). The end result is that the weight of C_1 is considerably decreased, to the benefit of the other two classes.

Example 3.6 (Reweighting multi-class scores). We illustrate the procedure for a three-class probabilistic classifier. The probability vectors $\hat{\mathbf{p}}(x) = \left(\hat{p}_1(x), \hat{p}_2(x), \hat{p}_3(x)\right)$ can be thought of as points inside the unit cube. Since the probabilities add up to 1, the points lie in an equilateral triangle connecting three corners of the cube (Figure 3.1 (left)). Each corner of this triangle represents one of the classes; the probability assigned to a particular class in a given point is proportional to the distance to the opposite side.

Any decision rule of the form $\text{argmax}_i\ w_i\hat{s}_i(x)$ cuts the triangle in three areas using lines perpendicular to the sides. For the unweighted decision rule these lines intersect in the triangle's centre of mass (Figure 3.1 (right)). Optimising the separation between C_2 against C_1 means moving this point along a line parallel to the base of the triangle, moving away from the class that receives greater weight. Once the optimal point on this line is found, we optimise the separation of C_3 against the first two classes by moving in a direction perpendicular to the previous line.

Finally, we briefly look at the issue of obtaining calibrated multi-class

probabilities. This is not a solved problem and several approaches have been suggested in the literature. One of the simplest and most robust of these calculates normalised coverage counts. Specifically, we take the summed or averaged coverage counts of all classifiers that fire, and normalise these to obtain probability vectors whose components sum to one. Equivalently, we can obtain probability vectors for each classifier separately, and take a weighted average of these with weights determined by the relative coverage of each classifier.

Example 3.7 (Multi-class probabilities from coverage counts). In Example 3.4 on p.87 we can divide the class counts by the total number of positive predictions. This results in the following class distributions: (0.80, 0, 0.20) for the first classifier, (0.10, 0.85, 0.05) for the second classifier, and (0.29, 0.14, 0.57) for the third. The probability distribution associated with the combination of the first and third classifiers is

$$\frac{10}{24}(0.80,0,0.20)+\frac{14}{24}(0.29,0.14,0.57)=(0.50,0.08,0.42)$$

which is the same distribution as obtained by normalising the combined counts (12, 2, 10). Similarly, the distribution associated with all three classifiers is

$$\frac{10}{44}(0.80,0,0.20)+\frac{20}{44}(0.10,0.85,0.05)+\frac{14}{44}(0.29,0.14,0.57)=(0.32,0.43,0.25)$$

Matrix notation describes this very succinctly as

$$\begin{pmatrix} 10/24 & 0 & 14/24 \\ 10/44 & 20/44 & 14/44 \end{pmatrix}\begin{pmatrix} 0.80 & 0.00 & 0.20 \\ 0.10 & 0.85 & 0.05 \\ 0.29 & 0.14 & 0.57 \end{pmatrix}=\begin{pmatrix} 0.50 & 0.08 & 0.42 \\ 0.32 & 0.43 & 0.25 \end{pmatrix}$$

The middle matrix is a row-normalised version of the middle matrix in Equation 3.1. *Row normalisation* works by dividing each entry by the sum of the entries in the row in which it occurs. As a result the entries in each row sum to one, which means that each row can be interpreted as a probability distribution. The left matrix combines two pieces of information: (*i*) which classifiers fire for each example (for instance, the second classifier doesn't fire for the first example); and (*ii*) the coverage of

each classifier. The right-hand side then gives the class distribution for each example. Notice that the product of row-normalised matrices again gives a row-normalised matrix.

In this section we have seen that many interesting issues arise, once we have more than two classes. The general way of addressing a k-class learning problem with binary classifiers is to (*i*) break the problem up into l binary learning problems; (*ii*) train l binary classifiers on two-class versions of the original data; and (*iii*) combine the predictions from these l classifiers into a single k-class prediction. The most common ways to do the first and third step is one-versus-one or one-versus-rest, but the use of code matrices gives the opportunity of implementing other schemes. We have also looked at ways of obtaining multi-class scores and probabilities from the binary classifiers, and discussed a heuristic method to calibrate the multi-class decision rule by reweighting.

This concludes our discussion of classification, arguably the most common task in machine learning. In the remainder of this chapter we will look at one more supervised predictive task in the next section, before we turn our attention to unsupervised and descriptive learning in Section 3.3.

3.2 Regression

In all the tasks considered so far – classification, scoring, ranking and probability estimation – the label space was a discrete set of classes. In this section we will consider the case of a real-valued target variable. A *function estimator*, also called a *regressor*, is a mapping $\hat{f}:\mathscr{X}\rightarrow\mathbb{R}$. The regression learning problem is to learn a function estimator from examples $(x_i, f(x_i))$. For instance, we might want to learn an estimator for the Dow Jones index or the FTSE 100 based on selected economic indicators.

While this may seem a natural and innocuous generalisation of discrete classification, it is not without its consequences. For one thing, we switched from a relatively low-resolution target variable to one with infinite resolution. Trying to match this precision in the function estimator will almost certainly lead to overfitting – besides, it is highly likely that some part of the target values in the examples is due to fluctuations that the model is unable to capture. It is therefore entirely reasonable to assume that the examples are noisy, and that the

estimator is only intended to capture the general trend or shape of the function.

Figure 3.2. (left) Polynomials of different degree fitted to a set of five points. From bottom to top in the top right-hand corner: degree 1 (straight line), degree 2 (parabola), degree 3, degree 4 (which is the lowest degree able to fit the points exactly), degree 5. **(right)** A piecewise constant function learned by a grouping model; the dotted reference line is the linear function from the left figure.

Example 3.8 (Line fitting example). Consider the following set of five points:

x	y
1.0	1.2
2.5	2.0
4.1	3.7
6.1	4.6
7.9	7.0

We want to estimate y by means of a polynomial in x. Figure 3.2 (left) shows the result for degrees of 1 to 5 using ☞ *linear regression*, which will be explained in Chapter 7. The top two degrees fit the given points exactly (in general, any set of n points can be fitted by a polynomial of degree no more than $n - 1$), but they differ considerably at the extreme ends: e.g., the polynomial of degree 4 leads to a decreasing trend from $x = 0$ to $x = 1$, which is not really justified by the data.

To avoid overfitting the kind of data exemplified in Example 3.8 it is advisable to choose the degree of the polynomial as low as possible – often a simple linear relationship is assumed.

Regression is a task where the distinction between grouping and grading

models comes to the fore. The philosophy of grouping models is to cleverly divide the instance space into segments and learn a local model in each segment that is as simple as possible. For instance, in decision trees the local model is a majority class classifier. In the same spirit, to obtain a regression tree we could predict a constant value in each leaf. In the univariate problem of Example 3.8 this would result in the piecewise constant curve of Figure 3.2 (right). Notice that such a grouping model is able to fit the given points exactly, just as a polynomial of sufficiently high degree, and the same caveat regarding overfitting applies.

We can understand the phenomenon of overfitting a bit better by looking at the number of parameters that each model has. An n-degree polynomial has $n + 1$ parameters: e.g., a straight line $y = a \cdot x + b$ has two parameters, and the polynomial of degree 4 that fits the five points exactly has five parameters. A piecewise constant model with n segments has $2n - 1$ parameters: n y-values and $n - 1$ x-values where the 'jumps' occur. So the models that are able to fit the points exactly are the models with more parameters. A rule of thumb is that, *to avoid overfitting, the number of parameters estimated from the data must be considerably less than the number of data points*.

We have seen that classification models can be evaluated by applying a loss function to the margins, penalising negative margins (misclassifications) and rewarding positive margins (correct classifications). Regression models are evaluated by applying a loss function to the *residuals* $f(x) - \hat{f}(x)$. Unlike classification loss functions a regression loss function will typically be symmetric around 0 (although it is conceivable that positive and negative residuals have different weights). The most common choice here is to take the squared residual as the loss function. This has the advantage of mathematical convenience, and can also be justified by the assumption that the observed function values are the true values contaminated by additive, normally distributed noise. However, it is well-known that squared loss is sensitive to outliers: you can see an example of this in Figure 7.2 on p.199.

If we underestimate the number of parameters of the model, we will not be able to decrease the loss to zero, regardless of how much training data we have. On the other hand, with a larger number of parameters the model will be more dependent on the training sample, and small variations in the training sample can result in a considerably different model. This is sometimes called the *bias–variance dilemma*: a low-complexity model suffers less from variability due to random variations in the training data, but may introduce a systematic bias that even large amounts of training data can't resolve; on the other hand, a high-complexity model eliminates such bias but can suffer non-systematic errors due

to variance.

We can make this a bit more precise by noting that expected squared loss on a training example x can be decomposed as follows:[3]

$$\mathbb{E}\left[\left(f(x)-\hat{f}(x)\right)^2\right] = \left(f(x)-\mathbb{E}\left[\hat{f}(x)\right]\right)^2 + \mathbb{E}\left[\left(\hat{f}(x)-\mathbb{E}\left[\hat{f}(x)\right]\right)^2\right] \tag{3.2}$$

Figure 3.3. A dartboard metaphor illustrating the concepts of bias and variance. Each dartboard corresponds to a different learning algorithm, and each dart signifies a different training sample. The top row learning algorithms exhibit low bias, staying close to the bull's eye (the true function value for a particular x) on average, while the ones on the bottom row have high bias. The left column shows low variance and the right column high variance.

It is important to note that the expectation is taken over different training sets and hence different function estimators, but the learning algorithm and the example are fixed. The first term on the right-hand side in Equation 3.2 is zero if these function estimators get it right on average; otherwise the learning algorithm exhibits a systematic *bias* of some kind. The second term quantifies the *variance* in the function estimates $\hat{f}(x)$ as a result of variations in the training set. Figure 3.3 illustrates this graphically using a dartboard metaphor. The best situation is clearly achieved in the top left-hand corner of the figure, but in practice this is rarely achievable and we need to settle either for a low bias and a high variance (e.g., approximating the target function by a high-degree polynomial) or for a high bias and a low variance (e.g., using a linear approximation). We will return to the bias–variance dilemma at several places in the book: although the decomposition is not unique for most loss functions other

than squared loss, it serves as a useful conceptual tool for understanding over- and underfitting.

3.3 Unsupervised and descriptive learning

So far, we have concerned ourselves exclusively with supervised learning of predictive models. That is, we learn a mapping from instance space $\mathscr{X}$ to output space $\mathscr{Y}$ using labelled examples $(x, l(x)) \in \mathscr{X} \times \mathscr{L}$ (or a noisy version thereof). This kind of learning is called 'supervised' because of the presence of the target variable $l(x)$ in the training data, which has to be supplied by a 'supervisor' or 'teacher' with some knowledge about the true labelling function l. Furthermore, the models are called 'predictive' because the outputs produced by the models are either direct estimates of the target variable or provide us with further information about its most likely value. Thus, we have only paid attention to the top-left entry in Table 3.1. In the remainder of this chapter we will briefly introduce the other three learning settings by means of selected examples:

- unsupervised learning of a predictive model in the form of predictive clustering;
- unsupervised learning of a descriptive model, exemplified by descriptive clustering and association rule discovery;
- supervised learning of a descriptive model, with subgroup discovery as practical realisation.

	Predictive model	*Descriptive model*
Supervised learning	classification, regression	**subgroup discovery**
Unsupervised learning	**predictive clustering**	**descriptive clustering, association rule discovery**

Table 3.1. The learning settings indicated in **bold** are introduced in the remainder of this chapter.

It is worthwhile reflecting for a moment on the nature of descriptive learning. The task here is to come up with a description of the data – to produce a descriptive model. It follows that the task output, being a model, is of the same kind as the learning output. Furthermore, it makes no sense to employ a separate

training set to produce the descriptive model, as we want the model to describe our actual data rather than some hold-out set. In other words, *in descriptive learning the task and learning problem coincide* (Figure 3.4). This makes some things harder: for example, it is unlikely that a 'ground truth' or 'gold standard' is available to test the descriptive models against, and hence evaluating descriptive learning algorithms is much less straightforward than evaluating predictive ones. On the other hand, one could say that descriptive learning leads to the *discovery* of genuinely new knowledge, and it is often situated at the intersection of machine learning and data mining.

Figure 3.4. In descriptive learning the task and learning problem coincide: we do not have a separate training set, and the task is to produce a descriptive model of the data.

Predictive and descriptive clustering

The distinction between predictive and descriptive models can be clearly observed in clustering tasks. One way to understand clustering is as learning a new labelling function from unlabelled data. So we could define a 'clusterer' in the same way as a classifier, namely as a mapping $\hat{q}:\mathscr{X}\rightarrow\mathscr{C}$, where $\mathscr{C}=\{C_1,C_2,\ldots,C_k\}$ is a set of new labels. This corresponds to a *predictive* view of clustering, as the domain of the mapping is the entire instance space, and hence it generalises to unseen instances. A *descriptive* clustering model learned from given data $D\subseteq\mathscr{X}$ would bea mapping $\hat{q}:D\rightarrow\mathscr{C}$ whose domain is D rather than $\mathscr{X}$. In either case the labels have no intrinsic meaning, other than to express whether two instances belong to the same cluster. So an alternative way to define a clusterer is as an equivalence relation $\hat{q}\subseteq\mathscr{X}\times\mathscr{X}$ or $\hat{q}\subseteq D\times D$ (see Background 2.1 on p.51 for the definition of an equivalence relation), or, equivalently, as a partition of $\mathscr{X}$ or D.

The distinction between predictive and descriptive clustering is subtle and not always articulated clearly in the literature. Several well-known clustering algorithms including ☞ *K-means* (discussed in more detail in Chapter 8) learn a predictive clustering. Thus, they learn a clustering model from training data that

can subsequently be used to assign new data to clusters. This is in keeping with our distinction between the task (clustering arbitrary data) and the learning problem (learning a clustering model from training data). However, this distinction isn't really applicable to descriptive clustering methods: here, the clustering model learned from D can only be used to cluster D. In effect, the task becomes learning a suitable clustering model for the given data.

Without any further information, any clustering is as good as any other. What distinguishes a good clustering is that the data is partitioned into *coherent* groups or clusters. 'Coherence' here means that, on average, two instances from the same cluster have more in common – are more similar – than two instances from different clusters. This assumes some way of assessing the similarity or, as is usually more convenient, the dissimilarity or distance of an arbitrary pair of instances. If our features are numerical, i.e., $\mathscr{X} = \mathbb{R}^d$, the most obvious distance measure is Euclidean distance, but other choices are possible, some of which generalise to non-numerical features. Most distance-based clustering methods depend on the possibility of defining a 'centre of mass' or *exemplar* for an arbitrary set of instances, such that the exemplar minimises some distance-related quantity over all instances in the set, called its *scatter*. A good clustering is then one where the scatter summed over each cluster – the *within-cluster scatter* – is much smaller than the scatter of the entire data set.

This analysis suggests a definition of the clustering problem as finding a partition $D = D_1 \uplus \ldots \uplus D_K$ that minimises the within-cluster scatter. However, there are a few issues with this definition:

- the problem as stated has a trivial solution: set $K = |D|$ so that each 'cluster contains a single instance from D and thus has zero scatter;
- if we fix the number of clusters K in advance, the problem cannot be solve efficiently for large data sets (it is NP-hard).

The first problem is the clustering equivalent of overfitting the training data. It could be dealt with by penalising large K. Most approaches, however, assume that an educated guess of K can be made. This leaves the second problem, which is that finding a globally optimal solution is intractable for larger problems. This is a well-known situation in computer science and can be dealt with in two ways:

- by applying a heuristic approach, which finds a 'good enough' solution rather than the best possible one;
- by relaxing the problem into a 'soft' clustering problem, by allowing

instances a degree of membership in more than one cluster.

Most clustering algorithms follow the heuristic route, including the K-means algorithm. The soft clustering approach can be addressed in various ways, including ☞ *Expectation-Maximisation* (Section 9.4) and ☞ *matrix decomposition* (Section 10.3). Figure 3.5 illustrates the heuristic and soft clustering approaches. Notice that a soft clustering generalises the notion of a partition, in the same way that a probability estimator generalises a classifier.

The representation of clustering models depends on whether they are predictive, descriptive or soft. A descriptive clustering of n data points into c clusters could be represented by a *partition matrix*: an n-by-c binary matrix with exactly one 1 in each row (and at least one 1 in each column, otherwise there would be empty clusters). A soft clustering corresponds to a row-normalised n-by-c matrix. A predictive clustering partitions the whole instance space and is therefore not suitable for a matrix representation. Typically, predictive clustering methods represent a cluster by their *centroid* or *exemplar*: in that case, the cluster boundaries are a set of straight lines called a *Voronoi diagram* (Figure 3.5 (left)). More generally, each cluster could be represented by a probability density, with the boundaries occurring where densities of neighbouring clusters are equal; this would allow non-linear cluster boundaries.

Figure 3.5. (left) An example of a predictive clustering. The coloured dots were sampled from three bivariate Gaussians centred at (1,1), (1,2) and (2,1). The crosses and solid lines are the cluster exemplars and cluster boundaries found by 3-means. **(right)** A soft clustering of the same data found by matrix decomposition.

Example 3.9 (Representing clusterings). The cluster exemplars in Figure

3.5 (left) can be given as a c-by-2 matrix:

$$\begin{pmatrix} 0.92 & 0.93 \\ 0.98 & 2.02 \\ 2.03 & 1.04 \end{pmatrix}$$

The following n-by-c matrices represent a descriptive clustering (left) and a soft clustering (right) of given data points:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ \cdots & \cdots & \cdots \end{pmatrix} \qquad \begin{pmatrix} 0.40 & 0.30 & 0.30 \\ 0.40 & 0.51 & 0.09 \\ 0.44 & 0.29 & 0.27 \\ 0.35 & 0.08 & 0.57 \\ \cdots & \cdots & \cdots \end{pmatrix}$$

An interesting question is how clustering models should be evaluated. In the absence of labelled data we cannot use a test set in the same way as we would in classification or regression. We can use within-cluster scatter as a measure of the quality of a clustering. For a predictive clustering it is possible to evaluate within-cluster scatter on hold-out data that wasn't used to build the clusters in the first place. An alternative way of evaluating a clustering arises if we have some knowledge about instances that should, or should not, be clustered together.

Example 3.10 (Evaluating clusterings). Suppose we have five test instances that we think should be clustered as $\{e1, e2\}$, $\{e3, e4, e5\}$. So out of the $5 \cdot 4 = 20$ possible pairs, 4 are considered 'must-link' pairs and the other 16 as 'must-not-link' pairs. The clustering to be evaluated clusters these as $\{e1, e2, e3\}$, $\{e4, e5\}$ – so two of the must-link pairs are indeed clustered together ($e1$–$e2$, $e4$–$e5$), the other two are not ($e3$–$e4$, $e3$–$e5$), and so on.

We can tabulate this as follows:

	Are together	*Are not together*	
Should be together	2	2	4
Should not be together	2	14	16
	4	16	20

We can now treat this as a two-by-two contingency table, and evaluate it accordingly. For instance, we can take the proportion of pairs on the 'good' diagonal, which is 16/20 = 0.8. In classification we would call this accuracy, but in the clustering context this is known as the *Rand index*.

Note that there are usually many more must-not-link pairs than must-link pairs, and it is a good idea to compensate for this. One way to do that is to calculate the harmonic mean of precision and recall (the latter the same as true positive rate, see Table 2.3 on p.57), which in the information retrieval literature is known as the *F-measure*.[4] Precision is calculated on the left column of the contingency table and recall on the top row; as a result the bottom right-hand cell (the must-not-link pairs that are correctly not clustered together) are ignored, which is precisely what we want. In the example both precision and recall are 2/4 = 0.5, and so is the F-measure. This shows that the relatively good Rand index is mostly accounted for by the must-not-link pairs that end up in different clusters.

Other descriptive models

To wrap up our catalogue of machine learning tasks we will briefly look at two other descriptive models, one learned in a supervised fashion from labelled data and the other entirely unsupervised.

Subgroup models don't try to approximate the labelling function, but rather aim at identifying subsets of the data exhibiting a class distribution that is significantly different from the overall population. Formally, a *subgroup* is a mapping $\hat{g}: D \rightarrow \{\text{true}, \text{false}\}$ and is learned from a set of labelled examples $l: \mathscr{X} \rightarrow \mathscr{C}$, where $G = \{x \in D | \hat{g}(x) =$ is the true labelling function. Note that $\hat{g}$ is the characteristic function of the set $G = \{x \in D | \hat{g}(x) = \text{true}\}$, which is called the *extension* of the subgroup. Note also that we used the given data D rather than the whole instance space $\mathscr{X}$ for the domain of a subgroup, since it is a descriptive model.

Example 3.11 (Subgroup discovery). Imagine you want to market the new version of a successful product. You have a database of people who have been sent information about the previous version, containing all kinds of demographic, economic and social information about those people, as well as whether or not they purchased the product. If you were to build a classifier or ranker to find the most likely customers for your product, it is unlikely to outperform the majority class classifier (typically, relatively few people will have bought the product). However, what you are really interested in is finding reasonably sized subsets of people with a proportion of customers that is significantly higher than in the overall population. You can then target those people in your marketing campaign, ignoring the rest of your database.

A subgroup is essentially a binary classifier, and so one way to develop a subgroup discovery system is to adapt an existing classifier training algorithm. This may not involve much more than adapting the search heuristic to reflect the specific objective of a subgroup (to identify subsets of the data with a significantly different class distribution). However, this would only give us a single subgroup. Rule learners are particularly appropriate for subgroup discovery since every rule can be interpreted as a separate subgroup.

How do we distinguish interesting subgroups from uninteresting ones? This can be determined by constructing a contingency table similar to the ones we use in binary classification. For three classes such a table looks as follows:

	In subgroup	*Not in subgroup*	
Labelled C_1	g_1	$C_1 - g_1$	C_1
Labelled C_2	g_2	$C_2 - g_2$	C_2
Labelled C_3	g_3	$C_3 - g_3$	C_3
	$\|G\|$	$\|D\| - \|G\|$	$\|D\|$

where $g_i = |\{x \in D | \hat{g}(x) = \text{true} \wedge l(x) = C_i\}|$ and C_i is shorthand for $|\{x \in D | l(x) = C_i\}|$. From here there are a number of possibilities. One idea is to measure the extent to which the class distribution in the left column is different from the class distribution in the row marginals (the right-most column). As we shall see later (Example 6.6 on p.180), this boils down to using an adaptation of average recall as evaluation measure. Another idea is to treat the subgroup as a

decision tree split and borrow splitting criteria from ☞ *decision tree* learning (Section 5.1). It is also possible to use the χ^2 statistic to evaluate the extent to which each g_i differs from what would be expected on the basis of the marginals C_i and $|G|$. What these evaluation measures have in common is that they prefer different class distributions in the subgroup and its complement from the overall distribution in D, and also larger subgroups over smaller ones. Most of these measures are actually symmetric in that they assign the same evaluation to a subgroup and its complement, from which it follows that they also prefer larger complements over smaller ones – in other words, they prefer subgroups that are about half the size of the data (other things being equal).

I will now give an example of unsupervised learning of descriptive models. Associations are things that usually occur together. For example, in market basket analysis we are interested in items frequently bought together. An example of an association rule is ·**if** beer **then** crisps·, stating that customers who buy beer tend to also buy crisps. Association rule discovery starts with identifying feature values that often occur together. There is some superficial similarity with subgroups here, but these so-called frequent item sets are identified in a purely unsupervised manner, without need for labelled training data. Item sets then give rise to rules describing co-occurrences between feature values. These association rules are if-then rules similar to classification rules, except that the then-part isn't restricted to a particular class variable and can contain any feature (or even several features). Rather than adapting a given learning algorithm we need a new algorithm that first finds frequent item sets and then turns them into association rules. The process needs to take into account a mix of statistics in order to avoid generating trivial rules.

Example 3.12 (Association rule discovery). In a motorway service station most clients will buy petrol. This means that there will be many frequent item sets involving petrol, such as {newspaper, petrol}. This might suggest the construction of an association rule ·**if** newspaper **then** petrol· –however, this is predictable given that {petrol} is already a frequent item set (and clearly at least as frequent as {newspaper, petrol}). Of more interest would be the converse rule ·**if** petrol **then** newspaper· which expresses that a considerable proportion of the people buying petrol also buy a newspaper.

We clearly see a relationship with subgroup discovery in that association rules also identify subsets that have a different distribution when compared with the full data set, namely with respect to the then-part of the rule. The difference is that the then-part is not a fixed target variable but it is found as part of the discovery process. Both subgroup discovery and association rule discovery will be discussed in the context of rule learning in Section 6.3.

3.4 Beyond binary classification: Summary and further reading

While binary classification is an important task in machine learning, there are many other relevant tasks and in this chapter we looked at a number of them.

- In Section 3.1 we considered classification tasks with more than two classes. We shall see in the coming chapters that some models handle this situation very naturally, but if our models are essentially two-class (such a linear models) we have to approach it via a combination of binary classification tasks. One key idea is the use of a code matrix to combine th results of several binary classifiers, as proposed by Dietterich and Bakiri (1995) under the name 'error-correcting output codes' and developed by Allwein *et al.* (2000). We also looked at ways to obtain scores for more than two classes and to evaluate those scores using multi-class adaptations of the area under the ROC curve. One of these multi-class extensions of AUC was proposed and analysed by Hand and Till (2001). The heuristic procedure for reweighting multi-class scores in Example 3.6 on p.89 was proposed by Lachiche and Flach (2003); Bourke *et al.* (2008) demonstrate that it achieves good performance in comparison with a number of alternative approaches.

- Section 3.2 was devoted to regression: predicting a real-valued target valu This is a classical data analysis problem that was already studied by Carl Friedrich Gauss in the late eighteenth century. It is natural to use a quadratic loss function on the residuals, although this carries with it a certain sensitivity to outliers. Grading models are most common here, although it is also possible to learn a grouping model that divides the instance space into segments that admit a simple local model. Since it is often possible to fit a set of points exactly (e.g., with a high-degree polynomial), care must be taken to avoid overfitting. Finding the right

balance between over-and underfitting is sometimes called the bias–variance dilemma; an extensive discussion (including the dartboard metaphor) can be found in Rajnarayan and Wolpert (2010).

☞ In Section 3.3 we considered unsupervised and descriptive learning tasks. We saw that in descriptive learning the task and learning problem coincide A clustering model can be either predictive or descriptive: in the former case it is meant to construct classes in a wholly unsupervised manner, afte which the learned model can be applied to unseen data in the usual way. Descriptive clustering, on the other hand, only applies to the data at hand. should be noted that the distinction between predictive and descriptive clustering is not universally recognised in the literature; sometimes the ter 'predictive clustering' is used in the slightly different sense of clustering simultaneously on the target variable and the features (Blockeel *et al.*, 1998).

☞ Like descriptive clustering, association rule discovery is another descriptiv task which is wholly unsupervised. It was introduced by Agrawal, Imielinski and Swami (1993) and has given rise to a very large body of work in the data mining literature. Subgroup discovery is a form of supervised learning of descriptive models aimed at finding subsets of the data with a significantly different distribution of the target variable. It was first studied by Klösgen (1996) and extended to the more general notion o exceptional model mining in order to deal with, e.g., real-valued target variables by Leman *et al.* (2008). More generally, unsupervised learning o descriptive models is a large subject that was pioneered by Tukey (1977).

---

[1]This is a slight generalisation of the *Hamming distance* for binary strings, which counts the number of positions in which the two strings differ.

[2]Notice that with two classes such a weighted decision rule assigns class C_1 if $w_1 \hat{s}_1(x) > w_2 \hat{s}_2(x)$, or equivalently, $\hat{s}_1(x)/\hat{s}_2(x) > w_2/w_1$. This can be interpreted as a threshold on suitably transformed scores, so the weighted decision rule indeed generalises the two-class decision threshold.

[3]The derivation expands the squared difference term, making use of the linearity of $\mathbb{E}[\cdot]$ and that $\mathbb{E}[f(x)] = f(x)$, after which terms can be rearranged to yield Equation 3.2.

[4]The harmonic mean of precision and recall is $\frac{2}{1/prec + 1/rec} = \frac{2 prec \cdot rec}{prec + rec}$. The harmonic mean is appropriate for averaging ratios; see Background 10.1 on p.300.

CHAPTER 4

Concept learning

HAVING DISCUSSED A VARIETY of tasks in the preceding two chapters, we are now in an excellent position to start discussing machine learning models and algorithms for learning them. This chapter and the next two are devoted to logical models, the hallmark of which is that they use logical expressions to divide the instance space into segments and hence construct grouping models. The goal is to find a segmentation such that the data in each segment is more homogeneous, with respect to the task to be solved. For instance, in classification we aim to find a segmentation such that the instances in each segment are predominantly of one class, while in regression a good segmentation is such that the target variable is a simple function of a small number of

predictor variables. There are essentially two kinds of logical models: tree models and rule models. Rule models consist of a collection of implications or if–then rules, where the if-part defines a segment, and the then-part defines the behaviour of the model in this segment. Tree models are a restricted kind of rule model where the if-parts of the rules are organised in a tree structure.

In this chapter we consider methods for learning logical expressions or *concepts* from examples, which lies at the basis of both tree models and rule models. In concept learning we only learn a description for the positive class, and label everything that doesn't satisfy that description as negative. We will pay particular attention to the generality ordering that plays an important role in logical models. In the next two chapters we consider tree and rule models, which go considerably beyond concept learning as they can handle multiple classes, probability estimation, regression, as well as clustering tasks.

The simplest logical expressions are equalities of the form Feature = Value and, for numerical features, inequalities of the form Feature < Value; these are called *literals*. Complex Boolean expressions can be built using logical connectives: *conjunction* $\wedge$, *disjunction* $\vee$, *negation* $\neg$ and *implication* $\rightarrow$. The following equivalences hold (the left two are called the *De Morgan laws*):

$$\neg(A \wedge B) \equiv \neg A \vee \neg B \qquad \neg\neg A \equiv A$$
$$\neg(A \vee B) \equiv \neg A \wedge \neg B \qquad A \rightarrow B \equiv \neg A \vee B$$

If Boolean expression A is true of instance x, we say that A *covers* x. The set of instances covered by expression A is called its *extension* and denoted $\mathscr{X}_A = \{x \in \mathscr{X} | A \text{ covers } x\}$, where $\mathscr{X}$ denotes the instance space which acts as the universe of discourse (see Background 2.1 on p.51). There is a direct correspondence between logical connectives and operations on sets: e.g., $\mathscr{X}_{A \wedge B} = \mathscr{X}_A \cap \mathscr{X}_B$, $\mathscr{X}_{A \vee B} = \mathscr{X}_A \cup \mathscr{X}_B$ and $\mathscr{X}_{\neg A} = \mathscr{X} \setminus \mathscr{X}_A$. If $\mathscr{X}_A \supseteq \mathscr{X}_{A'}$, we say that A is *at least as general as* A', and if in addition $\mathscr{X}_A \not\subseteq \mathscr{X}_{A'}$ we say that A is *more general than* A'. This *generality ordering* is a partial order on logical expressions as defined in Background 2.1. (More precisely: it is a partial order on the equivalence classes of the relation of logical equivalence $\equiv$.) A *clause* is an implication $P \rightarrow Q$ such that P is a conjunction of literals and Q is a disjunction of literals. Using the equivalences above we can rewrite such an implication as

$$(A \wedge B) \rightarrow (C \vee D) \equiv \neg(A \wedge B) \vee (C \vee D) \equiv \neg A \vee \neg B \vee C \vee D$$

and hence a clause can equivalently be seen as a disjunction of literals or their negations. Any logical expression can be rewritten as a conjunction of clauses; this is referred to as *conjunctive normal form* (*CNF*). Alternatively, any logical expression can be written as a disjunction of conjunctions of literals or their negation; this is called *disjunctive normal form* (*DNF*). A *rule* is a clause $A \rightarrow B$ where B is a single literal; this is also often referred to as a *Horn clause*, after the American logician Alfred Horn.

Background 4.1. Some logical concepts and notation.

4.1 The hypothesis space

The simplest concept learning setting is where we restrict the logical expressions describing concepts to conjunctions of literals (see Background 4.1 for a review of important

definitions and notation from logic). The following example illustrates this.[1]

Example 4.1 (Learning conjunctive concepts). Suppose you come across a number of sea animals that you suspect belong to the same species. You observe their length in metres, whether they have gills, whether they have a prominent beak, and whether they have few or many teeth. Using these features, the first animal can described by the following conjunction: Length = 3 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many The next one has the same characteristics but is a metre longer, so you drop the length condition and generalise the conjunction to Gills = no ∧ Beak = yes ∧ Teeth = many The third animal is again 3 metres long, has a beak, no gills and few teeth, so your description becomes Gills = no ∧ Beak = yes All remaining animals satisfy this conjunction, and you finally decide they are some kind of dolphin.

Despite the simplicity of this example, the space of possible concepts – usually called the *hypothesis space* – is already fairly large. Let's assume we have three different lengths: 3, 4 and 5 metres, while the other three features have two values each. We then have $3 \cdot 2 \cdot 2 \cdot 2 = 24$ possible instances. How many conjunctive concepts are there using these same features? We can answer this question if we treat the absence of a feature as an additional 'value'. This gives a total of $4 \cdot 3 \cdot 3 \cdot 3 = 108$ different concepts. While this seems quite a lot, you should realise that the number of possible extensions – sets of instances – is much larger: 2^{24}, which is more than 16 million! That is, if you pick a random set of instances, the odds that you can't find a conjunctive concept that exactly describes those instances are well over 100 000 to 1. This is actually a good thing, as it forces the learner to generalise beyond the training data and cover instances that it hasn't seen before. Figure 4.1 depicts this hypothesis space, making use of the generality ordering (i.e., the subset relationship between concept extensions; see Background 4.1).

Figure 4.1. The hypothesis space corresponding to Example 4.1. The bottom row corresponds to the 24 possible instances, which are complete conjunctions with four literals each. The next row up are all 44 concepts with three literals each; then 30 concepts with two literals each; 9 concepts consisting of a single literal; and the top concept is the empty conjunction which is always true and hence covers all possible instances. A connecting line is drawn between concepts on consecutive layers if the higher one is more general than the lower one (i.e., the higher concept's extension is a superset of the lower's).

Least general generalisation *If we rule out all concepts that don't cover at least one of the instances in Example 4.1, the hypothesis space is reduced to 32 conjunctive concepts (Figure 4.2). Insisting that any hypothesis cover all three instances reduces this further to only four concepts, the least general one of which is the one found in the example – it is called their least general generalisation (LGG). Algorithm 4.1 formalises the procedure, which is simply to repeatedly apply a pairwise LGG operation (Algorithm 4.2) to an instance and the current hypothesis, as they both have the same logical form. The structure of the hypothesis space ensures that the result is independent of the order in which the instances are processed.*

Intuitively, the LGG of two instances is the nearest concept in the hypothesis space where paths upward from both instances intersect. The fact that this point is unique is a special property of many logical hypothesis spaces, and can be put to good use in learning. More precisely, such a hypothesis space forms a *lattice*: a partial order in which each two elements have a *least upper bound* (*lub*) and a *greatest lower bound* (*glb*). So, the LGG of a set of instances is exactly the least upper bound of the instances in that lattice. Furthermore, it is the greatest lower bound of the set of all generalisations of the instances: all possible generalisations are at least as general as the LGG. In this very precise sense, *the LGG is the most conservative generalisation that we can learn from the data*.

If we want to be a bit more adventurous, we could choose one of the more general hypotheses, such as Gills = no or Beak = yes. However, we probably don't want to choose the most general hypothesis, which is simply that every animal is a dolphin, as this would clearly be an over-generalisation. Negative examples are very useful to prevent over-generalistion.

Algorithm 4.1: LGG-Set(D) – find least general generalisation of a set of instances.

```
Input  : data D.
Output : logical expression H.
x ← first instance from D;
H ← x;
while instances left do
    x ← next instance from D;
    H ← LGG(H, x) ;          // e.g., LGG-Conj (Alg. 4.2) or LGG-Conj-ID (Alg. 4.3)
end
return H
```

Figure 4.2. Part of the hypothesis space in Figure 4.1 containing only concepts that are more general than at least one of the three given instances on the bottom row. Only four conjunctions, indicated in green at the top, are more general than all three instances; the least general of these is Gills = no ∧ Beak = yes. It can be observed that the two left-most and right-most instances would be sufficient to learn that concept.

Example 4.2 (Negative examples). In Example 4.1 we observed the following dolphins: p1: Length = 3 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many p2: Length = 4 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many p3: Length = 3 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few Suppose you next observe an animal that clearly doesn't belong to the species – a negative example. It is described by the following conjunction: n1: Length = 5 ∧ Gills = yes ∧ Beak = yes ∧ Teeth = many This negative example rules out some of the generalisations that were hitherto still possible: in particular, it rules out the concept Beak = yes, as well as the empty concept which postulates that everything is a dolphin.

The process is illustrated in Figure 4.3. We now have two hypotheses left, one which is least general and the other most general.

Internal disjunction You might be tempted to conclude from this and the previous example that we always have a unique most general hypothesis, but that is not the case in general. To demonstrate that, we are going to make our logical language slightly richer, by allowing a restricted form of disjunction called internal disjunction. The idea is very simple: if you observe one dolphin that is 3 metres long and another one of 4 metres, you may want to add the condition 'length is 3 or 4 metres' to your concept. We will write this as Length = [3, 4], which logically means Length = 3 ∨ Length = 4. This of course only makes sense for features that have more than two values: for instance, the internal disjunction Teeth = [many, few] is always true and can be dropped.

Algorithm 4.2: LGG-Conj(x, y) – find least general conjunctive generalisation of two conjunctions.

Input : conjunctions x, y.
Output : conjunction z.
1 $z \leftarrow$conjunction of all literals common to x and y;
2 **return** z

Figure 4.3. A negative example can rule out some of the generalisations of the LGG of the positive examples. Every concept which is connected by a red path to a negative example covers that negative and is therefore ruled out as a hypothesis. Only two conjunctions cover all positives and no negatives: Gills = no ∧ Beak = yes and Gills = no.

Example 4.3 (Internal disjunction). Using the same three positive examples as in Example 4.1, the second and third hypothesis are now Length = [3,4] ∧ Gills = no ∧ Beak = yes ∧ Teeth = many and

Length = [3,4] ∧ Gills = no ∧ Beak = yes We can drop any of the three conditions in the latter LGG without covering the negative example from Example 4.2. Generalising further to single conditions, we see that Length = [3, 4] and Gills = no are still OK but Beak = yes is not, as it covers the negative example.

Algorithm 4.3 details how we can calculate the LGG of two conjunctions employing internal disjunction. The function Combine-ID(v_x, v_y) returns [v_x, v_y] if v_x and v_y are constants, and their union if v_x or v_y are already sets of values:

e.g., Combine-ID([3, 4], [4, 5])= [3, 4, 5].

4.2 Paths through the hypothesis space As we can clearly see in Figure 4.4, in this example we have not one but two most general hypotheses. What we can also notice is that *every concept between the least general one and one of the most general ones is also a possible hypothesis*, i.e., covers all the positives and none of the negatives. Mathematically speaking we say that the set of hypotheses that agree with the data is a *convex set*, which basically means that we can interpolate between any two members of the set, and if we find a concept that is less general than one and more general than the other then that concept is also a member of the set. This in turn means that we can describe the set of all possible hypotheses by its least and most general members. This is summed up in the following definition.

Algorithm 4.3: LGG-Conj-ID(x, y) – find least general conjunctive generalisation of two conjunctions, employing internal disjunction.

```
Input  : conjunctions x, y.
Output : conjunction z.
z ← true;
for each feature f do
    if f = v_x is a conjunct in x and f = v_y is a conjunct in y then
        add f = Combine-ID(v_x, v_y) to z ;          // Combine-ID: see text
    end
end
return z
```

Figure 4.4. (top) A snapshot of the expanded hypothesis space that arises when internal disjunction is used for the 'Length' feature. We now need one more generalisation step to travel upwards from a completely specified example to the empty conjunction. **(bottom)** The version space consists of one least general hypothesis, two most general hypotheses, and three in between.

Definition 4.1 (Version space). *A concept is* complete *if it covers all positive examples. A concept is* consistent *if it covers none of the negative examples.*

The version space *is the set of all complete and consistent concepts. This set is convex and is fully defined by its least and most general elements.*

Figure 4.5. (left) A path in the hypothesis space of Figure 4.3 from one of the positive examples (p1, see Example 4.2 on p.110) all the way up to the empty concept. Concept A covers a single example; B covers one additional example; C and D are in the version space, and so cover all three positives; E and F also cover the negative. **(right)** The corresponding coverage curve, with ranking p1 – p2 – p3 – n1.

We can draw a useful connection between logical hypothesis spaces and the coverage plots introduced in Chapter 2. Suppose you were to follow a path in the hypothesis space from a positive example, through a selection of its generalisations, all the way up to the empty concept. The latter, by construction, covers all positives and all negatives, and hence occupies the top-right point (*Neg*, *Pos*) in the coverage plot. The starting point, being a single positive example, occupies the point (0, 1) in the coverage plot. In fact, it is customary to extend the hypothesis space with a bottom element which doesn't cover any examples and hence is less general than any other concept. Taking that point as the starting point of the path means that we start in the bottom-left point (0, 0) in the coverage plot.

Moving upwards in the hypothesis space by generalisation means that the numbers of covered positives and negatives can stay the same or increase, but never decrease. In other words, *an upward path through the hypothesis space corresponds to a coverage curve* and hence to a ranking. Figure 4.5 illustrates this for the running example. The chosen path is but one among many possible paths; however, notice that if a path, like this one, includes elements of the

version space, the corresponding coverage curve passes through 'ROC heaven' (0, *Pos*) and AUC = 1. In other words, such paths are optimal. Concept learning can be seen as the search for an optimal path through the hypothesis space.

What happens, you may ask, if the LGG of the positive examples covers one or more negatives? In that case, any generalisation of the LGG will be inconsistent as well. Conversely, any consistent hypothesis will be incomplete. It follows that the version space is empty in this case; we will say that the data is not *conjunctively separable*. The following example illustrates this.

Example 4.4 (Data that is not conjunctively separable). Suppose we have the following five positive examples (the first three are the same as in Example 4.1): p1: Length = 3 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many p2: Length = 4 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many p3: Length = 3 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few p4: Length = 5 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many p5: Length = 5 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few and the following negatives (the first one is the same as in Example 4.2): n1: Length = 5 ∧ Gills = yes ∧ Beak = yes ∧ Teeth = many n2: Length = 4 ∧ Gills = yes ∧ Beak = yes ∧ Teeth = many n3: Length = 5 ∧ Gills = yes ∧ Beak = no ∧ Teeth = many n4: Length = 4 ∧ Gills = yes ∧ Beak = no ∧ Teeth = many n5: Length = 4 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few The least general complete hypothesis is Gills = no ∧ Beak = yes as before, but this covers n5 and hence is inconsistent. There are seven most general consistent hypotheses, none of which are complete:

Length = 3	(covers p1 and p3)
Length = [3, 5] ∧ Gills = no	(covers all positives except p2)
Length = [3, 5] ∧ Teeth = few	(covers p3 and p5)
Gills = no ∧ Teeth = many	(covers p1, p2 and p4)
Gills = no ∧ Beak = no	
Gills = yes ∧ Teeth = few	
Beak = no ∧ Teeth = few	

The last three of these do not cover any positive examples.

Most general consistent hypotheses As this example suggests, finding most general consistent hypotheses is considerably more involved than finding least general complete ones. Essentially, the process is one of enumeration. Algorithm 4.4 gives an algorithm which returns all most general consistent specialisations of a given concept, where a minimal specialisation of a concept is one that can be reached in one downward step in the hypothesis lattice (e.g., by adding a conjunct, or removing a value from an internal disjunction). Calling the algorithm with C = true returns the most general consistent hypotheses.

Figure 4.6 shows a path through the hypothesis space of Example 4.4, and the corresponding coverage curve. We see that the path goes through three consistent hypotheses, which are consequently plotted on the *y*-axis of the coverage plot. The other three hypotheses are complete, and therefore end up on the top of the graph; one of these is, in fact, the LGG of the positives (D). The ranking corresponding to this coverage curve is p3 – p5 – [p1, p4] – [p2, n5] – [n1–4]. This ranking commits half a ranking error out of 25, and so AUC = 0.98. We can choose one concept from the ranking by applying the techniques discussed in Section 2.2. For instance, suppose that classification accuracy is the criterion we want to optimise. In coverage space, accuracy isometrics have slope 1, and so we see immediately that concepts C and D (or E) both achieve the best accuracy in Figure 4.6. If performance on the positives is more important we prefer the complete but inconsistent concept D; if performance on the negatives is valued more we choose the incomplete but consistent concept C.

Closed concepts It is worthwhile to reflect on the fact that concepts D and E occupy the same point in coverage space. What this means is that generalising D into E by dropping Beak = yes does not change the coverage in terms of positive and negative examples. One could say that the data suggests that, in the context of concept E, the condition Beak = yes is implicitly understood. A concept that includes all implicitly understood conditions is called a closed concept. Essentially, a closed

concept is the LGG of all examples that it covers. For instance, D and E both cover all positives and n5; the LGG of those six examples is Gills = no ∧ Beak = yes, which is D. Mathematically speaking we say that the closure of E is D, which is also its own closure – hence the term 'closed concept'. This doesn't mean that D and E are logically equivalent: on the contrary, since $\mathscr{X}_D \subset \mathscr{X}_E$ – the extension of D is a proper subset of the extension of E – there exist instances in $\mathscr{X}$ that are covered by E but not by D. However, none of these 'witnesses' are present in the data, and thus, as far as the data is concerned, D and E are indistinguishable. As can be seen in Figure 4.7, limiting attention to closed concepts can considerably reduce the hypothesis space.

Algorithm 4.4: MGConsistent(C, N) – find most general consistent specialisations of a concept.

```
Input  : concept C; negative examples N.
Output : set of concepts S.
if C doesn't cover any element from N then return {C};
S ← ∅;
for each minimal specialisation C' of C do
    S ← S ∪ MGConsistent(C', N);
end
return S
```

Figure 4.6. (left) A path in the hypothesis space of Example 4.4. Concept A covers a single positive (p3); B covers one additional positive (p5); C covers all positives except p4; D is the LGG of all five positive examples, but also covers a negative (n5), as does E. **(right)** The corresponding coverage curve.

In this section we have looked at the problem of learning a single logical expression that covers most or all positive examples and few or no negative examples. We have seen that such concepts live in a hypothesis space ordered by generality, and learning a concept can be understood as finding a good path through that hypothesis space. Such a path has a natural interpretation as a ranker, which allows a connection with coverage curves and ROC curves. On the other hand, insisting on a single conjunction of feature-value literals is a strong limitation; in the next section we look at ways to relax it.

Figure 4.7. The hypothesis space is reduced considerably if we restrict attention to closed concepts. There are three, rather than four, complete concepts (in green), and two, rather than seven, most general consistent closed concepts (in orange). Notice that the latter are both specialisations of the LGG of the positives, and hence it is possible to select a path that includes both the LGG and amost general consistent hypothesis.

4.3 Beyond conjunctive concepts Recall from Background 4.1 that a conjunctive normal form expression (CNF) is a conjunction of disjunctions of literals, or equivalently, a conjunction of clauses. The conjunctions of literals we have looked at until now are trivially in CNF where each disjunction consists of a single literal. CNF expressions are much more expressive, particularly since literals can occur in several clauses.

We will look at an algorithm for learning Horn theories, where each clause $A \rightarrow B$ is a Horn clause, i.e., A is a conjunction of literals and B is a single literal. For ease of notation we will restrict attention to Boolean features, and write F for F = true and ¬F for F = false. In the example below we adapt the dolphins example to Boolean variables ManyTeeth (standing for Teeth = many), Gills, Short (standing for Length = 3) and Beak.

When we looked at learning conjunctive concepts, the main intuition was that uncovered positive examples led us to generalise by dropping literals from the conjunction, while covered negative examples require specialisation by adding literals. This intuition still holds if we are learning Horn theories, but now we need to think 'clauses' rather than 'literals'. Thus, if a Horn theory doesn't cover a positive we need to drop all clauses that violate the positive, where a clause $A \rightarrow B$ violates a positive if all literals in the conjunction A are true in the example, and B is false.

Things get more interesting if we consider covered negatives, since then we need to find one or more clauses to add to the theory in order to exclude the negative. For example, suppose that our current hypothesis covers the negative ManyTeeth ∧ Gills ∧ Short ∧ ¬Beak To exclude it, we can add the following Horn clause to our theory: ManyTeeth ∧ Gills ∧ Short → Beak While there are other clauses that can exclude the negative (e.g., ManyTeeth → Beak) this is the most specific one, and hence least at risk of also excluding covered positives. However, the most specific clause excluding a negative is only unique if the negative has exactly one literal set to false. For example, if our covered negative is ManyTeeth ∧ Gills ∧ ¬Short ∧ ¬Beak then we have a choice between the following two Horn clauses: ManyTeeth ∧ Gills → Short ManyTeeth ∧ Gills → Beak Notice that, the fewer literals are set to true in the negative example, the more general the clauses excluding the negative are.

The approach of Algorithm 4.5 is to add *all* of these clauses to the hypothesis.

However, the algorithm applies two clever tricks. The first is that it maintains a list S of negative examples, from which it periodically rebuilds the hypothesis. The second is that, rather than simply adding new negative examples to the list, it tries to find negatives with fewer literals set to true, since this will result in more general clauses. This is possible if we assume we have access to a *membership oracle* Mb which can tell us whether a particular example is a member of the concept we're learning or not. So in line 7 of the algorithm we form the *intersection* of a new negative x and an existing one $s \in S$ – i.e., an example with only those literals set to true which are true in both x and s – and pass the result z to the membership oracle to check whether it belongs to the target concept. The algorithm also assumes access to an *equivalence oracle* Eq which either tells us that our current hypothesis h is logically equivalent to the target formula f, or else produces a *counter-example* that can be either a false positive (it is covered by h but not by f) or a false negative (it is covered by f but not by h).

Algorithm 4.5: Horn(Mb, Eq) – learn a conjunction of Horn clauses from membership and equivalence oracles.

```
Input  : equivalence oracle Eq; membership oracle Mb.
Output : Horn theory h equivalent to target formula f.
1  h ← true;                          // conjunction of Horn clauses, initially empty
2  S ← ∅ ;                            // a list of negative examples, initially empty
3  while Eq(h) returns counter-example x do
4      if x violates at least one clause of h then        // x is a false negative
5          specialise h by removing every clause that x violates
6      else                                               // x is a false positive
7          find the first negative example s ∈ S such that (i) z = s ∩ x has fewer true
           literals than s, and (ii) Mb(z) labels it as a negative;
8          if such an example exists then replace s in S with z, else append x to the
           end of S;
9          h ← true;
10         for all s ∈ S do                                // rebuild h from S
11             p ← the conjunction of literals true in s;
12             Q ← the set of literals false in s;
13             for all q ∈ Q do h ← h ∧ (p → q);
14         end
15     end
16 end
17 return h
```

Example 4.5 (Learning a Horn theory). Suppose the target theory f is (ManyTeeth ∧ Short → Beak) ∧ (ManyTeeth ∧ Gills → Short) This theory has 12 positive examples: eight in which ManyTeeth is false; another two in which ManyTeeth is true but both Gills and Short are false; and two more in which ManyTeeth, Short and Beak are true. The negative examples, then, are nl: ManyTeeth ∧ Gills ∧ Short ∧ ¬Beak n2: ManyTeeth ∧ Gills ∧ ¬Short ∧ Beak n3: ManyTeeth ∧ Gills ∧ ¬Short ∧ ¬Beak n4: ManyTeeth ∧ ¬Gills ∧ Short ∧ ¬Beak S is initialised to the empty list and h to the empty conjunction. We call the equivalence oracle which returns a counter-example which has to be a false positive (since every example satisfies our initial hypothesis), say n1 which violates the first clause in f. There are no negative examples in S yet, so we add n1 to S (step 8 of Algorithm 4.5). We then generate a new hypothesis from S (steps 9–13): p is ManyTeeth ∧ Gills ∧ Short and Q is {Beak}, so h becomes (ManyTeeth ∧ Gills ∧ Short

→ Beak). Notice that this clause is implied by our target theory: if ManyTeeth and Gills are true then so is Short by the second clause of f; but then so is Beak by f's first clause. But we need more clauses to exclude all the negatives.

Now, suppose the next counter-example is the false positive n2. We form the intersection with n1 which was already in S to see if we can get a negative example with fewer literals set to true (step 7). The result is equal to n3 so the membership oracle will confirm this as a negative, and we replace n1 in S with n3. We then rebuild h from S which gives (p is ManyTeeth ∧ Gills and Q is {Short, Beak}) (ManyTeeth ∧ Gills → Short) ∧ (ManyTeeth ∧ Gills → Beak) Finally, assume that n4 is the next false positive returned by the equivalence oracle. The intersection with n3 on S is actually a positive example, so instead of intersecting with n3 we append n4 to S and rebuild h. This gives the previous two clauses from n3 plus the following two from n4: (ManyTeeth ∧ Short → Gills) ∧ (ManyTeeth ∧ Short → Beak) The first of this second pair will subsequently be removed by a false negative from the equivalence oracle, leading to the final theory (ManyTeeth ∧ Gills → Short) ∧ (ManyTeeth ∧ Gills → Beak) ∧ (ManyTeeth ∧ Short → Beak) which is logically equivalent (though not identical) to f.

The Horn algorithm combines a number of interesting new ideas. First, it is an *active learning* algorithm: rather than learning from a provided data set, it constructs its own training examples and asks the membership oracle to label them. Secondly, the core of the algorithm is the list of cleverly chosen negative examples, from which the hypothesis is periodically rebuilt. The intersection step is crucial here: if the algorithm just remembered negatives, the hypothesis would consist of many specific clauses. It can be shown that, in order to learn a theory consisting of m clauses and n Boolean variables, the algorithm requires $O(mn)$ equivalence queries and $O(m^2n)$ membership queries. In addition, the runtime of the algorithm is quadratic in both m and n. While this is probably prohibitive in practice, the Horn algorithm can be shown to always successfully learn a Horn theory that is equivalent to the target theory. Furthermore, if we don't have access to an equivalence oracle the algorithm is still guaranteed to 'almost always' learn a Horn theory that is 'mostly correct'. This will be made more precise in Section 4.4.

Using first-order logic Another way to move beyond conjunctive concepts defined by simple features is to use a richer logical language. The languages we have been using so far are propositional: each literal is a proposition such as Gills = yes – standing for 'the dolphin has gills' – from which larger expressions are built using logical connectives. First-order predicate logic, or first-order logic for short, generalises this by building more complex literals from predicates and terms. For example, a first-order literal could be BodyPart(Dolphin42, PairOf(Gill)). Here, Dolphin42 and PairOf(Gill) are terms referring to objects: Dolphin42 is a constant, and PairOf(Gill) is a compound term consisting of the function symbol PairOf and the term Gills. Body Part is a binary predicate forming a proposition (something that can be true or false) out of two terms. This richer language brings with it a number of advantages:

- we can use terms such as Dolphin42 to refer to individual objects we're interested in;
- the structure of objects can be explicitly described; and
- we can introduce variables to refer to unspecified objects and quantify over them.

To illustrate the latter point, the first-order literal BodyPart(x, PairOf(Gill)) can be used to refer to the set of all objects having a pair of gills; and the following expression applies universal quantification to state that everything with a pair of gills is a fish:

$\forall x : \text{BodyPart}(x, \text{PairOf}(\text{Gill})) \rightarrow \text{Fish}(x)$ Since we modified the structure of literals, we need to revisit notions such as generalisation and LGG. Remember that for propositional literals with internal disjunction we used the function Combine-ID for merging two internal disjunctions: thus, for example, LGG-Conj-ID(Length = [3, 4], Length = [4, 5]) returns Length = [3, 4, 5]. In order to

generalise first-order literals we use variables. Consider, for example, the two first-order literals BodyPart(Dolphin42, PairOf(Gill)) and BodyPart(Human123, PairOf(Leg)): these generalise to BodyPart(x, PairOf (y)), signifying the set of objects that have a pair of some unspecified body part. There is a well-defined algorithm for computing LGGs of first-order literals called *anti-unification*, as it is the mathematical dual of the deductive operation of *unification*.

Example 4.6 (Unification and anti-unification). Consider the following terms:

BodyPart(x, PairOf (Gill))	describing the objects that have a pair of gills;
BodyPart(Dolphin42, PairOf(y))	describing the body parts that Dolphin42 has a pair of.

The following two terms are their unification and anti-unification, respectively:

BodyPart(Dolphin42, PairOf (Gill))	describing Dolphin42 as having a pair of gills;
BodyPart(x, PairOf (y))	describing the objects that have a pair of unspecified body parts.

So we see that in first-order logic literals already have quite a rich structure, owing to the use of variables. We will revisit this in Section 6.4 when we discuss how to learn classification rules in first-order logic.

4.4 Learnability In this chapter we have seen several hypothesis languages for concept

learning, including conjunctions of literals (possibly with internal disjunction), conjunctions of Horn clauses, and clauses in first-order logic. It is intuitively clear that these languages differ in expressivity: for example, a conjunction of literals is also a conjunction of Horn clauses with empty if-part, so Horn theories are strictly more expressive than conjunctive concepts. The downside of a more expressive concept language is that it may be harder to learn. The field of computational learning theory studies exactly this question of *learnability*.

To kick things off we need a *learning model*: a clear statement of what we mean if we say that a concept language is learnable. One of the most common learning models is the model of *probably approximately correct* (*PAC*) learning. PAC-learnability means that there exists a learning algorithm that gets it mostly right, most of the time. The model makes an allowance for mistakes on non-typical examples: hence the 'mostly right' or 'approximately correct'. The model also makes an allowance for sometimes getting it completely wrong, for example when the training data contains lots of non-typical examples: hence the 'most of the time' or 'probably'. We assume that typicality of examples is determined by some unspecified probability distribution D, and we evaluate the error rate err_D of a hypothesis with respect to this distribution D. More formally, for arbitrary allowable error rate $\in < 1/2$ and failure rate $\delta < 1/2$ we require a PAC-learning algorithm to output with probability at least $1 - \delta$ a hypothesis h such that $err_D < \in$.

Let's assume for the moment that our data is noise-free, and that the target hypothesis is chosen from our hypothesis language. Furthermore, we assume our learner always outputs a hypothesis that is complete and consistent with the training sample. There is a possibility that this zero training error is misleading, and that the hypothesis is actually a 'bad' one, having a true error over the

instance space that is larger than $\in$. We just want to make sure that this happens with probability less than δ. I will now show that this can be guaranteed by choosing the training sample large enough. Suppose our hypothesis space H contains a single bad hypothesis, then the probability it is complete and consistent on m independently sampled training examples is at most $(1 - \in)^m$. Since $1 - \in \leq e^{-\in}$ for any $0 \leq \in \leq 1$, we have that this probability is at most $e^{-m\in}$. We want this to be at most δ, which can be achieved by setting $m \geq \frac{1}{\epsilon} \ln \frac{1}{\delta}$. Now, H may contain several bad hypotheses, say $k \leq |H|$; then the probability that at least one of them is complete and consistent on m independently sampled training examples is at most $k(1 - \in)^m \leq |H|(1 - \in)^m \leq |H|e^{-m\in}$, which is at most δ if $m \geq \frac{1}{\epsilon}\left(\ln|H| + \ln\frac{1}{\delta}\right)$ (4.1)

This is called the *sample complexity* of a complete and consistent learner. The good news is that it is linear in $1/\in$ and logarithmic in $1/\delta$. Notice that this suggests that it is exponentially cheaper to reduce the failure rate than it is to reduce the error. Any learning algorithm that takes time polynomial in $1/\in$ and $1/\delta$ to process a single training example will therefore also take polynomial training time, another requirement for PAC-learnability. However, finding a complete and consistent hypothesis is not tractable in many hypothesis languages.

Notice that the term $\ln |H|$ arose because in the worst case almost all hypotheses in H are bad. However, in practice this means that the bound in Equation 4.1 is overly pessimistic. Still, it allows us to see that concept languages whose size is exponential in some parameter n are PAC-learnable. For example, the number of conjunctions over n Boolean variables is 3^n , since each variable can occur unnegated, negated or not at all. Consequently, the sample complexity is $(1/\in)$ $(n \ln 3 + \ln(1/\delta))$. For example, if we set $\delta = 0.05$ and $\in = 0.1$ then the sample complexity is approximately $10(n \cdot 1.1 + 3) = 11\ n + 30$. For our dolphin example with $n = 4$ this is clearly pessimistic, since there are only $2^4 = 16$ distinct examples! For larger n this is more realistic. Notice also that the PAC model is distribution-free: the learner is not given any information about the instance distribution D. This is another source for pessimism in the bound on the sample complexity.

We may not always be able to output a complete and consistent hypothesis: for instance, this may be computationally intractable, the target hypothesis may not be representable in our hypothesis language, or the examples may be noisy. A reasonable strategy would be to choose the hypothesis with lowest training error. A 'bad' hypothesis is then one whose true error exceeds the training error by at

least $\in$. Using some results from probability theory, we find that this probability is at most $e^{-2m\in^2}$. As a result, the $1/\in$ factor in Equation 4.1 is replaced by $1/2\in^2$: for $\in = 0.1$ we thus need 5 times as many training examples compared to the previous case.

It has already been mentioned that the $|H|$ term is a weak point in the above analysis. What we need is a measure that doesn't just count the size of the hypothesis space, but rather gives its expressivity or capacity in terms of classification. Such a measure does in fact exist and is called the *VC-dimension* after its inventors Vladimir Vapnik and Alexey Chervonenkis. We will illustrate the main idea by means of an example.

Example 4.7 (Shattering a set of instances). Consider the following instances:

$m =$ ManyTeeth ∧ ¬Gills ∧ ¬Short ∧ ¬Beak

$g =$ ¬Many Teeth ∧ Gills ∧ ¬Short ∧ ¬Beak

$s =$ ¬Many Teeth ∧ ¬Gills ∧ Short ∧ ¬Beak

$b =$ ¬Many Teeth ∧ ¬Gills ∧ ¬Short ∧ Beak

There are 16 different subsets of the set $\{m, g, s, b\}$. Can each of them be represented by its own conjunctive concept? The answer is yes: for every instance we want to exclude, we add the corresponding negated literal to the conjunction. Thus, $\{m, s\}$ is represented by ¬Gills ∧ ¬Beak, $\{g, s, b\}$ is represented by ¬ManyTeeth, $\{s\}$ is represented by ¬Many Teeth ∧ ¬Gills ∧ ¬Beak, and so on. We say that this set of four instances is *shattered* by the hypothesis language of conjunctive concepts.

The VC-dimension is the size of the largest set of instances that can be shattered by a particular hypothesis language or model class. The previous example shows that the VC-dimension of conjunctive concepts over d Boolean literals is at least d. It is in fact equal to d, although this is harder to prove (since it involves showing that no set of $d + 1$ instances can be shattered). This measures the capacity of the model class for representing concepts or binary

classifiers. As another example, the VC-dimension of a linear classifier in d dimensions is $d + 1$: a threshold on the real line can shatter two points but not three (since the middle point cannot be separated from the other two by a single threshold); a straight line in a two-dimensional space can shatter three points but not four; and so on.

The VC-dimension can be used to bound the difference between sample error and true error of a hypothesis (which is the step where $|H|$ appeared in our previous arguments). Consequently, it can also be used to derive a bound on the sample complexity of a complete and consistent learner in terms of the VC-dimension D rather than $|H|$:

$$m \geq \frac{1}{\epsilon} \max\left(8D\log_2 \frac{13}{\epsilon}, 4\log_2 \frac{2}{\delta}\right) \tag{4.2}$$

We see that the bound is linear in D, where previously it was logarithmic in $|H|$. This is natural, since to shatter D points we need at least 2^D hypotheses, and so $\log_2 |H| \geq D$. Furthermore, it is still logarithmic in $1/\delta$, but linear times logarithmic in $1/\in$. Plugging in our previous values of $\delta = 0.05$ and $\in = 0.1$, we obtain a sample complexity of max(562 · D, 2l3).

We conclude that the VC-dimension allows us to derive the sample complexity of infinite concept classes, as long as they have finite VC-dimension. It is furthermore worth mentioning a classical result from computational learning theory which says that a concept class is PAC-learnable if and only if its VC-dimension is finite.

4.5 Concept learning: Summary and further reading In this chapter we looked at methods for inductive concept learning: the process of constructing a logical expression defining a set of objects from examples. This problem was a focus of early work in artificial intelligence (Winston, 1970; Vere, 1975; Banerji, 1980), following the seminal work by psychologists Bruner, Goodnow and Austin (1956) and Hunt,

Marin and Stone (1966).

- In Section 4.1 we considered the structure of the hypothesis space: the set of possible concepts. Every hypothesis has an extension (the set of instances it covers), and thus relationships between extensions such as subset relationships carry over to the hypothesis space. This gives the hypothesis space a lattice structure: a partial order with least upper bounds and greatest lower bounds. In particular, the LGG is the least upper bound of a set of instances, and is the most conservative generalisation that we can learn from the data. The concept was defined in the context of first-order logic by Plotkin (1971), who showed that it was the mathematical dual of the deductive operation of unification. We can extend the hypothesis language with internal disjunction among values of a feature, which creates a larger hypothesis space that still has a lattice structure. Internal disjunction is a common staple of attribute-value languages for learning following the work of Michalski (1973). For further pointers regarding hypothesis language and hypothesis space the reader is referred to (Blockeel, 2010*a*, *b*).

- Section 4.2 defined complete and consistent hypotheses as concepts that cover all positive examples and no negative examples. The set of complete and consistent concepts is called the version space, a notion introduced by Mitchell (1977). The version space can be summarised by its least general and most general members, since any concept between one least general hypothesis and another most general one is also complete and consistent. Alternatively, we can describe the version space by all paths from a least general to a most general hypothesis. Such upward paths give rise to a coverage curve which describes the extension of each concept on the path in terms of covered positives and negatives. Concept learning can then be seen as finding an upward path that goes through ROC heaven. Syntactically different concepts can have the same extension in a particular data set: a closed concept is the most specific one of these (technically, the LGG of the instances in its extension). The notion is studied in formal concept analysis (Ganter and Wille, 1999) and was introduced in a data mining context by Pasquier, Bastide, Taouil and Lakhal (1999); Garriga, Kralj and Lavrač (2008) investigate its usefulness for labelled data.

In Section 4.3 we discussed the Horn algorithm for learning concepts described by conjunctions of Horn rules, first published in Angluin *et al.* (1992). The algorithm makes use of a membership oracle, which can be seen as an early form of active learning (Cohn, 2010; Dasgupta, 2010). Horn theories are superficially similar to classification rule models which will be studied in Chapter 6. However, there is an important difference, since those classification rules have the target variable in the then-part of the rule, while the Horn clauses we are looking at here can have any literal in the then-part. In fact, in this chapter the target variable is not part of the logical language at all. This setting is sometimes called *learning from interpretations*, since examples are truth-value assignments to our theory. The classification rule setting is called *learning from entailment*, since in order to find out whether a particular rule covers an example we need to apply logical inference. De Raedt (1997) explains and explores the differences between these two settings. Further introductions to first-order logic and its use in learning are given by Flach (2010*a*) and De Raedt (2010).

Section 4.4 briefly reviewed some basic concepts and results in learnability theory. My account partly followed Mitchell (1997, Chapter 7);another excellent introduction is given by Zeugmann (2010). PAC-learnability, which allows an error rate of $\in$ and a failure rate of δ, was introduced in a seminal paper by Valiant (1984). Haussler (1988) derived the sample complexity for complete and consis-tentlearners (Equation 4.1), which is linear in $1/\in$ and logarithmic in $1/\delta$ andthe size of the hypothesis space. The VC-dimension as a measure of the capacity of a hypothesis language was introduced by Vapnik and Chervonenkis (197l) in order to quantify the difference between training error and true error. This allows a statement of the sample complexity in terms of the VC-dimension (Equation 4.2) which is due to Blumer, Ehrenfeucht, Haussler and Warmuth (1989). This same paper proved that a model class is PAC-learnable if and only if its VC-dimension is finite.

[1] Inspired by www.cwtstrandings.org.

CHAPTER 5

Tree models

TREE MODELS ARE among the most popular models in machine learning. For example, the pose recognition algorithm in the Kinect motion sensing device for the Xbox game console has decision tree classifiers at its heart (in fact, an ensemble of decision trees called a random forest about which you will learn more in Chapter 11). Trees are expressive and easy to understand, and of particular appeal to computer scientists due to their recursive 'divide-and-conquer' nature.

In fact, the paths through the logical hypothesis space discussed in the previous chapter already constitute a very simple kind of tree. For instance, the feature tree in Figure 5.1 (left) is equivalent to the path in Figure 4.6 (left) on p.117. This equivalence is best seen by tracing the path and the tree from the bottom upward.

1. The left-most leaf of the feature tree represents the concept at the bottom of the path, covering a single positive example.
2. The next concept up in the path generalises the literal Length = 3 into Length = [3, 5] by means of internal disjunction; the added coverage (one positive example) is represented by the second leaf from the left in the feature tree.
3. By dropping the condition Teeth = few we add another two covered positives.
4. Dropping the 'Length' condition altogether (or extending the internal disjunction with the one remaining value '4') adds the last positive, and also a negative.
5. Dropping Beak = yes covers no additional examples (remember the discussion about closed concepts in the previous chapter).
6. Finally, dropping Gills = no covers the four remaining negatives.

Figure 5.1. (left) The path from Figure 4.6 on p.117, redrawn in the form of a tree. The coverage numbers in the leaves are obtained from the data in Example 4.4. (**right**) A decision tree learned on the same data. This tree separates the positives and negatives perfectly.

We see that a path through the hypothesis space can be turned into an equivalent feature tree. To obtain a tree that is equivalent to the *i*-th concept from the bottom in the path, we can either truncate the tree by combining the left-most *i* leaves into a single leaf representing the concept; or we can label the left-most *i* leaves positive and the remaining leaves negative, turning the feature tree into a decision tree.

Decision trees do not employ internal disjunction for features with more than two values, but instead allow branching on each separate value. They also allow leaf labellings that do not follow the left-to-right order of the leaves. Such a tree is shown in Figure 5.1 (right). This tree can be turned into a logical expression in many different ways, including: (Gills = no ∧ Length = 3) ∨ (Gills = no ∧ Length = 4 ∧ Teeth = many) ∨(Gills = no ∧ Length = 5) Gills = no ∧ [Length = 3 ∨ (Length = 4 ∧ Teeth = many) ∨ Length = 5]

¬ [(Gills = no ∧ Length = 4 ∧ Teeth = few) ∨ Gills = yes]

(Gills = yes ∨ Length = [3, 5] ∨ Teeth = many) ∧ Gills = no The first expression is in disjunctive normal form (DNF, see Background 4.1 on p.105) and is obtained by forming a disjunction of all paths from the root of the tree to leaves labelled positive, where each path gives a conjunction of literals. The second expression is a simplification of the first using the distributive equivalence $(A \wedge B) \vee (A \wedge C) \equiv A \wedge (B \vee C)$. The third expression is obtained by first forming a

DNF expression representing the negative class, and then negating it. The fourth expression turns this into CNF by using the De Morgan laws $\neg(A \wedge B) \equiv \neg A \vee \neg B$ and $\neg(A \vee B) \equiv \neg A \wedge \neg B$.

There are many other logical expressions that are equivalent to the concept defined by the decision tree. Perhaps it would be possible to obtain an equivalent conjunctive concept? Interestingly, the answer to this question is no: some decision trees represent a conjunctive concept, but many trees don't and this is one of them.[1] *Decision trees are strictly more expressive than conjunctive concepts*. In fact, since decision trees correspond to DNF expressions, and since every logical expression can be equivalently written in DNF, it follows that decision trees are maximally expressive: the only data that they cannot separate is data that is inconsistently labelled, i.e., the same instance appears twice with different labels. This explains why data that isn't conjunctively separable, as in our example, can be separated by a decision tree.

There is a potential problem with using such an expressive hypothesis language. Let Δ be the disjunction of all positive examples, then Δ is in disjunctive normal form. Δ clearly covers all positives – in fact, Δ's extension is exactly the set of positive examples. In other words, in the hypothesis space of DNF expressions (or of decision trees), Δ is the LGG of the positive examples, but it doesn't cover any other instances. So Δ does not generalise beyond the positive examples, but merely memorises them – talk about overfitting! Turning this argument around, we see that *one way to avoid overfitting and encourage learning is to deliberately choose a restrictive hypothesis language*, such as conjunctive concepts: in such a language, even the LGG operation typically generalises beyond the positive examples. And if our language is expressive enough to represent any set of positive examples, we must make sure that the learning algorithm employs other mechanisms to force generalisation beyond the examples and avoid overfitting – this is called the *inductive bias* of the learning algorithm. As we will see, most learning algorithms that operate in expressive hypothesis spaces have an inductive bias towards less complex hypotheses, either implicitly through the way the hypothesis space is searched, or explicitly by incorporating a complexity penalty in the objective function.

Tree models are not limited to classification but can be employed to solve almost any machine learning task, including ranking and probability estimation, regression and clustering. The tree structure that is common to all those models can be defined as follows.

Definition 5.1 (Feature tree). *A* feature tree *is a tree such that each internal node (the nodes that are not leaves) is labelled with a feature, and each edge*

emanating from an internal node is labelled with a literal. The set of literals at a node is called a split. *Each leaf of the tree represents a logical expression, which is the conjunction of literals encountered on the path from the root of the tree to the leaf. The extension of that conjunction (the set of instances covered by it) is called the* instance space segment *associated with the leaf.*

Essentially, a feature tree is a compact way of representing a number of conjunctive concepts in the hypothesis space. The learning problem is then to decide which of the possible concepts will be best to solve the given task. While rule learners (discussed in the next chapter) essentially learn these concepts one at a time, tree learners perform a top–down search for all these concepts at once.

Algorithm 5.1 gives the generic learning procedure common to most tree learners. It assumes that the following three functions are defined: **Homogeneous**(D) returns true if the instances in D are homogeneous enough to be labelled with a single label, and false otherwise; **Label**(D) returns the most appropriate label for a set of instances D; **BestSplit**(D, F) returns the best set of literals to be put at the root of the tree.

These functions depend on the task at hand: for instance, for classification tasks a set of instances is homogeneous if they are (mostly) of a single class, and the most appropriate label would be the majority class. For clustering tasks a set of instances is homogenous if they are close together, and the most appropriate label would be some exemplar such as the mean (more on exemplars in Chapter 8).

Algorithm 5.1: GrowTree(D, F) – grow a feature tree from training data.

Input : data D; set of features F.
Output : feature tree T with labelled leaves.

```
if Homogeneous(D) then return Label(D) ;    // Homogeneous, Label: see text
S ← BestSplit(D, F);                        //e.g., BestSplit-Class (Algorithm 5.2)
split D into subsets D_i according to the literals in S;
for each i do
    if D_i ≠ ø then T_i ← GrowTree(D_i, F) else T_i is a leaf labelled with Label(D);
end
return a tree whose root is labelled with S and whose children are T_i
```

Algorithm 5.1 is a *divide-and-conquer* algorithm: it divides the data into subsets, builds a tree for each of those and then combines those subtrees into a single tree. Divide-and-conquer algorithms are a tried-and-tested technique in computer science. They are usually implemented recursively, because each subproblem (to build a tree for a subset of the data) is of the same form as the original problem. This works as long as there is a way to stop the recursion, which is what the first line of the algorithm does. However, it should be noted that such algorithms are *greedy*: whenever there is a choice (such as choosing the best split), the best alternative is selected on the basis of the information then available, and this choice is never reconsidered. This may lead to sub-optimal choices. An alternative would be to use a *backtracking search* algorithm, which can return an optimal solution, at the expense of increased computation time and memory requirements, but we will not explore that further in this book.

In the remainder of this chapter we will instantiate the generic Algorithm 5.1 to classification, ranking and probability estimation, clustering and regression tasks.

5.1 Decision trees

As already indicated, for a classification task we can simply define a set of instances *D* to be homogenous if they are all from the same class, and the function Label(*D*) will then obviously return that class. Notice that in line 5 of Algorithm 5.1 we may be calling Label(*D*) with a non-homogeneous set of instances in case one of the D_i is empty, so the general definition of Label(*D*) is that it returns the majority class of the instances in *D*.[2] This leaves us to decide how to define the function BestSplit(*D, F*).

Let's assume for the moment that we are dealing with Boolean features, so *D* is

split into D_1 and D_2. Let's also assume we have two classes, and denote by $D^{\oplus}$ and $D^{\ominus}$ the positives and negatives in D (and likewise for $D_1^{\oplus}$ etc.). The question is how to assess the utility of a feature in terms of splitting the examples into positives and negatives. Clearly, the best situation is where $D_1^{\oplus} = D^{\oplus}$ and $D_1^{\ominus} = \emptyset$, or where $D_1^{\oplus} = \emptyset$ and $D_1^{\ominus} = D^{\ominus}$. In that case, the two children of the split are said to be *pure*. So we need to measure the impurity of a set of $n^{\oplus}$ positives and $n^{\ominus}$ negatives. One important principle that we will adhere to is that the impurity should only depend on the relative magnitude of $n^{\oplus}$ and $n^{\ominus}$, and should not change if we multiply both with the same amount. This in turn means that impurity can be defined in terms of the proportion $\dot{p} = n^{\oplus}/(n^{\oplus} + n^{\ominus})$, which we remember from Section 2.2 as the ☞*empirical probability* of the positive class. Furthermore, impurity should not change if we swap the positive and negative class, which means that it should stay the same if we replace $\dot{p}$ with $1 - \dot{p}$. We also want a function that is 0 whenever $\dot{p} = 0$ or $\dot{p} = 1$ and that reaches its maximum for $\dot{p} = 1/2$. The following functions all fit the bill.

Figure 5.2. (left) Impurity functions plotted against the empirical probability of the positive class. From the bottom: the relative size of the minority class, $\min(\dot{p}, 1 - \dot{p})$; the Gini index, $2\dot{p}(1 - \dot{p})$; entropy, $-\dot{p}\log_2 \dot{p} - (1 - \dot{p})\log_2(1 - \dot{p})$ (divided by 2 so that it reaches its maximum in the same point as the others); and the (rescaled) square root of the Gini index, $\sqrt{\dot{p}(1-\dot{p})}$ – notice that this last function describes a semi-circle. **(right)** Geometric construction to determine the impurity of a split (Teeth = [many, few] from Example 5.1): $\dot{p}$ is the empirical probability of the parent, and $\dot{p}_1$ and $\dot{p}_2$ are the empirical probabilities of the children.

Minority class $\min(\dot{p}, 1 - \dot{p})$ – this is sometimes referred to as the error rate, as it measures the proportion of misclassified examples if the leaf was labelled with the majority class; the purer the set of examples, the fewer errors this will make. This impurity measure can equivalently be written as $1/2 - |\dot{p} - 1/2|$.

Gini index $2\dot{p}(1 - \dot{p})$ – this is the expected error if we label examples in the leaf randomly: positive with probability $\dot{p}$ and negative with probability $1 - \dot{p}$. The probability of a false positive is then $\dot{p}(1 - \dot{p})$ and the probability of a false negative $(1 - \dot{p})\dot{p}$.[3]

entropy – $\dot{p}\log_2 \dot{p} - (1 - \dot{p}) \log_2(1 - \dot{p})$ – this is the expected information, in bits, conveyed by somebody telling you the class of a randomly drawn example; the purer the set of examples, the more predictable this message becomes and the smaller the expected information.

A plot of these three impurity measures can be seen in Figure 5.2 (left), some of them rescaled so that they all reach their maximum at (0.5, 0.5). I have added a fourth one: the square root of the Gini index, which I will indicate as $\sqrt{\text{Gini}}$, and which has an advantage over the others, as we will see later. Indicating the impurity of a single leaf D_j as $\text{Imp}(D_j)$, the impurity of a set of mutually exclusive leaves $\{D_1, \ldots, D_1\}$ is then defined as a weighted average

$$\text{Imp}(\{D_1, \ldots, D_l\}) = \sum_{j=1}^{l} \frac{|D_j|}{|D|} \text{Imp}(D_j) \tag{5.1}$$

where $D = D_1 \cup \ldots \cup D_1$. For a binary split there is a nice geometric construction to find $\text{Imp}(\{D_1, D_2\})$ given the empirical probabilities of the parent and the children, which is illustrated in Figure 5.2 (right):

1. We first find the impurity values $\text{Imp}(D_1)$ and $\text{Imp}(D_2)$ of the two children on the impurity curve (here the Gini index).
2. We then connect these two values by a straight line, as any weighted average of the two must be on that line.
3. Since the empirical probability of the parent is also a weighted average of the empirical probabilities of the children, with the same weights (i.e., $\dot{p} = \frac{|D_1|}{|D|}\dot{p}_1 + \frac{|D_2|}{|D|}\dot{p}_2$ – the derivation is given in Equation 5.2 on p.139), $\dot{p}$ gives us the correct interpolation point.

This construction will work with any of the impurity measures plotted in Figure 5.2 (left). Note that, if the class distribution in the parent is very skewed, the empirical probability of both childrenmay end up to the left or to the right of the $\dot{p} = 0.5$ vertical. This isn't a problem – except for the minority class impurity measure, as the geometric construction makes it clear that all such splits will be evaluated as having the same weighted average impurity. For this reason its use as an impurity measure is often discouraged.

Example 5.1 (Calculating impurity). Consider again the data in Example

4.4 on p.115. We want to find the best feature to put at the root of the decision tree. The four features available result in the following splits:

Length = [3, 4, 5]	[2+, 0−][1+, 3−][2+, 2−]
Gills = [yes, no]	[0+, 4−][5+, 1−]
Beak = [yes, no]	[5+, 3−][0+, 2−]
Teeth = [many, few]	[3+, 4−][2+, 1−]

Let's calculate the impurity of the first split. We have three segments: the first one is pure and so has entropy 0; the second one has entropy $-(1/4)\log_2(1/4)-(3/4)\log_2(3/4) = 0.5 + 0.31 = 0.81$; the third one has entropy 1. The total entropy is then the weighted average of these, which is $2/10 \cdot 0+4/10 \cdot 0.81+4/10 \cdot 1 = 0.72$.

Similar calculations for the other three features give the following entropies:

Gills $4/10 \cdot 0 + 6/10 \cdot (-(5/6)\log_2(5/6) - (1/6)\log_2(1/6)) = 0.39;$

Beak $8/10 \cdot (-(5/8)\log_2(5/8)-(3/8)\log_2(3/8))+2/10 \cdot 0 = 0.76;$

Teeth $7/10 \cdot (-(3/7)\log_2(3/7)-(4/7)\log_2(4/7))$
$+3/10 \cdot (-(2/3)\log_2(2/3)-(1/3)\log_2(1/3)) = 0.97.$

We thus clearly see that 'Gills' is an excellent feature to split on; 'Teeth' is poor; and the other two are somewhere in between.

The calculations for the Gini index are as follows (notice that these are on a scale from 0 to 0.5):

Length $2/10 \cdot 2 \cdot (2/2 \cdot 0/2) + 4/10 \cdot 2 \cdot (1/4 \cdot 3/4) + 4/10 \cdot 2 \cdot (2/4 \cdot 2/4) = 0.35;$

Gills $4/10 \cdot 0 + 6/10 \cdot 2 \cdot (5/6 \cdot 1/6) = 0.17;$

Beak $8/10 \cdot 2 \cdot (5/8 \cdot 3/8) + 2/10 \cdot 0 = 0.38;$

Teeth $7/10 \cdot 2 \cdot (3/7 \cdot 4/7) + 3/10 \cdot 2 \cdot (2/3 \cdot 1/3) = 0.48$.

As expected, the two impurity measures are in close agreement. See Figure 5.2 (right) for a geometric illustration of the last calculation concerning 'Teeth'.

Adapting these impurity measures to $k > 2$ classes is done by summing the per-class impurities in a one-versus-rest manner. In particular, k-class entropy is defined as $\sum_{i=1}^{k} -\dot{p}_i \log_2 \dot{p}_i$, and the k-class Gini index as $\sum_{i=1}^{k} \dot{p}_i(1-\dot{p}_i)$. In assessing the quality of a feature for splitting a parent node D into leaves $D_1, \ldots, D_l$, it is customary to look at the purity gain $\mathrm{Imp}(D) - \mathrm{Imp}(\{D_1, \ldots, D_l\})$. If purity is measured by entropy, this is called the *information gain* splitting criterion, as it measures the increase in information about the class gained by including the feature. However, note that Algorithm 5.1 only compares splits with the same parent, and so we can ignore the impurity of the parent and search for the feature which results in the lowest weighted average impurity of its children (Algorithm 5.2).

We now have a fully instantiated decision tree learning algorithm, so let's see what tree it learns on our dolphin data. We have already seen that the best feature to split on at the root of the tree is 'Gills': the condition Gills = yes leads to a pure leaf [0+, 4−] labelled negative, and a predominantly positive child [5+, 1−]. For the next split we have the choice between 'Length' and 'Teeth', as splitting on 'Beak' does not decrease the impurity. 'Length' results in a [2+, 0−][1+, 1−][2+, 0−] split and 'Teeth' in a [3+, 0−][2+, 1−] split; both entropy and Gini index consider the former purer than the latter. We then use 'Teeth' to split the one remaining impure node. The resulting tree is the one shown previously in Figure 5.1 on p.130, and reproduced in Figure 5.3 (left). We have learned our first decision tree!

Figure 5.3. (left) Decision tree learned from the data in Example 4.4 on p.115. **(right)** Each internal and leaf node of the tree corresponds to a line segment in coverage space: vertical segments for pure positive nodes, horizontal segments for pure negative nodes, and diagonal segments for impure nodes.

Algorithm 5.2: BestSplit-Class(D, F) – find the best split for a decision tree.

Input : data D; set of features F.
Output : feature f to split on.

```
I_min ← 1;
for each f ∈ F do
    split D into subsets D_1, ..., D_l according to the values v_f of f;
    if Imp({D_1, ..., D_l}) < I_min then
        I_min ← Imp({D_1, ..., D_l});
        f_best ← f;
    end
end
return f_best
```

The tree represents a partition of the instance space, and therefore also assigns a class to the 14 instances that were not part of the training set – which is why we can say that the tree generalises the training data. Leaf C leaves three feature values unspecified, with a total of 3·2·2 = 12 possible combinations of values; four of these were supplied as training examples, so leaf C covers eight unlabelled instances and classifies them as negative. Similarly, two unlabelled instances are classified as positive by leaf D, and a further two by leaf F; one is classified as negative by leaf G, and the remaining one as positive by leaf H. The fact that more unlabelled instances are classified as negative (9) than as positive

(5) is thus mostly due to leaf C: because it is a leaf high up in the tree, it covers many instances. One could argue that the fact that four out of five negatives have gills is the strongest regularity found in the data.

It is also worth tracing the construction of this tree in coverage space (Figure 5.3 (right)). Every node of the tree, internal or leaf, covers a certain number of positives and negatives and hence can be plotted as a line segment in coverage space. For instance, the root of the tree covers all positives and all negatives, and hence is represented by the ascending diagonal A. Once we add our first split, segment A is replaced by segment B (an impure node and hence diagonal) and segment C, which is pure and not split any further. Segment B is further split into D (pure and positive), E (impure) and F (pure and positive). Finally, E is split into two pure nodes.

This idea of a decision tree coverage curve 'pulling itself up' from the ascending diagonal in a divide-and-conquer fashion is appealing – but unfortunately it is not true in general. The ordering of coverage curve segments is purely based on the class distributions in the leaves and does not bear any direct relationship to the tree structure. To understand this better, we will now look at how tree models can be turned into rankers and probability estimators.

5.2 Ranking and probability estimation trees

Grouping classifiers such as decision trees divide the instance space into segments, and so can be turned into rankers by learning an ordering on those segments. Unlike some other grouping models, decision trees have access to the local class distributions in the segments or leaves, which can directly be used to construct a leaf ordering that is optimal for the training data. So, for instance, in Figure 5.3 this ordering is [D – F] – H – G – C, resulting in a perfect ranking (AUC = 1). The

ordering can simply be obtained from the empirical probabilities $\dot{p}$, breaking ties as much as possible by giving precedence to leaves covering a larger number of positives.[4] Why is this ordering optimal? Well, the slope of a coverage curve segment with empirical probability $\dot{p}$ is $\dot{p}/(1 - \dot{p})$; since $\dot{p} \mapsto \frac{\dot{p}}{1-\dot{p}}$ is a monotonic transformation (if $\dot{p} > \dot{p}'$ then $\frac{\dot{p}}{1-\dot{p}} > \frac{\dot{p}'}{1-\dot{p}'}$) sorting the segments on non-increasing empirical probabilities ensures that they are also sorted on non-increasing slope, and so the curve is convex. This is an important point, so I'll say it again: *the ranking obtained from the empirical probabilities in the leaves of a decision tree yields a convex ROC curve on the training data*. As we shall see later in the book, some other grouping models including ☞*rule lists* (Section 6.1) share this property, but no grading model does.

As already noted, the segment ordering cannot be deduced from the tree structure. The reason is essentially that, even if we know the empirical probability associated with the parent of a split, this doesn't constrain the empirical probabilities of its children. For instance, let $[n^{\oplus}, n^{\ominus}]$ be the class distribution in the parent with $n = n^{\oplus} + n^{\ominus}$, and let $[n_1^{\oplus}, n_1^{\ominus}]$ and $[n_2^{\oplus}, n_2^{\ominus}]$ be the class distributions in the children, with $n_1 = n_1^{\oplus} + n_1^{\ominus}$ and $n_2 = n_2^{\oplus} + n_2^{\ominus}$. We then

have $\dot{p} = \frac{n^{\oplus}}{n} = \frac{n_1}{n}\frac{n_1^{\oplus}}{n_1} + \frac{n_2}{n}\frac{n_2^{\oplus}}{n_2} = \frac{n_1}{n}\dot{p}_1 + \frac{n_2}{n}\dot{p}_2$ (5.2)

In other words, the empirical probability of the parent is a weighted average of the empirical probabilities of its children; but this only tells us that $\dot{p}_1 \leq \dot{p} \leq \dot{p}_2$ or $\dot{p}_2 \leq \dot{p} \leq \dot{p}_1$. Even if the place of the parent segment in the coverage curve is known, its children may come much earlier or later in the ordering.

Example 5.2 (Growing a tree). Consider the tree in Figure 5.4 (top). Each node is labelled with the numbers of positive and negative examples covered by it: so, for instance, the root of the tree is labelled with the overall class distribution (50 positives and 100 negatives), resulting in the trivial ranking [50+, 100−]. The corresponding one-segment coverage curve is the ascending diagonal (Figure 5.4 (bottom)). Adding split (1) refines this ranking into [30+, 35−][20+, 65−], resulting in a two-segment curve. Adding splits (2) and (3) again breaks up the segment corresponding to the parent into two segments corresponding to the children. However, the ranking produced by the full tree – [15+, 3−][29+, 10−][5+, 62−][1+, 25−] – is different from the left-to-right ordering of its leaves, hence we need to reorder the segments of the coverage curve, leading to the top-most, solid curve.

So, adding a split to a decision tree can be interpreted in terms of coverage curves as the following two-step process:

- split the corresponding curve segment into two or more segments;
- reorder the segments on decreasing slope.

The whole process of growing a decision tree can be understood as an iteration of these two steps; or alternatively as a sequence of splitting steps followed by one overall reordering step. It is this last step that guarantees that the coverage curve is convex (on the training data).

Figure 5.4. (top) Abstract representation of a tree with numbers of positive and negative examples covered in each node. Binary splits are added to the tree in the order indicated. **(bottom)** Adding a split to the tree will add new segments to the coverage curve as indicated by the arrows. After a split is added the segments may need reordering, and so only the solid lines represent actual coverage curves.

It is instructive to take this analysis a step further by considering all possible rankings that can be constructed with the given tree. One way to do that is to consider the tree as a feature tree, without any class labels, and ask ourselves in how many ways we can label the tree, and what performance that would yield, given that we know the numbers of positives and negatives covered in each leaf. In general, if a feature tree has l leaves and we have c classes, then the number of possible labellings of leaves with classes is c^l; in the example of Figure 5.4 this is $2^4 = 16$. Figure 5.5 depicts these 16 labellings in coverage space. As you might expect, there is a lot of symmetry in this plot. For instance, labellings occur in pairs (say + − + − and − + − +) that occur in opposite locations in the plot (see if you can figure out what is meant by 'opposite' here). We obtain a ranking by starting in −−−− in the lower left-hand corner, and switching each

leaf to + in some order. For instance, the optimal coverage curve follows the order , −−−−, −−+−, +−+−, ++++ . For a tree with l leaves there are $l!$ permutations of its leaves and thus $l!$ possible coverage curves (24 in our example).

Figure 5.5. Graphical depiction of all possible labellings and all possible rankings that can be obtained with the four-leaf decision tree in Figure 5.4. There are $2^4 = 16$ possible leaf labellings; e.g., '+−+−' denotes labelling the first and third leaf from the left as + and the second and fourth leaf as −. Also indicated are some pairwise symmetries (dotted lines): e.g., +−+− and −+−+ are each other's inverse and end up at opposite ends of the plot. There are 4! = 24 possible blue-violet-red-orange paths through these points which start in −−−− and switch each leaf to + in some order; these represent all possible four-segment coverage curves or rankings.

If I were to choose a single image that would convey the essence of tree models, it would be Figure 5.5. What it visualises is that the class distributions in the leaves of an unlabelled feature tree can be used to turn one and the same tree into a decision tree, a ranking tree, or a probability estimation tree:

- to turn a feature tree into a ranker, we order its leaves on non-increasing empirical probabilities, which is provably optimal on the training set;
- to turn the tree into a probability estimator, we predict the empirical probabilities in each leaf, applying Laplace or m-estimate smoothing to make these estimates more robust for small leaves;
- to turn the tree into a classifier, we choose the operating conditions and fir the

The last procedure was explained in Section 2.2. We will illustrate it here,

assuming the training set class ratio $clr = 50/100$ is representative. We have a choice of five labellings, depending on the expected cost ratio $c = C_{FN}/C_{FP}$ of misclassifying a positive in proportion to the cost of misclassifying a negative:

$+-+-$ would be the labelling of choice if $c = 1$, or more generally if $10/29 < c < 62/5$;

$+-++$ would be chosen if $62/5 < c < 25/1$;

$++++$ would be chosen if $25/1 < c$; i.e., we would always predict positive if false negatives are more than 25 times as costly as false positives, because then even predicting positive in the second leaf would reduce cost;

$--+-$ would be chosen if $3/15 < c < 10/29$;

$----$ would be chosen if $c < 3/15$; i.e., we would always predict negative if false positives are more than 5 times as costly as false negatives, because then even predicting negative in the third leaf would reduce cost.

The first of these options corresponds to the majority class labelling, which is what most textbook treatments of decision trees recommend, and also what I suggested when I discussed the function Label(D) in the context of Algorithm 5.1. In many circumstances this will indeed be the most practical thing to do. However, it is important to be aware of the underlying assumptions of such a labelling: these assumptions are that the training set class distribution is representative and the costs are uniform; or, more generally, that the product of the expected cost and class ratios is equal to the class ratio as observed in the training set. (This actually suggests a useful device for manipulating the training set to reflect an expected class ratio: to mimic an expected class ratio of c, we can oversample the positive training examples with a factor c if $c > 1$, or oversample the negatives with a factor $1/c$ if $c < 1$. We will return to this suggestion below.) So let's assume that the class distribution is representative and that false negatives (e.g., not diagnosing a disease in a patient) are about 20 times more costly than false positives. As we have just seen, the optimal labelling under these operating conditions is $+-++$, which means that we only use the second leaf to filter out negatives. In other words, the right two leaves can be merged into one – their parent. Rather aptly, the operation of merging all leaves in a subtree is called *pruning* the subtree. The process is illustrated in Figure 5.6. The advantage of pruning is that we can simplify the tree without affecting the chosen operating point, which is sometimes useful if we want to

communicate the tree model to somebody else. The disadvantage is that we lose ranking performance, as illustrated in Figure 5.6 (bottom). Pruning is therefore not recommended unless (*i*) you only intend to use the tree for classification, not for ranking or probability estimation; and (*ii*) you can define the expected operating conditions with sufficient precision. One popular algorithm for pruning decision trees is called *reduced-error pruning*, and is given in Algorithm 5.3. The algorithm employs a separate *pruning set* of labelled data not seen during training, as pruning will never improve accuracy over the training data. However, if tree simplicity is not really an issue, I recommend keeping the entire tree intact and choosing the operating point through the leaf labelling only; this can similarly be done using a hold-out data set.

Figure 5.6. (top) To achieve the labelling +−++ we don't need the right-most split, which can therefore be pruned away. **(bottom)** Pruning doesn't affect the chosen operating point, but it does decrease the ranking performance of the tree.

Sensitivity to skewed class distributions I just mentioned in passing that one way to make sure the training set reflects the right

operating conditions is to duplicate positives or negatives so that the training set class ratio is equal to the product of expected cost and class ratios in deployment of the model. Effectively, this changes the aspect ratio of the rectangle representing the coverage space. The advantage of this method is that it is directly applicable to any model, without need to interfere with search heuristics or evaluation measures. The disadvantage is that it will increase training time – and besides, it may not actually make a difference for the model being learned. I will illustrate this with an example.

Example 5.3 (Cost-sensitivity of splitting criteria). Suppose you have 10 positives and 10 negatives, and you need to choose between the two splits [8+, 2−][2+, 8−] and [10+, 6−][0+, 4−]. You duly calculate the weighted average entropy of both splits and conclude that the first split is the better one. Just to be sure, you also calculate the average Gini index, and again the first split wins. You then remember somebody telling you that the square root of the Gini index was a better impurity measure, so you decide to check that one out as well. Lo and behold, it favours the second split…! What to do?

You then remember that mistakes on the positives are about ten times as costly as mistakes on the negatives. You're not quite sure how to work out the maths, and so you decide to simply have ten copies of every positive: the splits are now [80+, 2−] [20+, 8−] and [100+, 6−][0+, 4−]. You recalculate the three splitting criteria and now all three favour the second split. Even though you're slightly bemused by all this, you settle for the second split since all three splitting criteria are now unanimous in their recommendation.

Algorithm 5.3: PruneTree(T, D) – reduced-error pruning of a decision tree.

Input : decision tree T; labelled data D.

Output : pruned tree T'.

```
for every internal node N of T, starting from the bottom do
    T_N ← subtree of T rooted at N;
    D_N ← {x ∈ D | x is covered by N};
    if accuracy of T_N over D_N is worse than majority class in D_N then
        replace T_N in T by a leaf labelled with the majority class in D_N;
    end
end
return pruned version of T
```

So what is going on here? Let's first look at the situation with the inflated numbers of positives. Intuitively it is clear that here the second split is preferable, since one of the children is pure and the other one is fairly good as well, though perhaps not as good as [80+, 2−]. But this situation changes if we have only one-tenth of the number of positives, at least according to entropy and Gini index. Using notation introduced earlier, this can be understood as follows. The Gini index of the parent is $2\frac{n^{\oplus}}{n}\frac{n^{\ominus}}{n}$, and the weighted Gini index of one of the children is $\frac{n_1}{n}2\frac{n_1^{\oplus}}{n_1}\frac{n_1^{\ominus}}{n_1}$. So the weighted impurity of the child *in proportion* to the parent's impurity is $\frac{n_1^{\oplus}n_1^{\ominus}/n_1}{n^{\oplus}n^{\ominus}/n}$; let's call this *relative impurity*. The same calculations for $\sqrt{\text{Gini}}$ give

- impurity of the parent: $\sqrt{\frac{n^{\oplus}}{n}\frac{n^{\ominus}}{n}}$;
- weighted impurity of the child: $\frac{n_1}{n}\sqrt{\frac{n_1^{\oplus}}{n_1}\frac{n_1^{\ominus}}{n_1}}$;
- relative impurity: $\sqrt{\frac{n_1^{\oplus}n_1^{\ominus}}{n^{\oplus}n^{\ominus}}}$.

The important thing to note is that this last ratio doesn't change if we multiply all numbers involving positives with a factor c. That is, $\sqrt{\text{Gini}}$ is designed to minimise relative impurity, and thus is insensitive to changes in class distribution. In contrast, relative impurity for the Gini index includes the ratio n_1/n, which changes if we inflate the number of positives. Something similar happens with entropy. As a result, these two splitting criteria emphasise children covering more examples.

A picture will help to explain this further. Just as accuracy and average recall have isometrics in coverage and ROC space, so do splitting criteria. Owing to

their nonlinear nature, these isometrics are curved rather than straight. They also occur on either side of the diagonal, as we can swap the left and right child without changing the quality of the split. One might imagine the impurity landscape as a mountain looked down on from above – the summit is a ridge along the ascending diagonal, representing the splits where the children have the same impurity as the parent. This mountain slopes down on either side and reaches ground level in ROC heaven as well as its opposite number ('ROC hell'), as this is where impurity is zero. The isometrics are the contour lines of this mountain – walks around it at constant elevation.

Consider Figure 5.7 (top). The two splits among which you needed to choose in Example 5.3 (before inflating the positives) are indicated as points in this plot. I have drawn six isometrics in the top-left of the plot: two splits times three splitting criteria. A particular splitting criterion prefers the split whose isometric is the *highest* (closest to ROC heaven) of the two: you can see that only one of the three ($\sqrt{\text{Gini}}$) prefers the split on the top-right. Figure 5.7 (bottom) demonstrates how this changes when inflating the positives with a factor 10 (a coverage plot would run off the page here, so I have plotted this in ROC space with the grid indicating how the class distribution has changed). Now all three splitting criteria prefer the top-right split, because the entropy and Gini index 'mountains' have rotated clockwise (Gini index more so than entropy), while the $\sqrt{\text{Gini}}$ mountain hasn't moved at all.

Figure 5.7. (top) ROC isometrics for entropy in blue, Gini index in violet and $\sqrt{\text{Gini}}$ in red through the splits [8+, 2−][2+, 8−] (solid lines) and [10+, 6−][0+, 4−] (dotted lines). Only $\sqrt{\text{Gini}}$ prefers the second split. **(bottom)** The same isometrics after inflating the positives with a factor 10. All splitting criteria now favour the second split; the $\sqrt{\text{Gini}}$ isometrics are the only ones that haven't moved.

The upshot of all this is that if you learn a decision tree or probability estimation tree using entropy or Gini index as impurity measure – which is what virtually all available tree learning packages do – then your model will change if you change the class distribution by oversampling, while if you use $\sqrt{\text{Gini}}$ you will learn the same tree each time. More generally, *entropy and Gini index are sensitive to fluctuations in the class distribution*, $\sqrt{\text{Gini}}$ *isn't*. So which one

should you choose? My recommendation echoes the ones I gave for majority class labelling and pruning: use a distribution-insensitive impurity measure such as $\sqrt{\text{Gini}}$ unless the training set operating conditions are representative.[5]

Let's wrap up the discussion on tree models so far. How would *you* train a decision tree on a given data set, you might ask me? Here's a list of the steps I would take:

1. First and foremost, I would concentrate on getting good ranking behaviour, because from a good ranker I can get good classification and probability estimation, but not necessarily the other way round.
2. I would therefore try to use an impurity measure that is distribution-insensitive, such as $\sqrt{\text{Gini}}$; if that isn't available and I can't hack the code, I would resort to oversampling the minority class to achieve a balanced class distribution.
3. I would disable pruning and smooth the probability estimates by means of the Laplace correction (or the *m*-estimate).
4. Once I know the deployment operation conditions, I would use these to select the best operating point on the ROC curve (i.e., a threshold on the predicted probabilities, or a labelling of the tree).
5. (optional) Finally, I would prune away any subtree whose leaves all have the same label.

Even though in our discussion we have mostly concentrated on binary classification tasks, it should be noted that decision trees can effortlessly deal with more than two classes – as, indeed, can any grouping model. As already mentioned, multi-class impurity measures simply sum up impurities for each class in a one-versus-rest manner. The only step in this list that isn't entirely obvious when there are more than two classes is step 4: in this case I would learn a weight for each class as briefly explained in Section 3.1, or possibly combine it with step 5 and resort to reduced-error pruning (Algorithm 5.3) which might be already implemented in the package you're using.

5.3 Tree learning as variance reduction We will now consider how to adapt decision trees to regression and clustering tasks. This will turn out to be surprisingly straightforward,

and is based on the following idea. Earlier, we defined the two-class Gini index $2\dot{p}(1 - \dot{p})$ of a leaf as the expected error resulting from labelling instances in the leaf randomly: positive with probability $\dot{p}$ and negative with probability $1 - \dot{p}$. You can picture this as tossing a coin, prepared such that it comes up heads with probability $\dot{p}$, to classify examples. Representing this as a random variable with value 1 for heads and 0 for tails, the expected value of this random variable is $\dot{p}$ and its variance $\dot{p}(1 - \dot{p})$ (look up 'Bernoulli trial' online if you want to read up on this). This leads to an alternative interpretation of the Gini index as a variance term: the purer the leaf, the more biased the coin will be, and the smaller the variance. For *k* classes we simply add up the variances of all one-versus-rest random variables.[6]

More specifically, consider a binary split into n_1 and $n_2 = n - n_1$ examples with empirical probabilities $\dot{p}_1$ and $\dot{p}_2$, then the weighted average impurity of these children in terms of the Gini index is

$$\frac{n_1}{n}2\dot{p}_1(1-\dot{p}_1)+\frac{n_2}{n}2\dot{p}_2(1-\dot{p}_2)=2\left(\frac{n_1}{n}\sigma_1^2+\frac{n_2}{n}\sigma_2^2\right)$$

where σ_j^2 is the variance of a Bernoulli distribution with success probability $\dot{p}_j$. So, finding a split with minimum weighted average Gini index is equivalent to

minimising weighted average variance (the factor 2 is common to all splits and so can be omitted), and learning a decision tree boils down to partitioning the instance space such that each segment has small variance.

Regression trees In regression problems the target variable is continuous rather than binary, and in that case we can define the variance of a set Y of target values as the average squared distance from the mean: $\mathrm{Var}(Y) = \frac{1}{|Y|}\sum_{y \in Y}(y-\overline{y})^2$

where $\overline{y} = \frac{1}{|Y|}\sum_{y \in Y} y$ is the mean of the target values in Y; see Background 5.1 for some useful properties of variance. If a split partitions the set of target values Y into mutually exclusive sets $\{Y_1, \ldots, Y_l\}$, the weighted average variance is then

$$\mathrm{Var}(\{Y_1,\ldots,Y_l\}) = \sum_{j=1}^{l}\frac{|Y_j|}{|Y|}\mathrm{Var}(Y_j) = \sum_{j=1}^{l}\frac{|Y_j|}{|Y|}\left(\frac{1}{|Y_j|}\sum_{y \in Y_j} y^2 - \overline{y}_j^2\right) = \frac{1}{|Y|}\sum_{y \in Y} y^2 - \sum_{j=1}^{l}\frac{|Y_j|}{|Y|}\overline{y}_j^2 \tag{5.4}$$

The *variance* of a set of numbers $X \subseteq \mathbb{R}$ is defined as the average squared difference from the mean: $\mathrm{Var}(X) = \frac{1}{|X|}\sum_{x \in X}(x-\overline{x})^2$

where $\overline{x} = \frac{1}{|X|}\sum_{x \in X} x$ is the mean of X. Expanding $(x-\overline{x})^2 = x^2 - 2\overline{x}x + \overline{x}^2$ this can be written as

$$\mathrm{Var}(X) = \frac{1}{|X|}\left(\sum_{x \in X} x^2 - 2\overline{x}\sum_{x \in X} x + \sum_{x \in X}\overline{x}^2\right) = \frac{1}{|X|}\left(\sum_{x \in X} x^2 - 2\overline{x}|X|\overline{x} + |X|\overline{x}^2\right) = \frac{1}{|X|}\sum_{x \in X} x^2 - \overline{x}^2 \tag{5.3}$$

So the variance is the difference between the mean of the squares and the square of the mean.

It is sometimes useful to consider the average squared difference from another value $x' \in \mathbb{R}$, which can similarly be expanded: $\frac{1}{|X|}\sum_{x \in X}(x-x')^2 = \frac{1}{|X|}\left(\sum_{x \in X} x^2 - 2x'|X|\overline{x} + |X|x'^2\right) = \mathrm{Var}(X) + (x'-\overline{x})^2$

The last step follows because from Equation 5.3 we have $\frac{1}{|X|}\sum_{x \in X} x^2 = \mathrm{Var}(X) + \overline{x}^2$.

Another useful property is that the average squared difference between any two elements of X is twice the variance: $\frac{1}{|X|^2}\sum_{x' \in X}\sum_{x \in X}(x-x')^2 = \frac{1}{|X|}\sum_{x' \in X}(\mathrm{Var}(X) + (x'-\overline{x})^2) = \mathrm{Var}(X) + \frac{1}{|X|}\sum_{x' \in X}(x'-\overline{x})^2 = 2\mathrm{Var}(X)$

If $X \subseteq \mathbb{R}^d$ is a set of d-vectors of numbers, we can define the variance $\mathrm{Var}_i(X)$ for each of the d coordinates. We can then interpret the sum of

variances $\sum_{i=1}^{d} \mathrm{Var}_i(X)$ as the average squared Euclidean distance of the vectors in X to their vector mean $\overline{\mathbf{x}} = \frac{1}{|X|}\sum_{\mathbf{x}\in X}\mathbf{x}$.
(You will sometimes see sample variance defined as $\frac{1}{|X|-1}\sum_{x\in X}(x-\overline{x})^2$, which is a somewhat larger value. This version arises if we are estimating the variance of a population from which X is a random sample: normalising by $|X|$ would underestimate the population variance because of differences between the sample mean and the population mean. Here, we are only concerned with assessing the spread of the given values X and not with some unknown population, and so we can ignore this issue.)

Background 5.1. Variations on variance.

So, in order to obtain a regression tree learning algorithm, we replace the impurity measure Imp in Algorithm 5.2 with the function Var. Notice that $\frac{1}{|Y|}\sum_{y\in Y} y^2$ is constant for a given set Y, and so minimising variance over all possible splits of a given parent is the same as maximising the weighted average of squared means in the children. The function Label(Y) is similarly adapted to return the mean value in Y, and the function Homogeneous(Y) returns true if the variance of the target values in Y is zero (or smaller than a low threshold).

Figure 5.8. A regression tree learned from the data in Example 5.4.

Example 5.4 (Learning a regression tree). Imagine you are a collector of vintage Hammond tonewheel organs. You have been monitoring an online auction site, from which you collected some data about interesting transactions:

#	Model	Condition	Leslie	Price
1.	B3	excellent	no	4513
2.	T202	fair	yes	625
3.	A100	good	no	1051
4.	T202	good	no	270
5.	M102	good	yes	870
6.	A100	excellent	no	1770
7.	T202	fair	no	99
8.	A100	good	yes	1900
9.	E112	fair	no	77

From this data, you want to construct a regression tree that will help you determine a reasonable price for your next purchase.

There are three features, hence three possible splits:

Model = [A100, B3, E112, M102, T202]	[1051, 1770, 1900][4513][77][870][99, 270, 625]
Condition = [excellent, good, fair]	[1770, 4513][270, 870, 1051, 1900][77, 99, 625]
Leslie = [yes, no]	[625, 870, 1900][77, 99, 270, 1051, 1770, 4513]

The means of the first split are 1574, 4513, 77, 870 and 331, and the weighted average of squared means is $3.21 \cdot 10^6$. The means of the second split are 3142, 1023 and 267, with weighted average of squared means $2.68 \cdot 10^6$; for the third split the means are 1132 and 1297, with weighted average of squared means $1.55 \cdot 10^6$. We therefore branch on Model at the top level. This gives us three single-instance leaves, as well as three A100s and three T202s.

For the A100s we obtain the following splits:

Condition = [excellent, good, fair]	[1770][1051, 1900][]
Leslie = [yes, no]	[1900][1051, 1770]

Without going through the calculations we can see that the second split results in less variance (to handle the empty child, it is customary to set its variance equal to that of the parent). For the T202s the splits are as follows:

Condition = [excellent, good, fair]	[][270][99, 625]
Leslie = [yes, no]	[625][99, 270]

Again we see that splitting on Leslie gives tighter clusters of values. The learned regression tree is depicted in Figure 5.8.

Regression trees are susceptible to overfitting. For instance, if we have exactly one example for each Hammond model then branching on Model will reduce the average variance in the children to zero. The data in Example 5.4 is really too sparse to learn a good regression tree. Furthermore, it is a good idea to set aside a pruning set and to apply reduced-error pruning, pruning away a subtree if the average variance on the pruning set is lower without the subtree than with it (see Algorithm 5.3 on p.144). It should also be noted that predicting a constant value in a leaf is a very simple strategy, and methods exist to learn so-called *model trees*, which are trees with linear regression models in their leaves (☞*linear regression* is explained in Chapter 7). In that case, the splitting criterion would be based on correlation of the target variable with the regressor variables, rather than simply on variance.

Clustering trees *The simple kind of regression tree considered here also suggests a way to learn clustering trees. This is perhaps surprising, since regression is a supervised learning problem while clustering is unsupervised. The key insight is that regression trees find instance space segments whose target values are tightly clustered around the mean value in the segment – indeed, the variance of a set of target values is simply the (univariate) average squared Euclidean distance to the mean. An immediate generalisation is to use a vector of target values, as this doesn't change the mathematics in an*

essential way. More generally yet, we can introduce an abstract function Dis : $\mathscr{X} \times \mathscr{X} \rightarrow \mathbb{R}$ that measures the distance or dissimilarity of any two instances $x, x' \in \mathscr{X}$, such that the higher Dis(x, x') is, the less similar x and x' are. The cluster dissimilarity of a set of instances D is then calculated as

$$\mathrm{Dis}(D) = \frac{1}{|D|^2} \sum_{x \in D} \sum_{x' \in D} \mathrm{Dis}(x, x') \qquad (5.5)$$

The weighted average cluster dissimilarity over all children of a split then gives the *split dissimilarity*, which can be used to inform BestSplit(*D, F*) in the *GrowTree algorithm* (Algorithm 5.1 on p.132).

Example 5.5 (Learning a clustering tree using a dissimilarity matrix). Assessing the nine transactions on the online auction site from Example 5.4, using some additional features such as reserve price and number of bids (these features do not matter at the moment but are shown in Example 5.6), you come up with the following dissimilarity matrix:

0	11	6	13	10	3	13	3	12
11	0	1	1	1	3	0	4	0
6	1	0	2	1	1	2	2	1
13	1	2	0	0	4	0	4	0
10	1	1	0	0	3	0	2	0
3	3	1	4	3	0	4	1	3
13	0	2	0	0	4	0	4	0
3	4	2	4	2	1	4	0	4
12	0	1	0	0	3	0	4	0

This shows, for instance, that the first transaction is very different from the other eight. The average pairwise dissimilarity over all nine transactions is 2.94.

Using the same features from Example 5.4, the three possible splits are (now with transaction number rather than price):

Model = [A100, B3, E112, M102, T202]	[3, 6, 8][1][9][5][2, 4, 7]
Condition = [excellent, good, fair]	[1, 6][3, 4, 5, 8][2, 7, 9]
Leslie = [yes, no]	[2, 5, 8][1, 3, 4, 6, 7, 9]

The cluster dissimilarity among transactions 3, 6 and 8 is $\frac{1}{3^2}$ (**0**+**1**+**2**+**1**+**0**+**1**+**2**+**1**+**0**) = 0.89; and among transactions 2, 4 and 7 it is $\frac{1}{3^2}$ (**0**+**1**+**0**+**1**+**0**+**0**+**0**+**0**+**0**) = 0.22. The other three children of the first split contain only a single element and so have zero cluster dissimilarity. The weighted average cluster dissimilarity of the split is then 3/9·0.89 +1/9·0+1/9·0+1/9·0+3/9·0.22 = 0.37. For the second split, similar calculations result in a split dissimilarity of 2/9·1.5+4/9·1.19+3/9·0 = 0.86, and the third split yields 3/9·1.56+6/9·3.56 = 2.89. The Model feature thus captures most of the given dissimilarities, while the Leslie feature is virtually unrelated.

Most of the caveats of regression trees also apply to clustering trees: smaller clusters tend to have lower dissimilarity, and so it is easy to overfit. Setting aside a pruning set to remove the lower splits if they don't improve the cluster coherence on the pruning set is recommended. Single examples can dominate: in the above example, removing the first transaction reduces the overall pairwise dissimilarity from 2.94 to 1.5, and so it will be hard to beat a split that puts that transaction in a cluster of its own.

An interesting question is: how should the leaves of a clustering tree be labelled? Intuitively, it makes sense to label a cluster with its most representative instance. We can define an instance as most representative if its total dissimilarity to all other instances is lowest – this is defined as the medoid in Chapter 8. For instance, in the A100 cluster transaction 6 is most representative because its dissimilarity to 3 and 8 is 1, whereas the dissimilarity between 3 and 8 is 2. Likewise, in the T202 cluster transaction 7 is most representative. However, there is no reason why this should always be uniquely defined.

A commonly encountered scenario, which both simplifies the calculations involved in determining the best split and provides a unique cluster label, is when the dissimilarities are Euclidean distances derived from numerical features. As shown in Background 5.1, if Dis(x, x') is squared Euclidean distance, then Dis(D) is twice the average squared Euclidean distance to the mean. This simplifies calculations because both the mean and average squared distance to the mean can be calculated in $O(|D|)$ steps (a single sweep through the data), rather than the $O(|D|^2)$ required if all we have is a dissimilarity matrix. In fact, the average squared Euclidean distance is simply the sum of the variances of the individual features.

Figure 5.9. A clustering tree learned from the data in Example 5.6 using Euclidean distance on the numerical features.

Example 5.6 (Learning a clustering tree with Euclidean distance). We extend our Hammond organ data with two new numerical features, one indicating the reserve price and the other the number of bids made in the auction. Sales price and reserve price are expressed in hundreds of pounds in order to give the three numerical features roughly equal weight in the distance calculations.

Model	Condition	Leslie	Price	Reserve	Bids
B3	excellent	no	45	30	22
T202	fair	yes	6	0	9
A100	good	no	11	8	13
T202	good	no	3	0	1
M102	good	yes	9	5	2
A100	excellent	no	18	15	15
T202	fair	no	1	0	3
A100	good	yes	19	19	1
E112	fair	no	1	0	5

The means of the three numerical features are (13.3, 8.6, 7.9) and their variances are (158, 101.8, 48.8). The average squared Euclidean distance to the mean is then the sum of these variances, which is 308.6 (if preferred we can double this number to obtain the cluster dissimilarity as defined in Equation 5.5). For the A100 cluster these vectors are (16, 14, 9.7) and (12.7, 20.7, 38.2), with average squared distance to the mean 71.6; for the T202 cluster they are (3.3, 0, 4.3) and (4.2, 0, 11.6), with average squared distance 15.8. Using this split we can construct a clustering tree whose leaves are labelled with the mean vectors (Figure 5.9).

In this example we used categorical features for splitting and numerical features for distance calculations. Indeed, in all tree examples considered so far we have only used categorical features for splitting.[7] In practice, numerical features are frequently used for splitting: all we need to do is find a suitable threshold t so that feature F can be turned into a binary split with conditions $F \geq t$ and $F < t$. Finding the optimal split point is closely related to *discretisation* of numerical features, a topic we will look at in detail in Chapter 10. For the moment, the following observations give some idea how we can learn a threshold on a numerical feature:

- Although in theory there are infinitely many possible thresholds, in practic we only need to consider values separating two examples that end up next to each other if we sort the training examples on increasing (or decreasing value of the feature.
- We only consider consecutive examples of different class if our task is classification, whose target values are sufficiently different if our task is regression, or whose dissimilarity is sufficiently large if our task is clustering.
- Each potential threshold can be evaluated as if it were a distinct binary feature.

5.4 Tree models: Summary and further reading

Tree-based data structures are ubiquitous in computer science, and the situation is no different in machine learning. Tree models are concise, easy to interpret and learn, and can be applied to a wide range of tasks, including classification, ranking, probability estimation, regression and clustering. The tree-based classifier for human pose

recognition in the Microsoft Kinect motion sensing device is described in Shotton *et al.* (2011).

- I introduced the feature tree as the common core for all these tree-based models, and the recursive GrowTree algorithm as a generic divide-and-conquer algorithm that can be adapted to each of these tasks by suitable choices for the functions that test whether a data set is sufficiently homogeneous, find a suitable label if it is, and find the best feature to split on if it isn't.

- Using a feature tree to predict class labels turns them into decision trees, tl subject of Section 5.1. There are two classical accounts of decision trees ir machine learning, which are very similar algorithmically but differ in details such as heuristics and pruning strategies. Quinlan's approach was t use entropy as impurity measure, and progressed from the ID3 algorithm (Quinlan, 1986), which itself was inspired by Hunt, Marin and Stone (1966), to the sophisticated C4.5 system (Quinlan, 1993). The CART approach stands for 'classification and regression trees' and was developed by Breiman, Friedman, Olshen and Stone (1984); it uses the Gini index as impurity measure. The $\sqrt{\text{Gini}}$ impurity measure was introduced by Dietterich, Kearns and Mansour (1996), and is hence sometimes referred t as *DKM*. The geometric construction to find Imp($\{D_1, D_2\}$) in Figure 5.2 (right) was also inspired by that paper.

- Employing the empirical distributions in the leaves of a feature tree in ord to build rankers and probability estimators as described in Section 5.2 is a much more recent development (Ferri *et al.*, 2002; Provost and Domingos 2003). Experimental results demonstrating that better probability estimate are obtained by disabling tree pruning and smoothing the empirical probabilities by means of the Laplace correction are presented in the latter paper and corroborated by Ferri *et al.* (2003). The extent to which decisior tree splitting criteria are insensitive to unbalanced classes or misclassification costs was studied and explained by Drummond and Holt (2000) and Flach (2003). Of the three splitting criteria mentioned above, only $\sqrt{\text{Gini}}$ is insensitive to such class and cost imbalance.

- Tree models are grouping models that aim to minimise diversity in their

leaves, where the appropriate notion of diversity depends on the task. Very often diversity can be interpreted as some kind of variance, an idea that already appeared in (Breiman *et al.*, 1984) and was revisited by Langley (1994), Kramer (1996) and Blockeel, De Raedt and Ramon (1998), among others. In Section 5.3 we saw how this idea can be used to learn regression and clustering trees (glossing over many important details, such as when we should stop splitting nodes).

It should be kept in mind that the increased expressivity of tree models compared with, say, conjunctive concepts means that we should safeguard ourselves against overfitting. Furthermore, the greedy divide-and-conquer algorithm has the disadvantage that small changes in the training data may lead to a different choice of the feature at the root of the tree, which will influence the choice of feature at subsequent splits. We will see in Chapter 11 how methods such as bagging can be applied to help reduce this kind of model variance.

---

[1] If we allowed the creation of new conjunctive features, we could actually represent this tree as the conjunctive concept Gills = no ∧ F = false, where $F \equiv$ Length = 4 ∧ Teeth = few is a new conjunctive feature. The creation of new features during learning is called *constructive induction*, and as shown here can extend the representational power of a logical language.

[2] If there is more than one largest class we will make an arbitrary choice between them, usually uniformly random.

[3] When I looked up 'Gini index' on Wikipedia I was referred to a page describing the *Gini coefficient*, which – in a machine learning context – is a linear rescaling of the AUC to the interval [–1, 1]. This is quite a different concept, and the only thing that the Gini index and the Gini coefficient have in common is that they were both proposed by the Italian statistician Corrado Gini, so it is good to be aware of potential confusion.

[4] Tie breaking – although it does not alter the shape of the coverage curve and isn't essential in that sense – can also be achieved by subtracting $\epsilon \ll 1$ from the number of positives covered. The Laplace correction also breaks ties in favour of larger leaves but isn't a monotonic transformation and so might change the shape of the coverage curve.

[5] It should be noted that it is fairly easy to make measures such as entropy and Gini index distribution-insensitive as well: essentially, this would involve compensating for an observed class ratio $clr \neq 1$ by dividing all counts of positives, or positive empirical probabilities, by clr.

[6] This implicitly assumes that the one-versus-rest variables are uncorrelated, which is not strictly true.

[7] Categorical features are features with a relatively small set of discrete values. Technically, they distinguish themselves from numerical features by not having a scale or an ordering. This is further explored in Chapter 10.

CHAPTER 6

Rule models

RULE MODELS ARE the second major type of logical machine learning models. Generally speaking, they offer more flexibility than tree models: for instance, while decision tree branches are mutually exclusive, the potential overlap of rules may give additional information. This flexibility comes at a price, however: while it is very tempting to view a rule as a single, independent piece of information, this is often not adequate because of the way the rules are learned. Particularly in supervised learning, a rule model is more than just a set of rules: the specification of how the rules are to be combined to form predictions is a crucial part of the model.

There are essentially two approaches to supervised rule learning. One is inspired by decision tree learning: find a combination of literals – the *body* of the rule, which is what we previously called a concept – that covers a sufficiently homogeneous set of examples, and find a label to put in the *head* of the rule. The second approach goes in the opposite direction: first select a class you want to learn, and then find rule bodies that cover (large subsets of) the examples of that class. The first approach naturally leads to a model consisting of an ordered sequence of rules – a *rule list* – as will be discussed in Section 6.1. The second approach treats collections of rules as unordered *rule sets* and is the topic of Section 6.2. We shall see how these models differ in the way they handle rule overlap. The third section of the chapter covers discovery of subgroups and association rules.

Figure 6.1. ROC isometrics for entropy (rescaled to have a maximum value of 1/2), Gini index and minority class. The grey dotted symmetry line is defined by $\dot{p}$ = 1/2: each isometric has two parts, one above the symmetry line (where impurity decreases with increasing empirical probability $\dot{p}$) and its mirror image below the symmetry line (where impurity is proportional to $\dot{p}$). If these impurity measures are used as search heuristic, as they are in rule learning, only the shape of the isometrics matters but not the associated impurity values, and hence all three impurity measures are equivalent.

6.1 Learning ordered rule lists The key idea of this kind of rule learning algorithm is to keep growing a conjunctive rule body by adding the literal that most improves its homogeneity. That is, we construct a downward path through the hypothesis space, of the kind discussed in Section 4.2, and we stop as soon as some homogeneity criterion is satisfied. It is natural to measure homogeneity in terms of purity, as we did with decision trees. You might think that adding a literal to a rule body is much the same as adding a binary split to a decision

tree, as the added literal splits the instances covered by the original rule body in two groups: those instances for which the new literal is true, and those for which the new literal is false. However, one key difference is that in decision tree learning we are interested in the purity of *both* children, which is why we use the weighted average impurity as our search heuristic when constructing the tree. In rule learning, on the other hand, we are only interested in the purity of one of the children: the one in which the added literal is true. It follows that we can directly use any of the impurity measures we considered in the previous chapter (see Figure 5.2 on p.134 if you want to remind yourself which they are), without the need for averaging.

In fact, it doesn't even matter which of those impurity measures we use to guide the search, since they will all give the same result. To see this, notice that the impurity of a concept decreases with the empirical probability $\dot{p}$ (the relative frequency of covered positives) if $\dot{p} > 1/2$ and increases with $\dot{p}$ if $\dot{p} < 1/2$; see Figure 6.1. Whether this increase or decrease is linear or not matters if we are averaging the impurities of several concepts, as in decision tree learning, but not if we are evaluating single concepts. In other words, the difference between these impurity measures vanishes in rule learning, and we might as well take the proportion of the minority class $\min(\dot{p}, 1 - \dot{p})$ (or, if you prefer, $1/2 - |\dot{p} - 1/2|$), which is arguably the simplest, as our impurity measure of choice in this section. Just keep in mind that if other authors use entropy or Gini index to compare the impurity of literals or rule bodies this will give the same results (not in terms of

impurity values but in terms of which one is best).

We introduce the main algorithm for learning rule lists by means of an example.

Example 6.1 (Learning a rule list). Consider again our small dolphins data set with positive examples p1: Length = 3 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many p2: Length = 4 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many p3: Length = 3 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few p4: Length = 5 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many p5: Length = 5 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few and negatives

n1: Length = 5 ∧ Gills = yes ∧ Beak = yes ∧ Teeth = many n2: Length = 4 ∧ Gills = yes ∧ Beak = yes ∧ Teeth = many n3: Length = 5 ∧ Gills = yes ∧ Beak = no ∧ Teeth = many n4: Length = 4 ∧ Gills = yes ∧ Beak = no ∧ Teeth = many n5: Length = 4 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few The nine possible literals are shown with their coverage counts in Figure 6.2 (top). Three of these are pure; in the impurity isometrics plot in Figure 6.2 (bottom) they end up on the *x*-axis and *y*-axis. One of the literals covers two positives and two negatives, and therefore has the same impurity as the overall data set; this literal ends up on the ascending diagonal in the coverage plot.

Although impurity in itself does not distinguish between pure literals (we will return to this point later), one could argue that Gills = yes is the best of the three as it covers more examples, so let's formulate our first rule as:

·**if** Gills = yes **then** Class = ⊖.

Figure 6.2. (top) All literals with their coverage counts on the data in Example 6.1. The ones in green (red) are pure for the positive (negative) class. **(bottom)** The nine literals plotted as points in coverage space, with their impurity values indicated by impurity isometrics (away from the ascending diagonal is better). Impurity values are colour-coded: towards green if $\dot{p} > 1/2$, towards red if $\dot{p} < 1/2$, and orange if $\dot{p} = 1/2$ (on a 45 degree isometric). The violet arrow indicates the selected literal, which excludes all five positives and one negative.

The corresponding coverage point is indicated by the arrow in Figure 6.2 (bottom). You can think of this arrow as the right-most bit of the coverage curve that results if we keep on following a downward path through the hypothesis space by adding literals. In this case we are not interested in following the path further because the concept we found is already pure (we shall see examples later where we have to add several literals before we hit one of the axes). One new thing that we haven't seen before is that this coverage curve lies below the diagonal – this is a consequence of the fact that we haven't fixed the class in advance, and therefore we are just as happy diving deep beneath the ascending diagonal as we would be flying high above it. Another way of thinking about this is that if we swap the labels this affects the heads but not the bodies of the

learned rules.

Figure 6.3. (left) Revised coverage counts after removing the four negative examples covered by the first rule found (literals not covering any examples are omitted). **(right)** We are now operating in the right-most 'slice' of Figure 6.2 on p.160.

Most rule learning algorithms now proceed as follows: they remove the examples covered by the rule just learned from consideration, and proceed with the remaining examples. This strategy is called *separate-and-conquer*, in analogy with the divide-and-conquer strategy of decision trees (the difference is that in separate-and-conquer we end up with one remaining subproblem rather than several as in divide-and-conquer). So we are left with five positive examples and one negative, and we again search for literals with minimum impurity. As is shown in Figure 6.3, we can understand this as working in a smaller coverage space. After going through the numbers, we find the next rule learned is ·**if** Teeth = many **then** Class = ⊕ ·

As I mentioned earlier, we should be cautious when interpreting this rule on its own, as against the original data set it actually covers more negatives than positives! In other words, the rule implicitly assumes that the previous rule doesn't 'fire'; in the final rule model we will precede it with 'else'.

We are now left with two positives and one negative (Figure 6.4). This time it makes sense to choose the rule that covers the single remaining negative, which is

·**if** Length = 4 **then** Class = ⊖·

Since the remaining examples are all positive, we can invoke a *default rule* to cover those examples for which all other rules fail. Put together, the learned rule model is then as follows:

Figure 6.4. (left) The third rule covers the one remaining negative example, so that the remaining positives can be swept up by a default rule. **(right)** This will collapse the coverage space.

·**if** Gills = yes **then** Class = ⊖·
·**else if** Teeth = many **then** Class = ⊕·
·**else if** Length = 4 **then** Class = ⊖·
·**else** Class = ⊕·

Organising rules in a list is one way of dealing with overlaps among rules. For example, we know from the data that there are several examples with both Gills = yes and Teeth = many, but the rule list above tells us that the first rule takes precedence in such cases. Alternatively, we could rewrite the rule list such that the rules are mutually exclusive. This is useful because it means that we can use each rule without reference to the other rules, and also ignore their ordering. The only slight complication is that we need negated literals (or internal disjunction) for those features that have more than two values, such as 'Length': ·**if** Gills = yes **then** Class = ⊖·
·**if** Gills = no ∧ Teeth = many **then** Class = ⊕·
·**if** Gills = no ∧ Teeth = few ∧ Length = 4 **then** Class = ⊖·
·**if** Gills = no ∧ Teeth = few ∧ Length ≠ 4 **then** Class = ⊕·

In this example we rely on the fact that this particular set of rules has a single literal in each rule – in the general case we would need non-conjunctive rule bodies. For example, consider the following rule list: ·**if** $P \wedge Q$ **then** Class = ⊕·
·**else if** R **then** Class = ⊖·

If we wanted to make these mutually exclusive the second rule would become ·**if** $\neg(P \wedge Q) \wedge R$ **then** Class = ⊖·

or equivalently, ·**if** $\neg(P \vee \neg Q) \wedge R$ **then** Class = ⊖·

Clearly, making rules mutually exclusive leads to less compact rules, which explains why rule lists are a powerful and popular format.

Algorithm 6.1 specifies the separate-and-conquer rule learning strategy in

more detail. While there are still training examples left, the algorithm learns another rule and removes all examples covered by the rule from the data set. This algorithm, which is the basis for the majority of rule learning systems, is also called the *covering algorithm*. The algorithm for learning a single rule is given in Algorithm 6.2. Similar to decision trees, it uses the functions Homogeneous(D) and Label(D) to decide whether further specialisation is needed and what class to put in the head of the rule, respectively. It also employs a function BestLiteral (D, L) that selects the best literal to add to the rule from the candidates in L given data D; in our example above, this literal would be selected on purity.

Many variations on these algorithms exist in the literature. The conditions in the while-loops are often relaxed to other *stopping criteria* in order to deal with noisy data. For example, in Algorithm 6.1 we may want to stop when no class has more than a certain number of examples left, and include a default rule for the remaining examples. Likewise, in Algorithm 6.2 we may want to stop if D drops below a certain size.

Rule lists have much in common with decision trees. We can therefore analyse the construction of a rule list in the same way as we did in Figure 5.3 on p.137. This is shown for the running example in Figure 6.5. For example, adding the first rule is depicted in coverage space by splitting the ascending diagonal A into a horizontal segment B representing the new rule and another diagonal segment C representing the new coverage space. Adding the second rule causes segment C to split into vertical segment D (the second rule) and diagonal segment E (the third coverage space). Finally, E is split into a horizontal and a vertical segment (the third rule and the default rule, respectively). The remaining segments B, D, F and G are now all horizontal or vertical, signalling that the rules we learned are pure.

Algorithm 6.1: LearnRuleList(D) – learn an ordered list of rules.

Input : labelled training data D.

Output : rule list R.

```
R ← ∅;
while D ≠ ∅ do
    r ← LearnRule(D) ;                    // LearnRule: see Algorithm 6.2
    append r to the end of R;
    D ← D \ {x ∈ D | x is covered by r};
end
return R
```

Figure 6.5. (left) A right-branching feature tree corresponding to a list of single-literal rules. **(right)** The construction of this feature tree depicted in coverage space. The leaves of the tree are either purely positive (in green) or purely negative (in red). Reordering these leaves on their empirical probability results in the blue coverage curve. As the rule list separates the classes this is a perfect coverage curve.

Rule lists for ranking and probability estimation Turning a rule list into a ranker or probability estimator is as easy as it was for decision trees. Due to the covering algorithm we have access to the local class distributions associated with each rule. We can therefore base our scores on the empirical probabilities. In the case of two classes we can rank the instances on decreasing empirical probability of the positive class, giving rise to a coverage curve with one segment for each rule. It is important to note that the ranking order of the rules is different from their order in the rule list, just as the ranking order of the leaves of a tree is different from their left-to-right order.

Algorithm 6.2: LearnRule(D) – learn a single rule.

```
Input  : labelled training data D.
Output : rule r.
b ← true;
L ← set of available literals;
while not Homogeneous(D) do
    l ← BestLiteral(D, L) ;                    // e.g., highest purity; see text
    b ← b ∧ l;
    D ← {x ∈ D | x is covered by b};
    L ← L \ {l′ ∈ L | l′ uses same feature as l};
end
C ← Label(D) ;                                 // e.g., majority class
r ← ·if b then Class = C·;
return r
```

Example 6.2 (Rule lists as rankers). Consider the following two concepts:

(A)	Length = 4	p2	n2, n4–5
(B)	Beak = yes	pl–5	n1–2, n5

Indicated on the right is each concept's coverage over the whole training set. Using these concepts as rule bodies, we can construct the rule list AB:

·**if** Length = 4 **then** Class = ⊖·	[1+, 3−]
·**else if** Beak = yes **then** Class = ⊕·	[4+, 1−]
·**else** Class = ⊖·	[0+, 1−]

The coverage curve of this rule list is given in Figure 6.6. The first segment of the curve corresponds to all instances which are covered by B but not by A, which is why we use the set-theoretical notation B\A. Notice that while this segment corresponds to the second rule in the rule list, it comes first in the coverage curve because it has the highest proportion of positives. The second coverage segment corresponds to rule A, and the third coverage

segment denoted '-' corresponds to the default rule. This segment comes last, not because it represents the last rule, but because it happens to cover no positives.

We can also construct a rule list in the opposite order, BA:

·**if** Beak = yes **then** Class = ⊕·	[5+, 3−]
·**else if** Length = 4 **then** Class = ⊖·	[0+, 1−]
·**else** Class = ⊖·	[0+, 1−]

The coverage curve of this rule list is also depicted in Figure 6.6. This time, the first segment corresponds to the first segment in the rule list (B), and the second and third segment are tied between rule A (after the instances covered by B are taken away: A\B) and the default rule.

Which of these rule lists is a better ranker? We can see that AB makes fewer ranking errors than BA (4.5 vs. 7.5), and thus has better AUC (0.82 vs. 0.70). We also see that, if accuracy is our performance criterion, AB would be optimal, achieving 0.80 accuracy (*tpr* = 0.80 and *tnr* = 0.80) where BA only manages 0.70 (*tpr* = 1 and *tnr* = 0.40). However, if performance on the positives is 3 times as important as performance on the negatives, then BA's optimal operating point outperforms AB's. Hence, each rule list contains information not present in the other, and so neither is uniformly better.

Figure 6.6. Coverage curves of two rule lists consisting of the rules from Example 6.2, in different order

(AB in blue and BA in violet). B \ A corresponds to the coverage of rule B once the coverage of rule A is taken away, and '-' denotes the default rule. Neither curve dominates the other, and thus each has operating conditions under which it is superior. The dotted segment in red connecting the two curves corresponds to the overlap of the two rules A ∧ B, which is not accessible by either rule list.

The main reason for this is that the segment A ∧ B – the overlap of the two rules – is not accessible by either rule list. In Figure 6.6 this is indicated by the dotted segment connecting the segment B from rule list BA and the segment B \ A from rule list AB. It follows that this segment contains exactly those examples that are in B but not in B \ A, hence in A ∧ B. In order to access the rule overlap, we need to either combine the two rule lists or go beyond the power of rule lists. This will be investigated further at the end of the next section.

There are thus several connections between rule lists and decision trees. Furthermore, *rule lists are similar to decision trees in that the empirical probabilities associated with each rule yield convex ROC and coverage curves on the training data*. We have access to those empirical probabilities because of the coverage algorithm, which removes all training instances covered by one rule before learning the next (Algorithm 6.1). As a result, rule lists produce probabilities that are well-calibrated on the training set. Some rule learning algorithms in the literature reorder the rule list after all rules have been constructed. In this case, convexity cannot be guaranteed unless we re-evaluate the coverage of each rule in the reordered rule list.

6.2 Learning unordered rule sets

We next consider the alternative approach to rule learning, where rules are learned for one class at a time. This means we can further simplify our search heuristic: rather than minimising min($\dot{p}$, 1 − $\dot{p}$), we can maximise $\dot{p}$, the empirical probability of the class we are learning. This search heuristic is conventionally referred to by its 'evaluation measure name' *precision* (see Table 2.3 on p.57).

Example 6.3 (Learning a rule set for one class). We continue the dolphin example. Figure 6.7 shows that the first rule learned for the positive class is ·**if** Length = 3 **then** Class = ⊕·

The two examples covered by this rule are removed, and a new rule is learned. We now encounter a new situation, as none of the candidates is pure (Figure 6.8). We thus start a second-level search, from which the following pure rule emerges: ·**if** Gills = no ∧ Length = 5 **then** Class = ⊕·

To cover the remaining positive, we again need a rule with two conditions (Figure 6.9): ·**if** Gills = no ∧ Teeth = many **then** Class = ⊕·

Notice that, even though these rules are overlapping, their overlap only covers positive examples (since each of them is pure) and so there is no need to organise them in an if-then-else list.

We now have a rule set for the positive class. With two classes this might be considered sufficient, as we can classify everything that isn't covered by the positive rules as negative. However, this might introduce a bias towards the negative class as all difficult cases we're unsure about get automatically classified as negative. So let's learn some rules for the negative class. By the same procedure as in Example 6.3 we find the following rules (you may want to check this): ·**if** Gills = yes **then** Class = ⊖· first, followed by ·**if** Length = 4 ∧ Teeth = few **then** Class = ⊖·. The final rule set with rules for both classes is therefore

Figure 6.7. (top) The first rule is learned for the positive class. **(bottom)** Precision isometrics look identical to impurity isometrics (Figure 6.2); however, the difference is that precision is lowest on the *x*-axis and highest on the *y*-axis, while purity is lowest on the ascending diagonal and highest on both the *x*-axis and the *y*-axis.

(R1) ·**if** Length = 3 **then** Class = ⊕·

(R2) ·**if** Gills = no ∧ Length = 5 **then** Class = ⊕·

(R3) ·**if** Gills = no ∧ Teeth = many **then** Class = ⊕·

(R4) ·**if** Gills = yes **then** Class = ⊖·

(R5) ·**if** Length = 4 ∧ Teeth = few **then** Class = ⊖·

The algorithm for learning a rule set is given in Algorithm 6.3. The main differences with ☞*LearnRuleList* (Algorithm 6.1 on p.163) is that we now iterate over each class in turn, and furthermore that only covered examples for the class that we are currently learning are removed after a rule is found. The reason for

this second change is that rule sets are not executed in any particular order, and so covered negatives are not filtered out by other rules. Algorithm 6.4 gives the algorithm for learning a single rule for a particular class, which is very similar to ☞*LearnRule* (Algorithm 6.2 on p.164) except (*i*) the best literal is now chosen with regard to the class to be learned, C_i; and (*ii*) the head of the rule is always labelled with C_i. An interesting variation that is sometimes encountered in the literature is to initialise the set of available literals L to those occurring in a given *seed example* belonging to the class to be learned: the advantage is that this cuts back the search space, but a possible disadvantage is that the choice of seed example may be sub-optimal.

Figure 6.8. (top) The second rule needs two literals: we use maximum precision to select both. **(bottom)** The coverage space is smaller because the two positives covered by the first rule are removed. The blue box on the left indicates an even smaller coverage space in which the search for the second literal is carried out, after the condition Gills = no filters out four negatives. Inside the blue box precision isometrics overlap with those in the outer box (this is not necessarily the case with search heuristics other than precision).

Figure 6.9. (top) The third and final rule again needs two literals. **(bottom)** The first literal excludes four negatives, the second excludes the one remaining negative.

One issue with using precision as search heuristic is that it tends to focus a bit too much on finding pure rules, thereby occasionally missing near-pure rules that can be specialised into amore general pure rule. Consider Figure 6.10 (top): precision favours the rule ·**if** Length = 3 **then** Class = ⊕·, even though the near-pure literal Gills = no leads to the pure rule ·**if** Gills = no ∧ Teeth = many **then** Class = ⊕·. A convenient way to deal with this 'myopia' of precision is the Laplace correction, which ensures that [5+, 1−] is 'corrected' to [6+, 2−] and thus considered to be of the same quality as [2+, 0−] aka [3+, 1−] (Figure 6.10 (bottom)). Another way to reduce myopia further and break such ties is to employ a *beam search*: rather than greedily going for the best candidate, we maintain a fixed number of alternate candidates. In the example, a small beam size would already allow us to find the more general rule:

- the first beam would include the candidate bodies Length = 3 and Gills = no;
- we then add all possible specialisations of non-pure elements of the beam;
- of the remaining set – i.e., elements of the original beam plus all added

specialisations – we keep only the best few, preferring the ones that were already on the beam in case of ties, as they are shorter;

- we stop when all beam elements are pure, and we select the best one.

Now that we have seen how to learn a rule set, we turn to the question of how to employ a rule set model as a classifier. Suppose we encounter a new instance, say Length = 3 ∧ Gills = yes ∧ Beak = yes ∧ Teeth = many. With the rule list on p.162 the first rule would fire and hence the instance is classified as negative. With the rule set on p.168 we have that both R1 and R4 fire and make contradictory predictions. How can we resolve this? In order to answer that question, it is easier to consider a more general question first: how do we use a rule set for ranking and probability estimation?

Algorithm 6.3: LearnRuleSet(D) – learn an unordered set of rules.

Input : labelled training data D.
Output : rule set R.

```
R ← ∅;
for every class C_l do
    D_l ← D;
    while D_l contains examples of class C_l do
        r ← LearnRuleForClass(D_l, C_l) ;    // LearnRuleForClass: see Algorithm 6.4
        R ← R ∪ {r};
        D_l ← D_l \ {x ∈ C_l | x is covered by r} ;    // remove only positives
    end
end
return R
```

Algorithm 6.4: LearnRuleForClass(D, C_i) – learn a single rule for a given class.

Input : labelled training data D; class C_i.
Output : rule r.

```
b ← true;
L ← set of available literals ;                    // can be initialised by seed example
while not Homogeneous(D) do
    l ← BestLiteral(D, L, C_l) ;                   // e.g. maximising precision on class C_l
    b ← b ∧ l;
    D ← {x ∈ D | x is covered by b};
    L ← L \ {l' ∈ L | l' uses same feature as l};
end
r ← ·if b then Class = C_l·;
return r
```

Figure 6.10. (top) Using Laplace-corrected precision allows learning a better rule in the first iteration. **(bottom)** Laplace correction adds one positive and one negative pseudo-count, which means that the isometrics now rotate around (−1, −1) in coverage space, resulting in a preference for more general rules.

Rule sets for ranking and probability estimation In the general case, for a rule set consisting of r rules there are up to 2^r different ways in which rules can overlap, and hence 2^r instance space segments. Even though many of these segments will be empty because rules are mutually exclusive, in general we will have more instance space segments than rules. As a consequence, we have to estimate the coverage of some of these segments.

Example 6.4 (Rule sets as rankers). Consider the following rule set (the first two rules were also used in Example 6.2):

(A)	·**if** Length = 4 **then** Class = ⊖·	[1+, 3−]
(B)	·**if** Beak = yes **then** Class = ⊕·	[5+, 3−]
(C)	·**if** Length = 5 **then** Class = ⊖·	[2+, 2−]

The figures on the right indicate coverage of each rule over the whole training set. For instances covered by single rules we can use these coverage counts to calculate probability estimates: e.g., an instance covered only by rule A would receive probability $\hat{p}$(A) = 1/4 = 0.25, and similarly $\hat{p}$(B) = 5/8 = 0.63 and $\hat{p}$(C) = 2/4 = 0.50.

Clearly A and C are mutually exclusive, so the only overlaps we need to take into account are AB and BC. A simple trick that is often applied is to average the coverage of the rules involved: for example, the coverage of AB is estimated as [3+, 3−] yielding $\hat{p}$(AB) = 3/6 = 0.50. Similarly, $\hat{p}$(BC) = 3.5/6 = 0.58. The corresponding ranking is thus B – BC – [AB, C] – A, resulting in the orange training set coverage curve in Figure 6.11.

Let us now compare this rule set with the following rule list ABC:

·**if** Length = 4 **then** Class = ⊖·	[1+, 3−]
·**else if** Beak = yes **then** Class = ⊕·	[4+, 1−]
·**else if** Length = 5 **then** Class = ⊖·	[0+, 1−]

The coverage curve of this rule list is indicated in Figure 6.11 as the blue line. We see that the rule set outperforms the rule list, by virtue of being able to distinguish between examples covered by B only and those covered by both B and C.

While in this example the rule set outperformed the rule list, this cannot be guaranteed in general. Due to the fact that the coverage counts of some segments have to be estimated, a rule set coverage curve is not guaranteed to be convex even on the training set. For example, suppose that rule C also covers p1, then this won't affect the performance of the rule list (since p1 is already covered by B), but it would break the tie between AB and C in favour of the latter and thus introduce a concavity.

Figure 6.11. Coverage curves of the rule set in Example 6.4 (in orange) and the rule list ABC (in blue). The rule set partitions the instance space in smaller segments, which in this case lead to better ranking performance.

If we want to turn such a ranker into a classifier, we have to find the best operating point on the coverage curve. Assuming accuracy as our performance criterion, the point (fpr = 0.2, tpr = 0.8) is optimal, which can be achieved by classifying instances with $\hat{p} > 0.5$ as positive and the rest as negative. If such calibration of the decision threshold is problematic (for example, in the case of more than two classes), we can simply assign the class with the highest average

coverage, making a random choice in case of a tie.

A closer look at rule overlap We have seen that rule lists always give convex training set coverage curves, but that there is no globally optimal ordering of a given set of rules. The main reason is that rule lists don't give us access to the overlap of two rules $A \wedge B$: we either have access to $A = (A \wedge B) \vee (A \wedge \neg B)$ if the rule order is AB, or $B = (A \wedge B) \vee (\neg A \wedge B)$ if it is BA. More generally, a rule list of r rules results in only r instance space segments (or $r + 1$ in case we add a default rule). This means that we cannot take advantage of most of the 2^r ways in which rules can overlap. Rule sets, on the other hand, can potentially give access to such overlaps, but the need for the coverage counts of overlapping segments to be estimated means that we have to sacrifice convexity. In order to understand this further, we introduce in this section the concept of a rule tree: a complete feature tree using the rules as features.

Figure 6.12. A rule tree constructed from the rules in Example 6.5. Nodes are labelled with their coverage (dotted leaves have empty coverage), and branch labels indicate particular areas in the instance space (e.g., A-C denotes $A \wedge \neg B \wedge C$). The blue nodes are the instance space segments corresponding to the rule list ABC: the rule tree has better performance because it is able to split them further.

Example 6.5 (Rule tree). From the rules in Example 6.4 we can construct the rule tree in Figure 6.12. The use of a tree rather than a list allows further splitting of the segments of the rule list. For example, the node labelled A is further split into AB ($A \wedge B$) and A- ($A \wedge \neg B$). As the latter is pure, we obtain a better coverage curve (the red line in Figure 6.13).

As we see in this example, the rule tree coverage curve dominates the rule list coverage curve. This is true in general: there is no other information regarding rule overlap than that contained in a rule tree, and any given rule list will usually convey only part of that information. Conversely, we may wonder whether any operating point on the rule tree curve is reachable by a particular rule list. The answer to this is negative, as a simple counter-example shows (Figure 6.14).

Figure 6.13. The blue line is the coverage curve of the rule list ABC in Example 6.4. This curve is dominated by the red coverage curve, corresponding to the rule tree in Figure 6.12. The rule tree also improves upon the rule set (orange curve in Figure 6.11), as it has access to exact coverage counts in all segments and thus recognises that AB- goes before - -C.

In summary, of the three rule models considered, only rule trees can unlock the full potential of rule overlap as they have the capacity to represent all 2^r overlap

areas of *r* rules and give access to exact coverage counts for each area. Rule lists also convey exact coverage counts but for fewer segments; rule sets distinguish the same segments as rule trees but have to estimate coverage counts for the overlap areas. On the other hand, rule trees are expensive as their size is exponential in the number of rules. Another disadvantage is that the coverage counts have to be obtained in a separate step, after the rules have been learned. I have included rule trees here mainly for conceptual reasons: to gain a better understanding of the more common rule list and rule set models.

6.3 Descriptive rule learning

As we have seen, the rule format lends itself naturally to predictive models, built from rules with the target variable in the head. It is not hard to come up with ways to extend rule models to regression and clustering tasks, in a similar way to what we did for tree models at the end of Chapter 5, but I will not elaborate on that here. Instead I will show how the rule format can equally easily be used to build descriptive models. As explained in Section 1.1, descriptive models can be learned in either a supervised or an unsupervised way. As an example of the supervised setting we will discuss how to adapt the given rule learning algorithms to subgroup discovery. For unsupervised learning of descriptive rule models we will take a look at frequent item sets and association rule discovery.

Figure 6.14. (top) A rule tree built on two rules X and Y. **(bottom)** The rule tree coverage curve strictly dominates the convex hull of the two rule list curves. This means that there is an operating point [2+, 0−] that cannot be achieved by either rule list.

Rule learning for subgroup discovery When learning classification models it is natural to look for rules that identify pure subsets of the training examples: i.e., sets of examples that are all of the same class and that all satisfy the same conjunctive concept. However, as we have seen in Section 3.3, sometimes we are less interested in predicting a class and more interested in finding interesting patterns. We defined subgroups as

mappings $\hat{g} : \mathscr{X} \rightarrow \{true, false\}$ – or alternatively, subsets of the instance space – that are learned from a set of labelled examples $(x_i, l(x_i))$, where $l : \mathscr{X} \rightarrow \mathscr{C}$ is the true labelling function. A good subgroup is one whose class distribution is significantly different from the overall population. This is by definition true for pure subgroups, but these are not the only interesting ones. For instance, one could argue that the complement of a subgroup is as interesting as the subgroup itself: in our dolphin example, the concept Gills = yes, which covers four negatives and no positives, could be considered as interesting as its complement Gills = no, which covers one negative and all positives. This means that we need to move away from impurity-based evaluation measures.

Like concepts, subgroups can be plotted as points in coverage space, with the positives in the subgroup on the *y*-axis and the negatives on the *x*-axis. Any subgroup plotted on the ascending diagonal has the same proportion of positives as the overall population; these are the least interesting subgroups as they have the same statistics as random samples. Subgroups above (below) the diagonal have a larger (smaller) proportion of positives than the population. So one way to measure the quality of subgroups is to take one of the heuristics used for rule learning and measure the absolute deviation from the default value on the diagonal. For example, the precision of any subgroup on the diagonal is equal to the proportion of positives, so this leads to $|prec - pos|$ as one possible quality measure. For reasons already discussed it is often better to use Laplace-corrected precision $prec^L$, leading to the alternative measure $|prec^L - pos|$. As can be seen in Figure 6.15 (left), the introduction of pseudo-counts means that [5+, 1−] is evaluated as [6+, 2−] and is thus as interesting as the pure concept [2+, 0−] which is evaluated as [3+, 1−].

However, this doesn't quite put complementary subgroups on an equal footing, as [5+, 1−] is still considered to be of lower quality than [0+, 4−]. In order to achieve this complementarity we need an evaluation measure whose isometrics all run parallel to the ascending diagonal. As it turns out, we have already seen such an evaluation measure in Section 2.1, where we called it *average recall* (see, e.g., Figure 2.4 on p.61). Notice that subgroups on the diagonal always have average recall 0.5, regardless of the class distribution. So, a good subgroup evaluation measure is $|avg\text{-}rec - 0.5|$. Average recall can be written as $(1 + tpr -$

fpr)/2, and thus |*avg-rec* – 0.5| = |*tpr* – *fpr*|/2. It is sometimes desirable not to take the absolute value, so that the sign of the difference tells us whether we are above or below the diagonal. A related subgroup evaluation measure is *weighted relative accuracy*, which can be written as $pos \cdot neg(tpr - fpr)$.

Figure 6.15. (left) Subgroups and their isometrics according to Laplace-corrected precision. The solid, outermost isometrics indicate the best subgroups. **(right)** The ranking changes if we order the subgroups on average recall. For example, [5+, 1−] is now better than [3+, 0−] and as good as [0+, 4−].

Subgroup	*Coverage*	$prec^L$	*Rank*	*avg-rec*	*Rank*
Gills = yes	[0+,4−]	0.17	1	0.10	1–2
Gills = no ∧ Teeth = many	[3+,0−]	0.80	2	0.80	3
Gills = no	[5+,1−]	0.75	3–9	0.90	1–2
Beak = no	[0+,2−]	0.25	3–9	0.30	4–11
Gills = yes ∧ Beak = yes	[0+,2−]	0.25	3–9	0.30	4–11
Length = 3	[2+,0−]	0.75	3–9	0.70	4–11
Length = 4 ∧ Gills = yes	[0+,2−]	0.25	3–9	0.30	4–11
Length = 5 ∧ Gills = no	[2+,0−]	0.75	3–9	0.70	4–11
Length = 5 ∧ Gills = yes	[0+,2−]	0.25	3–9	0.30	4–11
Length = 4	[1+,3−]	0.33	10	0.30	4–11
Beak = yes	[5+,3−]	0.60	11	0.70	4–11

Table 6.1. Detailed evaluation of the top subgroups. Using Laplace-corrected precision we can evaluate the quality of a subgroup as $|prec^L - pos|$. Alternatively, we can use average recall to define the quality of a subgroup as |*avg-rec*- – 0.5|. These two quality measures result in slightly different rankings.

As can be seen by comparing the two isometrics plots in Figure 6.15, using average recall rather than Laplace-corrected precision has an effect on the ranking of some of the subgroups. Detailed calculations are given in Table 6.1.

Subgroup	*Coverage*	*avg-rec*	*Wgtd coverage*	*W-avg-rec*	*Rank*
Gills = yes	[0+, 4−]	0.10	[0+, 3−]	0.07	1–2
Gills = no	[5+, 1−]	0.90	[4.5+, 0.5−]	0.93	1–2
Gills = no ∧ Teeth = many	[3+, 0−]	0.80	[2.5+, 0−]	0.78	3
Length = 5 ∧ Gills = yes	[0+, 2−]	0.30	[0+, 2−]	0.21	4
Length = 3	[2+, 0−]	0.70	[2+, 0−]	0.72	5–6
Length = 5 ∧ Gills = no	[2+, 0−]	0.70	[2+, 0−]	0.72	5–6
Beak = no	[0+, 2−]	0.30	[0+, 1.5−]	0.29	7–9
Gills = yes ∧ Beak = yes	[0+, 2−]	0.30	[0+, 1.5−]	0.29	7–9
Beak = yes	[5+, 3−]	0.70	[4.5+, 2−]	0.71	7–9
Length = 4	[1+, 3−]	0.30	[0.5+, 1.5−]	0.34	10
Length = 4 ∧ Gills = yes	[0+, 2−]	0.30	[0+, 1−]	0.36	11

Table 6.2. The 'Wgtd coverage' column shows how the weighted coverage of the subgroups is affected if theweights of the examples covered by Length = 4 are reduced to 1/2. 'W-*avg-rec*' shows how how the *avg-rec* numbers as calculated in Table 6.1 are affected by the weighting, leading to further differentiation between subgroups that were previously considered equivalent.

Example 6.6 (Comparing Laplace-corrected precision and average recall).

Table 6.1 ranks ten subgroups in the dolphin example in terms of Laplace-corrected precision and average recall. One difference is that Gills = no ∧ Teeth = many with coverage [3+, 0−] is better than Gills = no with coverage [5+, 1−] in terms of Laplace-corrected precision, but worse in terms of average recall, as the latter ranks it equally with its complement Gills = yes.

The second difference between classification rule learning and subgroup discovery is that in the latter case we are naturally interested in overlapping rules, whereas the standard covering algorithm doesn't encourage this as examples already covered are removed from the training set. One way of dealing with this is by assigning weights to examples that are decreased every time an example is covered by a newly learned rule. A scheme that works well in practice is to initialise the example weights to 1 and halve them every time a new rule covers the example. Search heuristics are then evaluated in terms of the cumulative weight of covered examples, rather than just their number.

Example 6.7 (The effect of weighted covering). Suppose the first subgroup found is Length = 4, reducing the weight of the one positive and three negatives covered by it to 1/2. Detailed calculations of how this affects the weighted coverage of subgroups are given in Table 6.2. We can see how the coverage space shrinks to the blue box in Figure 6.16. It also affects the weighted coverage of the subgroups overlapping with Length = 4, as indicated by the arrows. Some subgroups end up closer to the diagonal and hence lose importance: for instance, Length = 4 itself, which moves from [3+, 1−] to [1.5+, 0.5−]. Others move away from the diagonal and hence gain importance: for example Length = 5 ∧ Gills = yes at [0+, 2−].

Figure 6.16. Visualisation of the effect of weighted covering. If the first subgroup found is Length = 4, then this halves the weight of one positive and three negatives, shrinking the coverage space to the blue box. The arrows indicate how this affects the weighted coverage of other subgroups, depending on which of the reduced-weight examples they cover.

The *weighted covering* algorithm is given in Algorithm 6.5. Notice that this algorithm can be applied to discover subgroups over $k > 2$ classes, as long as the evaluation measure used to learn single rules can handle more than two classes. This is clearly the case for average recall used in our examples. Other

possibilities include measures derived from the Chi-squared test and mutual information-based measures.

Association rule mining I will now introduce a new kind of rule that can be learned in a wholly unsupervised manner and is prominent in data mining applications. Suppose we observed eight customers who each bought one or more of apples, beer, crisps and nappies:

Transaction	*Items*
1	nappies
2	beer, crisps
3	apples, nappies
4	beer, crisps, nappies
5	apples
6	apples, beer, crisps, nappies
7	apples, crisps
8	crisps

Each *transaction* in this table involves a set of *items*; conversely, for each item we can list the transactions in which it was involved: transactions 1, 3, 4 and 6 for nappies, transactions 3, 5, 6 and 7 for apples, and so on. We can also do this for sets of items: e.g., beer and crisps were bought together in transactions 2, 4 and 6; we say that item set {beer, crisps} *covers* transaction set {2, 4, 6}. There are 16 of such item sets (including the empty set, which covers all transactions); using the subset relation between transaction sets as partial order, they form a lattice (Figure 6.17).

Let us call the number of transactions covered by an item set I its *support*, denoted Supp(I) (sometimes called frequency). We are interested in *frequent item sets*, which exceed a given support threshold f_0. Support is *monotonic*: when moving down a path in the item set lattice it can never increase. This means that the set of frequent item sets is *convex* and is fully determined by its lower boundary of largest item sets: in the example these maximal[1] frequent item sets are, for $f_0 = 3$: {apples}, {beer, crisps} and {nappies}. So, at least three transactions involved apples; at least three involved nappies; at least three

involved both beer and crisps; and any other combination of items was bought less often.

Algorithm 6.5: WeightedCovering(D) – learn overlapping rules by weighting examples.

```
R ←∅;
while some examples in D have weight 1 do
    r ←LearnRule(D) ;                    // LearnRule: see Algorithm 6.2
    append r to the end of R;
    decrease the weights of examples covered by r;
end
return R
```

Figure 6.17. An item set lattice. Item sets in dotted ovals cover a single transaction; in dashed ovals, two transactions; in triangles, three transactions; and in polygons with n sides, n transactions. The maximal item sets with support 3 or more are indicated in green.

Because of the monotonicity property of item set support, frequent item sets can be found by a simple enumerative breadth-first or level-wise search algorithm (Algorithm 6.6). The algorithm maintains a priority queue, initially holding only the empty item set which covers all transactions. Taking the next candidate item set I off the priority queue, it generates all its possible extensions (supersets containing one more item, the downward neighbours in the item set lattice), and adds them to the priority queue if they exceed the support threshold (at the back, to achieve the desired breadth-first behaviour). If at least one of I's extensions is frequent, I is not maximal and can be discarded; otherwise I is

added to the set of maximal frequent item sets found.

We can speed up calculations by restricting attention to *closed item sets*. These are completely analogous to the ☞*closed concepts* discussed at the end of Section 4.2: a closed item set contains all items that are involved in every transaction it covers. For example, {beer, crisps} covers transactions 2,4 and 6; the only items involved in each of those transactions are beer and crisps, and so the item set is closed. However, {beer} is not closed, as it covers the same transactions, hence its closure is {beer, crisps}. If two item sets that are connected in the lattice have the same coverage, the smaller item set cannot be closed. The lattice of closed item sets is shown in Figure 6.18. Notice that maximal frequent item sets are necessarily closed (as extending them will decrease their coverage below the support threshold, otherwise they aren't maximal), and are thus unaffected by this restriction; but it does allow a more efficient search.

So what is the point of these frequent item sets? The answer is that we will use them to build *association rules*, which are rules of the form ·**if** B **then** H· where both body B and head H are item sets that frequently appear in transactions together. Pick any edge in Figure 6.17, say the edge between {beer} and {nappies, beer}. We know that the support of the former is 3 and of the latter, 2: that is, three transactions involve beer and two of those involve nappies as well. We say that the *confidence* of the association rule ·**if** beer **then** nappies· is 2/3. Likewise, the edge between {nappies} and {nappies, beer} demonstrates that the confidence of the rule ·**if** nappies **then** beer· is 2/4. There are also rules with confidence 1, such as ·**if** beer **then** crisps·; and rules with empty bodies, such as ·**if** true **then** crisps·, which has confidence 5/8 (i.e., five out of eight transactions involve crisps).

But we only want to construct association rules that involve frequent items. The rule ·**if** beer ∧ apples **then** crisps· has confidence 1, but there is only one transaction involving all three and so this rule is not strongly supported by the data. So we first use Algorithm 6.6 to mine for frequent item sets; we then select bodies B and heads H from the frequent sets m, discarding rules whose confidence is below a given confidence threshold. Algorithm 6.7 gives the basic algorithm. Notice that we are free to discard some of the items in the maximal frequent sets (i.e., $H \cup B$ may be smaller than m), because any subset of a frequent item set is frequent as well.

Algorithm 6.6: FrequentIiems(D, f_0) – find all maximal item sets exceeding a given support threshold.

```
Input  : data D ⊆ 𝒳; support threshold f0.
Output : set of maximal frequent item sets M.
M ← ∅;
initialise priority queue Q to contain the empty item set;
while Q is not empty do
    I ← next item set deleted from front of Q;
    max ← true ;                    // flag to indicate whether I is maximal
    for each possible extension I' of I do
        if Supp(I') ≥ f0 then
            max ← false ;           // frequent extension found, so I is not maximal
            add I' to back of Q;
        end
    end
    if max = true then M ← M ∪ {I};
end
return M
```

Figure 6.18. Closed item set lattice corresponding to the item sets in Figure 6.17. This lattice has the property that no two adjacent item sets have the same coverage.

A run of the algorithm with support threshold 3 and confidence threshold 0.6 gives the following association rules:

·**if** beer **then** crisps·	support 3, confidence 3/3
·**if** crisps **then** beer·	support 3, confidence 3/5

·**if** true **then** crisps· support 5, confidence 5/8

Algorithm 6.7: AssociationRules(D, f_0, c_0) – find all association rules exceeding given support and confidence thresholds.

Input : data $D \subseteq \mathscr{X}$; support threshold f_0; confidence threshold c_0.
Output : set of association rules R.

```
R ← ∅;
M ← FrequentItems(D, f0) ;                    // FrequentItems: see Algorithm 6.6
for each m ∈ M do
    for each H ⊆ m and B ⊆ m such that H ∩ B = ∅ do
        if Supp(B ∪ H)/Supp(B) ≥ c0 then R ← R ∪ {·if B then H·}
    end
end
return R
```

Association rule mining often includes a *post-processing* stage in which superfluous rules are filtered out, e.g., special cases which don't have higher confidence than the general case. One quantity that is often used in post-processing is *lift*, defined as $\text{Lift}(\cdot\textbf{if } B \textbf{ then } H\cdot) = \frac{n \cdot \text{Supp}(B \cup H)}{\text{Supp}(B) \cdot \text{Supp}(H)}$ where n is the number of transactions. For example, for the the first two association rules above we would have lifts of $\frac{8 \cdot 3}{3 \cdot 5} = 1.6$, as Lift(·**if** B **then** H·) = Lift(·**if** H **then** B·). For the third rule we have Lift(·**if** true **then** crisps·) $= \frac{8 \cdot 5}{8 \cdot 5} = 1$. This holds for any rule with $B = \emptyset$, as $\text{Lift}(\cdot\textbf{if } \emptyset \textbf{ then } H\cdot) = \frac{n \cdot \text{Supp}(\emptyset \cup H)}{\text{Supp}(\emptyset) \cdot \text{Supp}(H)} = \frac{n \cdot \text{Supp}(H)}{n \cdot \text{Supp}(H)} = 1$ More generally, a lift of 1 means that Supp($B \cup H$) is entirely determined by the *marginal* frequencies Supp(B) and Supp(H) and is not the result of any meaningful interaction between B and H. Only association rules with lift larger than 1 are of interest.

Quantities like confidence and lift can also be understood from a probabilistic context. Let Supp(I)/n be an estimate of the probability $p(I)$ that a transaction involves all items in I, then confidence estimates the conditional probability $p(H|B)$. In a classification context, where H denotes the actual class and B the predicted class, this would be called precision (see Table 2.3 on p.57), and in this chapter we have already used it as a search heuristic in rule learning. Lift then measures whether the events 'a random transaction involves all items in B' and

‘a random transaction involves all items in *H*’ are statistically independent.

It is worth noting that the heads of association rules can contain multiple items. For instance, suppose we are interested in the rule ·**if** nappies **then** beer·, which has support 2 and confidence 2/4. However, {nappies, beer} is not a closed item set: its closure is {nappies, beer, crisps}. So ·**if** nappies **then** beer· is actually a special case of ·**if** nappies **then** beer ∧ crisps·, which has the same support and confidence but involves only closed item sets.

We can also apply frequent item set analysis to our dolphin data set, if we treat each literal Feature = Value as an item, keeping in mind that different values of the same feature are mutually exclusive. Item sets then correspond to concepts, transactions to instances, and the extension of a concept is exactly the set of transactions covered by an item set. The item set lattice is therefore the same as what we previously called the hypothesis space, with the proviso that we are not considering negative examples in this scenario (Figure 6.19). The reduction to closed concepts/item sets is shown in Figure 6.20. We can see that, for instance, the rule ·**if** Gills = no ∧ Beak = yes **then** Teeth = many·

has support 3 and confidence 3/5 (but you may want to check whether it has any lift!).

Figure 6.19. The item set lattice corresponding to the positive examples of the dolphin example in Example 4.4 on p.115. Each ‘item’ is a literal Feature = Value; each feature can occur at most once in an item set. The resulting structure is exactly the same as what was called the hypothesis space in Chapter 4.

Figure 6.20. Closed item set lattice corresponding to the item sets in Figure 6.19.

6.4 First-order rule learning In Section 4.3 we briefly touched upon using first-order logic as a concept language. The main difference is that literals are no longer simple feature-value pairs but can have a much richer structure. All rule learning approaches covered in this chapter have been upgraded in the literature to learn rules expressed in first-order logic. In this section we will take a brief look at how this might work.

Many approaches to learning in first-order logic are based on the logic programming language Prolog, and learning first-order rules is often called *inductive logic programming* (*ILP*). Logically speaking, Prolog rules are *Horn clauses* with a single literal in the head – we encountered Horn clauses before in Section 4.3. Prolog notation is slightly different from first-order logic notation. So, instead of $\forall x$: BodyPart (x, PairOf (Gill)) → Fish(x) we write

```
fish(X):-bodyPart (X, pairOf(gills)).
```

The main differences are:

- rules are written back-to-front in 'head-if-body' fashion;
- variables start with a capital letter; constants, predicates and function symbols (called *functors* in Prolog) start with lower-case;

☞ variables are implicitly universally quantified.

With regard to the third point, it is worth pointing out the difference between variables occurring in both head and body, and variables occurring in the body only. Consider the following Prolog clause: `someAnimal (X):-bodyPart (X, pairOf(Y)).`

There are two equivalent ways of writing this rule in first-order logic: ∀x : ∀y : BodyPart (x, PairOf (y)) → SomeAnimal(x) ∀x : (∃y : BodyPart (x, PairOf (y))) → SomeAnimal(x) The first logical statement reads 'for all *x* and *y*, if *x* has a pair of *y*s as body parts then *x* is some kind of animal' whereas the second states 'for all *x*, if there exists a *y* such that *x* has a pair of *y*s as body parts then *x* is some kind of animal'. Crucially, in the second form the scope of the existential quantifier is the if-part of the rule, whereas universal quantifiers always range over the whole clause. Variables occurring in the body but not in the head of Prolog clauses are called *local variables*; they are the source of much additional complexity in learning first-order rules over propositional rules.

If we want to learn an ordered list of Prolog clauses, we can reuse ☞ *LearnRuleList* (Algorithm 6.1 on p.163) in its entirety and most of ☞*LearnRule* (Algorithm 6.2 on p.164). What needs adjusting is the choice of literal to be added to the clause. Possible literals can be enumerated by listing the predicates, functors and constants that can be used to build a new literal. For example, if we have a binary predicate `bodyPart`, a unary functor `pairOf` and constants `gill` and `tail`, then we can build a variety of literals such as `bodyPart (X, Y)`

```
bodyPart (X, gill)
bodyPart (X, tail)
bodyPart (X, pairOf (Y) )
bodyPart (X, pairOf (gill) )
bodyPart (X, pairOf (tail) )
bodyPart (X, pairOf (pairOf (Y) ) )
bodyPart (X, pairOf (pairOf (gill) ) )
bodyPart (X, pairOf(pairOf (tail) ) )
```

and so on. Notice that the presence of functors means that our hypothesis language becomes infinite! Also, I have only listed literals that somehow 'made sense': there are many less sensible possibilities, such as `bodyPart (pairOf (gill), tail)` or `bodyPart (X, X)`, to name but a few. Although Prolog is an untyped language, many of these unwanted literals can be excluded by adding type information (in logic programming and ILP often done through 'mode declarations' which also specify particular input–output patterns of a predicate's arguments).

It is clear from these examples that there can be relationships between literals,

and therefore between the clauses that contain them. For example, consider the following three clauses:

```
fish(X):-bodyPart (X, Y).
fish(X):-bodyPart (X, pairOf (Z) ).
fish(X):-bodyPart (X, pairOf (gill) ).
```

The first clause defines everything with some body part to be a fish. The second clause specialises this to everything with a pair of unspecified body parts. The third specialises this to everything with a pair of gills. A reasonable search strategy would be to try hypotheses in this order, and only move on to a specialised version if the more general clause is ruled out by negative examples. This is what in fact happens in top–down ILP systems. A simple trick is to represent substitution of terms for variables explicitly by adding equality literals, so the above sequence of clauses becomes

```
fish(X):-bodyPart (X, Y).
fish(X):-bodyPart (X, Y), Y = pairOf (Z).
fish(X):-bodyPart (X, Y), Y = pairOf (Z), Z=gill.
```

As an alternative for enumerating the literals to be considered for inclusion in a clause body we can derive them from the data in a bottom–up fashion. Suppose we have the following information about a dolphin: `bodyPart (dolphin42, tail).`

```
bodyPart (dolphin42, pairOf (gills) ).
bodyPart (dolphin42, pairOf (eyes) ).
```

and this about a tunafish: `bodyPart (tuna123, pairOf (gills) ).`

By forming the LGG of each of the literals in the first example with the literal from the second example we obtain each of the generalised literals considered earlier.

This short discussion of rule learning in first-order logic has left out many important details and may therefore give an overly simplified view of the problem. While the problem of learning Prolog clauses can be stated quite succinctly, naive approaches are computationally intractable and ‘the devil is in the detail’. The basic approaches sketched here can be extended to include background knowledge, which then affects the generality ordering of the hypothesis space. For example, if our background knowledge includes the clause `bodyPart (X, scales):-bodyPart (X, pairOf (gill) ).`

then the first of the following two hypotheses is more general than the second:

```
fish(X):-bodyPart (X, scales).
fish(X):-bodyPart (X, pairOf (gill) ).
```

However, this cannot be determined purely by syntactic means and requires logical inference.

Another intriguing possibility offered by first-order logic is the possibility of learning recursive clauses. For instance, part of our hypothesis could be the following clause:

```
fish(X):-relatedSpecies(X, Y), fish (Y).
```

This blurs the distinction between background predicates that can be used in the body of hypotheses and target predicates that are to be learned, and introduces computational challenges such as non-termination. However, this doesn't mean that it cannot be done. Related techniques can be used to learn multiple, interrelated predicates at once, and to invent new background predicates that are completely unobserved.

6.5 Rule models: Summary and further reading In a decision tree, a branch from root to a leaf can be interpreted as a conjunctive classification rule. Rule models generalise this by being more flexible about the way in which several rules are combined into a model. The typical rule learning algorithm is the covering algorithm, which iteratively learns one rule and then removes the examples covered by that rule. This approach was pioneered by Michalski (1975) with his AQ system, which became highly developed over three decades (Wojtusiak *et al.*, 2006). General overviews are provided by Fürnkranz (1999,

2010) and Fürnkranz, Gamberger and Lavrač (2012). Coverage plots were first used by Fürnkranz and Flach (2005) to achieve a better understanding of rule learning algorithms and demonstrate the close relationship (and in many cases, equivalence) of commonly used search heuristics.

- Rules can overlap and thus we need a strategy to resolve potential conflict between rules. One such strategy is to combine the rules in an ordered rule list, which was the subject of Section 6.1. Rivest (1987) compares this approach with decision trees, calling the rule-based model a decision list (prefer the term 'rule list' as it doesn't carry a suggestion that the elements of the list are single literals). Well-known rule list learners include CN2 (Clark and Niblett, 1989) and Ripper (Cohen, 1995), the latter being particularly effective at avoiding overfitting through incremental reduced-error pruning (Fürnkranz and Widmer, 1994). Also notable is the Opus system (Webb, 1995), which distinguishes itself by performing a complete search through the space of all possible rules.
- In Section 6.2 we looked at unordered rule sets as an alternative to ordered rule lists. The covering algorithm is adapted to learn rules for a single class at a time, and to remove only covered examples of the class currently under consideration. CN2 can be run in unordered mode to learn rule sets (Clark and Boswell, 1991). Conceptually, both rule lists and rule sets are special cases of rule trees, which distinguish all possible Boolean combinations of a given set of rules. This allows us to see that rule lists lead to fewer instance space segments than rule sets (over the set of rules); on the other hand, rule list coverage curves can be made convex on the training set, whereas rule sets need to estimate the class distribution in the regions where rules overlap.
- Rule models can be used for descriptive tasks, and in Section 6.3 we considered rule learning for subgroup discovery. The weighted covering algorithm was introduced as an adaption of CN2 by Lavrač, Kavšek, Flach

and Todorovski (2004); Abudawood and Flach (2009) generalise this to more than two classes. Algorithm 6.7 learns association rules and is adapted from the well-known Apriori algorithm due to Agrawal, Mannila, Srikant, Toivonen and Verkamo (1996). There is a very wide choice of alternative algorithms, surveyed by Han *et al.* (2007). Association rules ca also be used to build effective classifiers (Liu *et al.*, 1998; Li *et al.*, 2001).

The topic of first-order rule learning briefly considered in Section 6.4 has been studied for the last 40 years and has a very rich history. De Raedt (2008) provides an excellent recent introduction, and an overview of recer advances and open problems is provided by Muggleton *et al.* (2012). Flac (1994) gives an introduction to Prolog and also provides high-level implementations of some of the key techniques in inductive logic programming. The FOIL system by Quinlan (1990) implements a top–down learning algorithm similar to the one discussed here. The bottom–up technique was pioneered in the Golem system (Muggleton and Feng, 1990 and further refined in Progol (Muggleton, 1995) and in Aleph (Srinivasan, 2007), two of the most widely used ILP systems. First-order rules can also be learned in an unsupervised fashion, for example by Tertius which learn first-order clauses (not necessarily Horn) (Flach and Lachiche, 2001) and Warmr which learns first-order association rules (King *et al.*, 2001). Higher-order logic provides more powerful data types that can be highly beneficial in learning (Lloyd, 2003). A more recent development is the combination of probabilistic modelling with first-order logic, leading to th area of statistical relational learning (De Raedt and Kersting, 2010).

---

[1] 'Maximal' here means that no superset is frequent.

CHAPTER 7

Linear models

AFTER DEALING WITH logical models in the preceding chapters we now move on to a quite different kind of model. The models in this chapter and the next are defined in terms of the geometry of instance space. Geometric models most often assume that instances are described by d real-valued features, and thus $\mathscr{X} = \mathbb{R}^d$. For example, we could describe objects by their position on a map in terms of longitude and latitude ($d = 2$), or in the real world by longitude, latitude and altitude ($d = 3$). While most real-valued features are not intrinsically geometric – think of a person's age or an object's temperature – we can still imagine them being plotted in a d-dimensional Cartesian coordinate system. We can then use geometric concepts such as lines and planes to impose structure on this space, for instance in order to build a classification model. Alternatively, we can use the geometric notion of distance to represent similarity, on the basis that if two points are close together they have similar feature values and thus can be expected to behave similarly with respect to the property of interest. Such distance-based models are the subject of the next chapter. In this chapter we will look at models that can be understood in terms of lines and planes, commonly called *linear models*.

Linearity plays a fundamental role in mathematics and related disciplines, and the mathematics of linear models is well-understood (see Background 7.1 for the most important concepts). In machine learning, linear models are of particular interest because of their simplicity (remember our rule of thumb 'everything should be made as simple as possible, but not simpler' that we introduced on p.30). Here are a couple of manifestations of this simplicity.

If x_1 and x_2 are two scalars or vectors of the same dimension and α and β are arbitrary scalars, then $\alpha x_1 + \beta x_2$ is called a *linear combination* of x_1 and x_2. If f is a *linear function* of x, then

$$f(\alpha x_1 + \beta x_2) = \alpha f(x_1) + \beta f(x_2)$$

In words, the function value of a linear combination of some inputs is a linear combination of their function values. As a special case, if $\beta = 1 - \alpha$ we are taking a weighted average of x_1 and x_2, and the linearity of f then means that the function value of the weighted average is the weighted average of the function values.

Linear functions take particular forms, depending on the domain and codomain of f. If x and $f(x)$ are scalars, it follows that f is of the form $f(x) = a + bx$ for some constants a and b; a is called the *intercept* and b the *slope*. If $\mathbf{x} = (x_1, \ldots, x_d)$ is a vector and $f(\mathbf{x})$ is a scalar, then f is of the form

$$f(\mathbf{x}) = a + b_1 x_1 + \ldots + b_d x_d = a + \mathbf{b} \cdot \mathbf{x} \tag{7.1}$$

with $\mathbf{b} = (b_1, \ldots, b_d)$. The equation $f(\mathbf{x}) = 0$ defines a plane in $\mathbb{R}^d$ perpendicular to the *normal vector* $\mathbf{b}$.

The most general case is where $f(\mathbf{x})$ is a d'-dimensional vector, in which case f is of the form $f(\mathbf{x}) = \mathbf{Mx} + \mathbf{t}$, where $\mathbf{M}$ is a d'-by-d matrix representing a *linear transformation* such as a rotation or a scaling, and $\mathbf{t}$ is a d'-vector representing a translation. In this case f is called an *affine transformation* (the difference between linear and affine transformations is that the former maps the origin to itself; notice that a linear function of the form of Equation 7.1 is a linear transformation only if the intercept is 0).

In all these forms we can avoid representing the intercept a or the translation $\mathbf{t}$ separately by using homogeneous coordinates. For instance, by writing $\mathbf{b}^\circ = (a, b_1, \ldots, b_d)$ and $\mathbf{x}^\circ = (1, x_1, \ldots, x_d)$ in Equation 7.1 we have $f(\mathbf{x}) = \mathbf{b}^\circ \cdot \mathbf{x}^\circ$ (see also Background 1.2 on p.24).

Examples of non-linear functions are the polynomials in x of degree $p > 1$: $g(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_p x^p = \sum_{i=0}^{p} a_i x^i$. Other non-linear functions can be approximated by a polynomial through their Taylor expansion. The *linear approximation* of a function g at x_0 is $g(x_0) + g'(x_0)(x - x_0)$, where $g'(x)$ is the derivative of x. A *piecewise linear* approximation is obtained by combining several linear approximations at different points x_0.

Background 7.1. Linear models.

Linear models are *parametric*, meaning that they have a fixed form with a small number of numeric parameters that need to be learned from data. Th is different from tree or rule models, where the structure of the model (e.g

which features to use in the tree, and where) is not fixed in advance.

- Linear models are stable, which is to say that small variations in the training data have only limited impact on the learned model. Tree models tend to vary more with the training data, as the choice of a different split a the root of the tree typically means that the rest of the tree is different as well.

- Linear models are less likely to overfit the training data than some other models, largely because they have relatively few parameters. The flipside of this is that they sometimes lead to *underfitting*: e.g., imagine you are learning where the border runs between two countries from labelled samples, then a linear model is unlikely to give a good approximation.

The last two points can be summarised by saying that linear models have low variance but high bias. Such models are often preferable when you have limited data and want to avoid overfitting. High variance–low bias models such as decision trees are preferable if data is abundant but underfitting is a concern. It is usually a good idea to start with simple, high-bias models such as linear models and only move on to more elaborate models if the simpler ones appear to be underfitting.

Linear models exist for all predictive tasks, including classification, probability estimation and regression. Linear regression, in particular, is a well-studied problem that can be solved by the leastsquares method, which is the topic of the next section. We will look at a number of other linear models in this chapter, including leastsquares classification (also in Section 7.1), the perceptron in Section 7.2, and the support vector machine in Section 7.3. We will also find out how these models can be turned into probability estimators in Section 7.4. Finally, Section 7.5 briefly discusses how each of these methods could learn non-linear models by means of so-called kernel functions.

7.1 The leastsquares method

We start by introducing a method that can be used to learn linear models for classification and regression. Recall that the regression problem is to learn a

function estimator $\hat{f} : \mathscr{X} \to \mathbb{R}$ from examples $(x_i, f(x_i))$, where in this chapter we assume $\mathscr{X} = \mathbb{R}^d$. The differences between the actual and estimated function values on the training examples are called *residuals* $\epsilon_i = f(x_i) - \hat{f}(x_i)$. The *leastsquares method*, introduced by Carl Friedrich Gauss in the late eighteenth century, consists in finding $\hat{f}$ such that $\sum_{i=1}^{n} \epsilon_i^2$ is minimised. The following example illustrates the method in the simple case of a single feature, which is called *univariate regression*.

Figure 7.1. The red solid line indicates the result of applying linear regression to 10 measurements of body weight (on the y-axis, in kilograms) against body height (on the x-axis, in centimetres). The orange dotted lines indicate the average height $\overline{h} = 181$ and the average weight $\overline{w} = 74.5$; the regression coefficient $\hat{b} = 0.78$. The measurements were simulated by adding normally distributed noise with mean 0 and variance 5 to the true model indicated by the blue dashed line ($b = 0.83$).

Example 7.1 (Univariate linear regression). Suppose we want to investigate the relationship between people's height and weight. We collect n height and weight measurements (h_i, w_i), $1 \leq i \leq n$. Univariate linear regression assumes a linear equation $w = a + bh$, with parameters a and b chosen such that the sum of squared residuals $\sum_{i=1}^{n} (w_i - (a + bh_i))^2$ is minimised. In order to find the parameters we take partial derivatives of this expression, set the partial derivatives to 0 and solve for a and b:

$$\frac{\partial}{\partial a}\sum_{i=1}^{n}(w_i-(a+bh_i))^2=-2\sum_{i=1}^{n}(w_i-(a+bh_i))=0 \qquad \Rightarrow \hat{a}=\overline{w}-\hat{b}\overline{h}$$

$$\frac{\partial}{\partial b}\sum_{i=1}^{n}(w_i-(a+bh_i))^2=-2\sum_{i=1}^{n}(w_i-(a+bh_i))h_i=0$$

$$\Rightarrow \hat{b}=\frac{\sum_{i=1}^{n}(h_i-\overline{h})(w_i-\overline{w})}{\sum_{i=1}^{n}(h_i-\overline{h})^2}$$

So the solution found by linear regression is $w = \hat{a} + \hat{b}h = \overline{w} + (h - \overline{h})$; see Figure 7.1 for an example.

It is worthwhile to note that the expression for the *regression coefficient* or slope $\hat{b}$ derived in this example has n times the covariance between h and w in the enumerator and n times the variance of h in the denominator. This is true in general: for a feature x and a target variable y, the regression coefficient is

$$\hat{b}=n\frac{\sigma_{xy}}{n\sigma_{xx}}=\frac{\sigma_{xy}}{\sigma_{xx}}$$

(Here I use σ_{xx} as an alternative notation for σ^2_x, the variance of variable x.) This can be understood by noting that the covariance is measured in units of x times units of y (e.g., metres times kilograms in Example 7.1) and the variance in units of x squared (e.g., metres squared), so their quotient is measured in units of y per unit of x (e.g., kilograms per metre).

We can notice a few more useful things. The intercept $\hat{a}$ is such that the regression line goes through $(\overline{x}, \overline{y})$. Adding a constant to all x-values (a translation) will affect only the intercept but not the regression coefficient (since it is defined in terms of deviations from the mean, which are unaffected by a translation). So we could *zero-centre* the x-values by subtracting $\overline{x}$, in which case the intercept is equal to $\overline{y}$. We could even subtract $\overline{y}$ from all y-values to achieve a zero intercept, without changing the problem in an essential way.

Furthermore, suppose we replace x_i with $x_i = x'_i / \sigma_{xx}$ and likewise $\overline{x}$ with $\overline{x'} = \overline{x}/\sigma_{xx}$, then we have that $\hat{b}=\frac{1}{n}\sum_{i=1}^{n}(x'_i-\overline{x'})(y_i-\overline{y})=\sigma_{x'y}$. In other words, if we *normalise* x by dividing all its values by x's variance, we can take the covariance between the normalised feature and the target variable as regression coefficient. In other words, univariate linear regression can be understood as consisting of two steps:

1. normalisation of the feature by dividing its values by the feature's variance;

2. calculating the covariance of the target variable and the normalised feature.

We will see below how these two steps change when dealing with more than one feature.

Another important point to note is that the sum of the residuals of the leastsquares solution is zero:

$$\sum_{i=1}^{n}(y_i - (\hat{a} + \hat{b}x_i)) = n(\overline{y} - \hat{a} - \hat{b}\overline{x}) = 0$$

The result follows because $\hat{a} = \overline{y} - \hat{b}\overline{x}$, as derived in Example 7.1. While this property is intuitively appealing, it is worth keeping in mind that it also makes linear regression susceptible to *outliers*: points that are far removed from the regression line, often because of measurement errors.

Figure 7.2. The effect of an outlier in univariate regression. One of the blue points got moved up 10 units to the green point, changing the red regression line to the green line.

Example 7.2 (The effect of outliers). Suppose that, as the result of a

transcription error, one of the weight values in Figure 7.1 is increased by 10 kg. Figure 7.2 shows that this has a considerable effect on the leastsquares regression line.

Despite this sensitivity to outliers, the leastsquares method usually works surprisingly well for such a simple method. How can it be justified? One way to look at this is to assume that the true function f is indeed linear, but that the observed y-values are contaminated with random noise. That is, our examples are $(x_i, f(x_i) + \epsilon_i)$ rather than $(x_i, f(x_i))$, and we assume that $f(x) = ax + b$ for some a and b. If we knew a and b we could work out exactly what the residuals are, and if we knew σ^2 we could calculate the probability of observing that set of residuals. Since we don't know a and b we have to estimate them, and the estimate we want is the value of a and b that maximises the probability of the residuals. We will see in Chapter 9 that this so-called ☞*maximum-likelihood estimate* is exactly the leastsquares solution.

Variants of the leastsquares method exist. Here we discussed *ordinary* least squares, which assumes that only the y-values are contaminated with random noise. *Total* least squares generalises this to the situation that both x- and y-values are noisy, but this does not necessarily have a unique solution.

X usually denotes an n-by–d data matrix containing n instances in rows described by d features or variables in columns. $\mathbf{X}_{r\cdot}$ denotes the r-th row of **X**, $\mathbf{X}_{\cdot c}$ denotes the c-th column, and $\mathbf{X}_{rc}$ denotes the entry in the r-th row and c-th column. We also use i and j to range over rows and columns, respectively. The j-th column mean is defined as $\mu_j = \frac{1}{n}\sum_{i=1}^{n} \mathbf{X}_{ij}$; μ^{T} is a row vector containing all column means. If **1** is an n-vector containing only ones, then is $\mathbf{1}\mu^{\mathrm{T}}$ an n-by-d matrix whose rows are μ^{T}; hence $\mathbf{X'} = \mathbf{X} - \mathbf{1}\mu^{\mathrm{T}}$ has mean zero in each column and is referred to as the *zero-centred* data matrix.

The *scatter matrix* is the d-by-d matrix $\mathbf{S} = \mathbf{X}'^{\mathrm{T}}\mathbf{X}' = \left(\mathbf{X} - \mathbf{1}\mu^{\mathrm{T}}\right)^{\mathrm{T}}\left(\mathbf{X} - \mathbf{1}\mu^{\mathrm{T}}\right) = \mathbf{X}^{\mathrm{T}}\mathbf{X} - n\mathbf{M}$, where $\mathbf{M} = \mu\mu^T$ is a d-by-d matrix whose entries are products of column means $\mathbf{M}_{jc} = \mu_j\ \mu_c$. The

covariance matrix of $\mathbf{X}$ is $\mathbf{\Sigma} = \frac{1}{n}\mathbf{S}$ whose entries are the pairwise covariances $\sigma_{jc} = \frac{1}{n}\sum_{i=1}^{n}\left(\mathrm{X}_{ij} - \mu_j\right)\left(\mathrm{X}_{ic} - \mu_c\right) = \frac{1}{n}\left(\sum_{i=1}^{n}\mathrm{X}_{ij}\mathrm{X}_{ic} - \mu_i\mu_c\right)$. Two uncorrelated features have a covariance close to 0; positively correlated features have a positive covariance, indicating a certain tendency to increase or decrease together; a negative covariance indicates that if one feature increases, the other tends to decrease and vice versa. $\sigma_{jj} = \frac{1}{n}\sum_{i=1}^{n}\left(\mathrm{X}_{ij} - \mu_j\right)^2 = \frac{1}{n}\left(\sum_{i=1}^{n}\mathrm{X}_{ij}^2 - \mu_j^2\right)$ is is the *variance* of column j, also denoted as σ^2_j. The variance is always positive and indicates the spread of the values of a feature around their mean.
A small example clarifies these definitions:

$$\mathbf{X} = \begin{pmatrix} 5 & 0 \\ 3 & 5 \\ 1 & 7 \end{pmatrix} \quad \mathbf{1}\mu^{\mathrm{T}} = \begin{pmatrix} 3 & 4 \\ 3 & 4 \\ 3 & 4 \end{pmatrix} \quad \mathbf{X}' = \begin{pmatrix} 2 & -4 \\ 0 & 1 \\ -2 & 3 \end{pmatrix} \quad \mathbf{G} = \begin{pmatrix} 25 & 15 & 5 \\ 15 & 34 & 38 \\ 5 & 38 & 50 \end{pmatrix}$$

$$\mathbf{X}^{\mathrm{T}}\mathbf{X} = \begin{pmatrix} 35 & 22 \\ 22 & 74 \end{pmatrix} \quad \mathbf{M} = \begin{pmatrix} 9 & 12 \\ 12 & 16 \end{pmatrix} \quad \mathbf{S} = \begin{pmatrix} 8 & -14 \\ -14 & 26 \end{pmatrix} \quad \mathbf{\Sigma} = \begin{pmatrix} 8/3 & -14/3 \\ -14/3 & 26/3 \end{pmatrix}$$

We see that the two features are negatively correlated and that the second feature has the larger variance. Another way to calculate the scatter matrix is as a sum of outer products, one for each data point: $\mathbf{S} = \sum_{i=1}^{n}\left(\mathbf{X}_{i\cdot} - \mu^{\mathrm{T}}\right)^{\mathrm{T}}\left(\mathbf{X}_{i\cdot} - \mu^{\mathrm{T}}\right)$. In our example we have

$$\left(\mathbf{X}_{1\cdot} - \mu^{\mathrm{T}}\right)^{\mathrm{T}}\left(\mathbf{X}_{1\cdot} - \mu^{\mathrm{T}}\right) = \begin{pmatrix} 2 \\ -4 \end{pmatrix}\begin{pmatrix} 2 & -4 \end{pmatrix} = \begin{pmatrix} 4 & -8 \\ -8 & 16 \end{pmatrix}$$

$$\left(\mathbf{X}_{2\cdot} - \mu^{\mathrm{T}}\right)^{\mathrm{T}}\left(\mathbf{X}_{2\cdot} - \mu^{\mathrm{T}}\right) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}\begin{pmatrix} 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\left(\mathbf{X}_{3\cdot} - \mu^{\mathrm{T}}\right)^{\mathrm{T}}\left(\mathbf{X}_{3\cdot} - \mu^{\mathrm{T}}\right) = \begin{pmatrix} -2 \\ 3 \end{pmatrix}\begin{pmatrix} -2 & 3 \end{pmatrix} = \begin{pmatrix} 4 & -6 \\ -6 & 9 \end{pmatrix}$$

Background 7.2. Some more matrix notation.

Multivariate linear regression

In order to deal with an arbitrary number of features it will be useful to employ

matrix notation (see Background 7.2). We can write univariate linear regression in matrix form as

$$\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} a + \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} b + \begin{pmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{pmatrix}$$

$$\mathbf{y} = \mathbf{a} + \mathbf{X}\mathbf{b} + \epsilon$$

In the second form of this equation, **y, a, X** and ϵ are n-vectors, and **b** is a scalar. In case of d features, all that changes is that **X** becomes an n-by-d matrix, and **b** becomes a d-vector of regression coefficients.

We can apply the by now familiar trick of using homogeneous coordinates to simplify these equations as follows:

$$\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{pmatrix}$$

$$\mathbf{y} = \mathbf{X}^\circ \mathbf{w} + \epsilon$$

with $\mathbf{X}^\circ$ an n-by-$(d + 1)$ matrix whose first column is all 1s and the remaining columns are the columns of **X**, and **w** has the intercept as its first entry and the regression coefficients as the remaining d entries. For convenience we will often blur the distinction between these two formulations and state the regression equation as $\mathbf{y} = \mathbf{X}\mathbf{w} + \epsilon$ with **X** having d columns and **w** having d rows – from the context it will be clear whether we are representing the intercept by means of homogeneous coordinates, or have rather zero-centred the target and features to achieve a zero intercept.

In the univariate case we were able to obtain a closed-form solution for **w:** can we do the same in the multivariate case? First, we are likely to need the covariances between every feature and the target variable. Consider the expression $\mathbf{X}^\mathrm{T}\mathbf{y}$, which is an n-vector, the j-th entry of which is the product of the j-th row of $\mathbf{X}^\mathrm{T}$ – i.e., the j-th column of **X**, which is $(x_{1j}, \ldots, x_{nj})$ – with $(y_1, \ldots, y_n)$:

$$(\mathbf{X}^\mathrm{T}\mathbf{y})_j = \sum_{i=1}^{n} x_{ij} y_i = \sum_{i=1}^{n} (x_{ij} - \mu_j)(y_i - \overline{y}) + n\mu_j \overline{y} = n(\sigma_{jy} + \mu_j \overline{y})$$

Assuming for the moment that every feature is zero-centred, we have $\mu_j = 0$ and thus $\mathbf{X}^T\mathbf{y}$ is an n-vector holding all the required covariances (times n).

In the univariate case we needed to normalise the features to have unit variance. In the multivariate case we can achieve this by means of a d-by-d scaling matrix: a diagonal matrix with diagonal entries $1/n\sigma_{jj}$. If $\mathbf{S}$ is a diagonal matrix with diagonal entries $n\sigma_{jj}$, we can get the required scaling matrix by simply inverting $\mathbf{S}$. So our first stab at a solution for the *multivariate regression* problem is

$$\hat{\mathbf{w}} = \mathbf{S}^{-1}\mathbf{X}^T\mathbf{y} \tag{7.2}$$

As it turns out, the general case requires a more elaborate matrix instead of $\mathbf{S}$:

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{7.3}$$

Let us try to understand the term $(\mathbf{X}^T\mathbf{X})^{-1}$ a bit better. Assume that the features are uncorrelated (meaning the covariance between every pair of different features is 0) in addition to being zero-centred. In the notation of Background 7.2, the covariance matrix $\mathbf{\Sigma}$ is diagonal with entries σ_{jj}. Since $\mathbf{X}^T\mathbf{X} = n(\mathbf{\Sigma} + \mathbf{M})$, and since the entries of $\mathbf{M}$ are 0 because the columns of $\mathbf{X}$ are zero-centred, this matrix is also diagonal with entries $n\sigma_{jj}$ – in fact, it is the matrix $\mathbf{S}$ referred to above. In other words, assuming zero-centred and uncorrelated features, $(\mathbf{X}^T\mathbf{X})^{-1}$ reduces to our scaling matrix $\mathbf{S}^{-1}$. In the general case we cannot make any assumptions about the features, and $(\mathbf{X}^T\mathbf{X})^{-1}$ *acts as a transformation that decorrelates, centres and normalises the features*.

To make this a bit more concrete, the next example shows how this works out in the bivariate case.

Example 7.3 (Bivariate linear regression in matrix notation). First, we derive the basic expressions.

$$
\begin{aligned}
\mathbf{X}^{\mathrm{T}}\mathbf{X} &= \begin{pmatrix} x_{11} & \cdots & x_{n1} \\ x_{12} & \cdots & x_{n2} \end{pmatrix} \begin{pmatrix} x_{11} & x_{12} \\ \vdots & \vdots \\ x_{n1} & x_{n2} \end{pmatrix} = n \begin{pmatrix} \sigma_{11} + \overline{x_1}^2 & \sigma_{12} + \overline{x_1}\,\overline{x_2} \\ \sigma_{12} + \overline{x_1}\,\overline{x_2} & \sigma_{22} + \overline{x_2}^2 \end{pmatrix} \\
(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1} &= \frac{1}{nD} \begin{pmatrix} \sigma_{22} + \overline{x_2}^2 & -\sigma_{12} - \overline{x_1}\,\overline{x_2} \\ -\sigma_{12} - \overline{x_1}\,\overline{x_2} & \sigma_{11} + \overline{x_1}^2 \end{pmatrix} \\
D &= (\sigma_{11} + \overline{x_1}^2)(\sigma_{22} + \overline{x_2}^2) - (\sigma_{12} + \overline{x_1}\,\overline{x_2})^2 \\
\mathbf{X}^{\mathrm{T}}\mathbf{y} &= \begin{pmatrix} x_{11} & \cdots & x_{n1} \\ x_{12} & \cdots & x_{n2} \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = n \begin{pmatrix} \sigma_{1y} + \overline{x_1}\,\overline{y} \\ \sigma_{2y} + \overline{x_2}\,\overline{y} \end{pmatrix}
\end{aligned}
$$

We now consider two special cases. The first is that **X** is in homogeneous coordinates, i.e., we are really dealing with a univariate problem. In that case we have $x_{i1} = 1$ for $1 \le i \le n$; $\overline{x_1} = 1$ and $\sigma_{11} = \sigma_{12} = \sigma_{1y} = 0$. We then obtain (we write x instead of instead of and x_2 instead of x_2, σ_{xx} instead of σ_{22} and σ_{xy} instead of σ_{2y}):

$$
\begin{aligned}
(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1} &= \frac{1}{n\sigma_{xx}} \begin{pmatrix} \sigma_{xx} + \overline{x}^2 & -\overline{x} \\ -\overline{x} & 1 \end{pmatrix} \\
\mathbf{X}^{\mathrm{T}}\mathbf{y} &= n \begin{pmatrix} \overline{y} \\ \sigma_{xy} + \overline{x}\,\overline{y} \end{pmatrix} \\
\hat{\mathbf{w}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} &= \frac{1}{\sigma_{xx}} \begin{pmatrix} \sigma_{xx}\overline{y} - \sigma_{xy}\overline{x} \\ \sigma_{xy} \end{pmatrix}
\end{aligned}
$$

This is the same result as obtained in Example 7.1.

The second special case we consider is where we assume x_1, x_2 and y to be zero-centred, which means that the intercept is zero and **w** contains the two regression coefficients. In this case we obtain

$$
\begin{aligned}
(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1} &= \frac{1}{n(\sigma_{11}\sigma_{22} - \sigma_{12}^2)} \begin{pmatrix} \sigma_{22} & -\sigma_{12} \\ -\sigma_{12} & \sigma_{11} \end{pmatrix} \\
\mathbf{X}^{\mathrm{T}}\mathbf{y} &= n \begin{pmatrix} \sigma_{1y} \\ \sigma_{2y} \end{pmatrix} \\
\hat{\mathbf{w}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} &= \frac{1}{(\sigma_{11}\sigma_{22} - \sigma_{12}^2)} \begin{pmatrix} \sigma_{22}\sigma_{1y} - \sigma_{12}\sigma_{2y} \\ \sigma_{11}\sigma_{2y} - \sigma_{12}\sigma_{1y} \end{pmatrix}
\end{aligned}
$$

The last expression shows, e.g., that the regression coefficient for x_1 may be nonzero even if x_1 doesn't correlate with the target variable ($\sigma_{1y} = 0$), on account of the correlation between x_1 and x_2 ($\sigma_{12} \neq 0$).

Notice that if we do assume $\sigma_{12} = 0$ then the components of $\hat{\mathbf{w}}$ reduce to σ_{jy}/σ_{jj}, which brings us back to Equation 7.2. *Assuming uncorrelated features effectively decomposes a multivariate regression problem into d univariate problems*. We shall see several other examples of decomposing multivariate learning problems into univariate problems in this book – in fact, we have already seen an example in the form of the ☞ *naive Bayes* classifier in Chapter 1. So, you may wonder, why take feature correlation into account at all?

The answer is that ignoring feature correlation can be harmful in certain situations. Consider Figure 7.3: on the left, there is little correlation among the features, and as a result the samples provide a lot of information about the true function. On the right, the features are highly negatively correlated in such a way that the sampled values $y = x_1 + x_2 + \epsilon$ appear nearly constant, as any increase in one feature is accompanied by a nearly equal decrease in the other. As a result, decomposing the problem into two univariate regression problems leads to learning a nearly constant function. To be fair, taking the full covariance matrix into account doesn't do so well either in this example. However, although we will not explore the details here, one advantage of the full covariance approach is that it allows us to recognise that we can't place much confidence in our estimates of the regression parameters in this situation. The computational cost of computing the closed-form solution in Equation 7.3 lies in inverting the d-by-d matrix $\mathbf{X}^T\mathbf{X}$, which can be prohibitive in high-dimensional feature spaces.

Figure 7.3. (left) Regression functions learned by linear regression. The true function is $y = x_1 + x_2$ (red plane). The red points are noisy samples of this function; the black points showthem projected onto the (x_1, x_2)-plane. The green plane indicates the function learned by linear regression; the blue plane is the result of

decomposing the problem into two univariate regression problems (blue points). Both are good approximations of the true function. (**right**) The same function, but now x_1 and x_2 are highly (negatively) correlated. The samples now give much less information about the true function: indeed, from the univariate decomposition it appears that the function is constant.

Regularised regression

We have just seen a situation in which leastsquares regression can become *unstable*: i.e., highly dependent on the training data. Instability is a manifestation of a tendency to overfit. *Regularisation* is a general method to avoid such overfitting by applying additional constraints to the weight vector. A common approach is to make sure the weights are, on average, small in magnitude: this is referred to as *shrinkage*. To show how this can be achieved, we first write down the leastsquares regression problem as an optimisation problem:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} (\mathbf{y} - \mathbf{X}\mathbf{w})^{\mathrm{T}} (\mathbf{y} - \mathbf{X}\mathbf{w})$$

The right-hand side is just a way to write the sum of squared residuals as a dot product. The regularised version of this optimisation is then as follows:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} (\mathbf{y} - \mathbf{X}\mathbf{w})^{\mathrm{T}} (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda ||\mathbf{w}||^2 \tag{7.4}$$

where $||\mathbf{w}||^2 = \Sigma_i w^2_i$ is the squared norm of the vector $\mathbf{w}$, or, equivalently, the dot product $\mathbf{w}^{\mathrm{T}}\mathbf{w}$; λ is a scalar determining the amount of regularisation. This regularised problem still has a closed-form solution:

$$\hat{\mathbf{w}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^{\mathrm{T}}\mathbf{y} \tag{7.5}$$

where $\mathbf{I}$ denotes the identity matrix with 1s on the diagonal and 0s everywhere else. Comparing this with Equation 7.3 on p.202 we see that regularisation amounts to adding λ to the diagonal of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$, a well-known trick to improve the numerical stability of matrix inversion. This form of leastsquares regression is known as *ridge regression*.

An interesting alternative form of regularised regression is provided by the *lasso*, which stands for 'least absolute shrinkage and selection operator'. It replaces the ridge regularisation term $\Sigma_i w^2_i$ with the sum of absolute weights $\Sigma_i |w_i|$. (Using terminology that will be introduced in Definition 8.2 on p.235: lasso uses L_1 regularisation where ridge regression uses the L_2 norm.) The result is

that some weights are shrunk, but others are set to 0, and so the lasso regression favours *sparse solutions*. It should be added that lasso regression is quite sensitive to the regularisation parameter λ, which is usually set on hold-out data or in cross-validation. Also, there is no closed-form solution and so some numerical optimisation technique must be applied.

Using leastsquares regression for classification

So far we have used the leastsquares method to construct function approximators. Interestingly, we can also use linear regression to learn a binary classifier by encoding the two classes as real numbers. For instance, we can label the *Pos* positive examples with $y^{\oplus} = +1$ and the *Neg* negative examples with $y^{\ominus} = -1$. It then follows that $\mathbf{X}^{\mathrm{T}}\mathbf{y} = Pos\ \mu^{\oplus} - \mathrm{Neg}\ \mu^{\ominus}$, where $\mu^{\oplus}$ and $\mu^{\ominus}$ are d-vectors containing each feature's mean values for the positive and negative examples, respectively.

Example 7.4 (Univariate leastsquares classifier). In the univariate case we have $\Sigma_i\, x_i\, y_i = Pos\ \mu^{\oplus} - \mathrm{Neg}\ \mu^{\ominus}$; we also know (see Example 7.3) that $\Sigma_i\, x_i\, y_i = n(\sigma_{xy} + \bar{x}\,\bar{y})$, and so $\sigma_{xy} = Pos\ \mu^{\oplus} - neg\ \mu^{\ominus} - \bar{x}\,\bar{y}$. Since $\bar{x} = Pos\ \mu^{\oplus} + neg\ \mu^{\ominus}$ and $\bar{y} = pos - neg$, we can rewrite the covariance between x and y as $\sigma_{xy} = 2pos \cdot \mathrm{neg}\ (\mu^{\oplus} - \mu^{\ominus})$, and so the slope of the regression line is

$$\hat{b} = 2pos \cdot neg\, \frac{\mu^{\oplus} - \mu^{\ominus}}{\sigma_{xx}} \tag{7.6}$$

This equation shows that the slope of the regression line increases with the separation between the classes (measured as the distance between the class means in proportion to the feature's variance), but also decreases if the class distribution becomes skewed.

The regression equation $y = \bar{y} + \hat{b}(x - \bar{x})$ can then be used to obtain a decision boundary. We need to determine the point (x_0, y_0) such that y_0 is halfway between $y^{\oplus}$ and $y^{\ominus}$ (i.e., $y_0 = 0$ in our case). We then have

$$x_0 = \bar{x} + \frac{y_0 - \bar{y}}{\hat{b}} = \bar{x} - \frac{pos - neg}{2pos \cdot neg}\frac{\sigma_{xx}}{\mu^{\oplus} - \mu^{\ominus}}$$

That is, if there are equal numbers of positive and negative examples we simply threshold the feature at the feature mean $\bar{x}$; in case of unequal class distribution we shift this threshold to the left or right as appropriate (Figure 7.4).

Figure 7.4. Using univariate linear regression to obtain a decision boundary. The 10 negative examples are labelled with $y^{\ominus} = -1$ and the 20 positive examples are labelled $y^{\oplus} = +1$. $\mu^{\ominus}$ and $\mu^{\oplus}$ are indicated by red circles. The blue line is the linear regression line $y = \bar{y} + \hat{b}(x - \bar{x})$, and the crosshair indicates the decision boundary $x_0 = \bar{x} - \bar{y}/\hat{b}$. This results in three examples being misclassified – notice that this is the best that can be achieved with the given data.

In the general case, the *leastsquares classifier* learns the decision boundary $\mathbf{w}\cdot\mathbf{x} = t$ with

$$\mathbf{w} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}(Pos\,\boldsymbol{\mu}^{\oplus} - Neg\,\boldsymbol{\mu}^{\ominus}) \tag{7.7}$$

We would hence assign class $\hat{y} = \mathrm{sign}(\mathbf{w} \cdot \mathbf{x} - t)$ to instance $\mathbf{x}$, where

$$\mathrm{sign}(x) = \begin{cases} +1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

Various simplifying assumptions can be made, including zero-centred features, equal-variance features, uncorrelated features and equal class prevalences. In the simplest case, when all these assumptions are made, Equation 7.7 reduces to $\mathbf{w} = c\,(\mu^{\oplus} - \mu^{\ominus})$ where c is some scalar that can be incorporated in the decision threshold t. We recognise this as the ☞ *ibasic linear classifier* that was

introduced in the Prologue. Equation 7.7 thus tells us how to adapt the basic linear classifier, using the leastsquares method, in order to take feature correlation and unequal class distributions into account.

In summary, *a general way of constructing a linear classifier with decision boundary* $\mathbf{w} \cdot \mathbf{x} = t$ *is by constructing* $\mathbf{w}$ *as* $\mathbf{M}^{-1}(n^{\oplus}\,\mu^{\oplus} - n^{\ominus}\,\mu^{\ominus})$, with different possible choices of $\mathbf{M}$, $n^{\oplus}$ and $n^{\ominus}$. The full covariance approach with $\mathbf{M} = \mathbf{X}^{\mathrm{T}}\mathbf{X}$has time complexity $O(n^2 d)$ for construction of $\mathbf{M}$ and $O(d^3)$ for inverting it, [1] so this approach becomes unfeasible with large numbers offeatures.

7.2 The perceptron

Recall from Chapter 1 that labelled data is called ☞*linearly separable* if there exists a linear decision boundary separating the classes. The leastsquares classifier may find a perfectly separating decision boundary if one exists, but this is not guaranteed. To see this, suppose that the basic linear classifier achieves perfect separation for a given training set. Now, move all but one of the positive points away from the negative class. The decision boundary will also move away from the negative class, at some point crossing the one positive that remains fixed. By construction, the modified data is still linearly separable, since the original decision boundary separates it; however, the statistics of the modified data are such that the basic linear classifier will misclassify the one positive outlier.

A linear classifier that will achieve perfect separation on linearly separable data is the *perceptron*, originally proposed as a simple neural network. The perceptron iterates over the training set, updating the weight vector every time it encounters an incorrectly classified example. For example, let $\boldsymbol{x}_i$ be a misclassified positive example, then we have $y_i = +1$ and $\mathbf{w}\cdot\mathbf{x}_i < t$. We therefore want to find $\mathbf{w}'$ such that $\mathbf{w}'\cdot\mathbf{x}_i > \mathbf{w}\cdot\mathbf{x}_i$, which moves the decision boundary towards and hopefully past x_i. This can be achieved by calculating the new weight vector as $\mathbf{w}' = \mathbf{w} + \eta\mathbf{x}_i$, where $0 < \eta \leq 1$ is the *learning rate*. We then have $\mathbf{w}'\cdot\mathbf{x}_i = \mathbf{w}\cdot\mathbf{x}_i + \eta\mathbf{x}_i\cdot\mathbf{x}_i > \mathbf{w}\cdot\mathbf{x}_i$ as required. Similarly, if $\mathbf{x}_j$ is a misclassified negative example, then we have $y_j = -1$ and $\mathbf{w}\cdot\mathbf{x}_j > t$. In this case we calculate the new weight vector as $\mathbf{w}' = \mathbf{w} - \eta\mathbf{x}_j$, and thus $\mathbf{w}'\cdot\mathbf{x}_j = \mathbf{w}\cdot\mathbf{x}_j - \eta\mathbf{x}_j\cdot\mathbf{x}_j < \mathbf{w}'\cdot\mathbf{x}_j$. The two cases can be combined in a single update rule:

$$\mathbf{w}' = \mathbf{w} + \eta y_i \mathbf{x}_i \tag{7.8}$$

The perceptron training algorithm is given in Algorithm 7.1. It iterates through the training examples until all examples are correctly classified. The algorithm can easily be turned into an *online* algorithm that processes a stream of examples, updating the weight vector only if the last received example is misclassified. The perceptron is guaranteed to converge to a solution if the training data is linearly separable, but it won't converge otherwise. Figure 7.5 gives a graphical illustration of the perceptron training algorithm. In this particular example I initialised the weight vector to the basic linear classifier, which means the learning rate does have an effect on how quickly we move away from the initial decision boundary. However, if the weight vector is initialised to the zero vector, it is easy to see that the learning rate is just a constant factor that does not affect convergence. We will set it to 1 in the remainder of this section.

The key point of the perceptron algorithm is that, every time an example $\mathbf{x}_i$ is misclassified, we add $y_i x_i$ to the weight vector. After training has completed, each example has been misclassified zero or more times – denote this number α_i for example $\mathbf{x}_i$.

Algorithm 7.1: Perceptron(D, η) – train a perceptron for linear classification.

```
Input  : labelled training data D in homogeneous coordinates;
         learning rate η.
Output : weight vector w defining classifier ŷ = sign(w·x).
w ← 0 ;                    // Other initialisations of the weight vector are possible
converged ← false;
while converged = false do
    converged ← true;
    for i = 1 to |D| do
        if y_i w·x_i ≤ 0          // i.e., ŷ_i ≠ y_i
        then
            w ← w + η y_i x_i;
            converged ← false;    // We changed w so haven't converged yet
        end
    end
end
```

Figure 7.5. (left) A perceptron trained with a small learning rate (η = 0.2). The circled examples are the ones that trigger the weight update. **(middle)** Increasing the learning rate to η = 0.5 leads in this case to a rapid convergence. **(right)** Increasing the learning rate further to η = 1 may lead to too aggressive weight updating, which harms convergence. The starting point in all three cases was the basic linear classifier.

Using this notation the weight vector can be expressed as

$$\mathbf{w} = \sum_{l=1}^{n} \alpha_l y_l \mathbf{x}_l \tag{7.9}$$

In other words, the weight vector is a linear combination of the training instances. The perceptron shares this property with, e.g., the basic linear classifier:

$$\mathbf{w}_{blc} = \boldsymbol{\mu}^{\oplus} - \boldsymbol{\mu}^{\ominus} = \frac{1}{Pos}\sum_{\mathbf{x}^{\oplus}\in Tr^{\oplus}} \mathbf{x}^{\oplus} - \frac{1}{Neg}\sum_{\mathbf{x}^{\ominus}\in Tr^{\ominus}} \mathbf{x}^{\ominus} = \sum_{\mathbf{x}^{\oplus}\in Tr^{\oplus}} \alpha^{\oplus} c(\mathbf{x}^{\oplus})\mathbf{x}^{\oplus} + \sum_{\mathbf{x}^{\ominus}\in Tr^{\ominus}} \alpha^{\ominus} c(\mathbf{x}^{\ominus})\mathbf{x}^{\ominus} \tag{7.10}$$

Algorithm 7.2: DualPerceptron(D) – perceptron training in dual form.

Input : labelled training data D in homogeneous coordinates.
Output : coefficients α_i defining weight vector $\mathbf{w} = \sum_{i=1}^{|D|} \alpha_i y_i \mathbf{x}_i$.

```
α_i ← 0 for 1 ≤ i ≤ |D|;
converged ← false;
while converged = false do
    converged ← true;
    for i = 1 to |D| do
        if y_i Σ_{j=1}^{|D|} α_j y_j x_i · x_j ≤ 0 then
            α_i ← α_i + 1;
            converged ← false;
        end
    end
end
```

Figure 7.6. Three differently trained linear classifiers on a data set of 100 positives (top-right) and 50 negatives (bottom-left): the basic linear classifier in red, the leastsquares classifier in orange and the perceptron in green. Notice that the perceptron perfectly separates the training data, but its heuristic approach may lead to overfitting in certain situations.

where $c(\mathbf{x})$ is the true class of example $\mathbf{x}$ (i.e., +1 or –1), $\alpha^{\oplus} = 1 / Pos$ and $\alpha^{\ominus} = 1/Neg$. *In the dual, instance-based view of linear classification we are learning instance weights α_i rather than feature weights w_j.* In this dual perspective, an instance $\mathbf{x}$ is classified as $\hat{y} = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x}\right)$. This means that, during training, the only information needed about the training data is all pairwise dot products: the n-by-n matrix $\mathbf{G} = \mathbf{X}\mathbf{X}^{\mathrm{T}}$ containing these dot products is called the *Gram matrix*. Algorithm 7.2 gives the dual formof the perceptron training algorithm. We will encounter this instance-based perspective again when we discuss support vector machines in the next section.

Figure 7.6 demonstrates the difference between the basic linear classifier, the leastsquares classifier and the perceptron on some random data. For this particular data set, neither the basic linear classifier nor the leastsquares classifier achieves perfect separation, but the perceptron does. One difference with other linear methods is that we cannot derive a closed-form solution for the weight vector learned by the perceptron, so it is a more heuristic approach.

The perceptron can easily be turned into a linear function approximator (Algorithm 7.3). To this end the update rule is changed to $\mathbf{w}' = \mathbf{w} + (y_i - \hat{y}_i)^2 \mathbf{x}_i$, which uses squared residuals. This is unlikely to converge to the exact function, so the algorithm simply runs for a fixed number of training epochs (an epoch is one complete run through the training data). Alternatively, one could run the algorithm until a bound on the sum of squared residuals is reached.

7.3 Support vector machines

Linearly separable data admits infinitely many decision boundaries that separate the classes, but intuitively some of these are better than others. For example, the left and middle decision boundaries in Figure 7.5 seem to be unnecessarily close to some of the positives; while the one on the right leaves a bit more space on either side, it doesn't seem particularly good either. To make this a bit more precise, recall that in Section 2.2 we defined the ☞ *margin* of an example assigned by a scoring classifier as $c(x)\hat{s}(x)$, where $c(x)$ is +1 for positive examples and –1 for negative examples and $\hat{s}(x)$ is the score of example x. If we

take $\hat{s}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} - t$, then a true positive $\mathbf{x}_i$ has margin $\mathbf{w} \cdot \mathbf{x}_i - t > 0$ and a true negative $\mathbf{x}_j$ has margin $-(\mathbf{w} \cdot \mathbf{x}_j - t) > 0$. For a given training set and decision boundary, let $m^{\oplus}$ be the smallest margin of any positive, and $m^{\ominus}$ the smallest margin of any negative, then we want the sum of these to be as large as possible. This sum is independent of the decision threshold t, as long as we keep the nearest positives and negatives at the right sides of the decision boundary, and so we re-adjust t such that $m^{\oplus}$ and $m^{\ominus}$ become equal. Figure 7.7 depicts this graphically in a two-dimensional instance space. The training examples nearest to the decision boundary are called *support vectors*: as we shall see, the decision boundary of a support vector machine (SVM) is defined as a linear combination of the support vectors.

The *margin* is thus defined as $m/||\mathbf{w}||$, where m is the distance between the decision boundary and the nearest training instances (at least one of each class) as measured along $\mathbf{w}$. Since we are free to rescale t, $||\mathbf{w}||$ and m, it is customary to choose $m = 1$. Maximising the margin then corresponds to minimising $||\mathbf{w}||$ or, more conveniently, $\frac{1}{2}||\mathbf{w}||^2$, provided of course that none of the training points fall inside the margin. This leads to a quadratic, constrained optimisation problem:

Algorithm 7.3: PerceptronRegression(D, T) – train a perceptron for regression.

```
Input  : labelled training data D in homogeneous coordinates;
         maximum number of training epochs T.
Output : weight vector w defining function approximator ŷ = w·x.
w ← 0; t ← 0;
while t < T do
    for i = 1 to |D| do
        w ← w + (y_i − ŷ_i)^2 x_i;
    end
    t ← t + 1;
end
```

Figure 7.7. The geometry of a support vector classifier. The circled data points are the support vectors, which are the training examples nearest to the decision boundary. The support vector machine finds the decision boundary that maximises the margin $m/||\mathbf{w}||$.

$$\mathbf{w}^*, t^* = \arg\min_{\mathbf{w},t} \frac{1}{2}||\mathbf{w}||^2 \qquad \text{subject to } y_i(\mathbf{w}\cdot\mathbf{x}_i - t) \geq 1, 1 \leq i \leq n$$

We will approach this using the method of Lagrange multipliers (see Background 7.3). Adding the constraints with multipliers α_i for each training example gives the Lagrange function

$$\begin{aligned}
\Lambda(\mathbf{w}, t, \alpha_1, \ldots, \alpha_n) &= \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \alpha_i (y_i(\mathbf{w}\cdot\mathbf{x}_i - t) - 1) \\
&= \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \alpha_i y_i (\mathbf{w}\cdot\mathbf{x}_i) + \sum_{i=1}^{n} \alpha_i y_i t + \sum_{i=1}^{n} \alpha_i \\
&= \frac{1}{2}\mathbf{w}\cdot\mathbf{w} - \mathbf{w}\cdot\left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i\right) + t\left(\sum_{i=1}^{n} \alpha_i y_i\right) + \sum_{i=1}^{n} \alpha_i
\end{aligned}$$

While this looks like a formidable formula, some further analysis will allow us to derive the simpler dual form of the Lagrange function.

By taking the partial derivative of the Lagrange function with respect to t and setting it to 0 we find that for the optimal threshold t we have $\sum_{i=1}^{n} \alpha_i y_i = 0$. Similarly, by taking the partial derivative of the Lagrange function with respect to w we see that the Lagrange multipliers define the weight vector as a linear combination of the training examples:

Optimisation is a broad term denoting the problem of finding the best item or value among a set of alternatives. We have already seen a very simple, unconstrained form of optimisation in Example 7.1 on p.197, where we found the values of a and b minimising the sum of squared residuals $f(a,b) = \sum_{i=1}^{n}(w_i - (a + bh_i))^2$; this can be denoted as

$$a^*, b^* = \underset{a,b}{\operatorname{argmin}}\, f(a,b)$$

f is called the *objective function;* it can be linear, quadratic (as in this case), or more complex. We found the minimum of f by setting the partial derivatives of f with respect to a and b to 0, and solving for a and b; the vector of these partial derivatives is called the *gradient* and denoted ∇f, so a succinct way of defining the unconstrained optimisation problem is: find a and b such that $\nabla f(a, b) = \mathbf{0}$. In this particular case the objective function is *convex*, which essentially means that there is a unique global minimum. This is, however, not always the case.

A *constrained optimisation* problem is one where the alternatives are subject to constraints, for instance

$$a^*, b^* = \underset{a,b}{\operatorname{argmin}}\, f(a,b) \qquad \text{subject to } g(a,b) = c$$

If the relationship expressed by the constraint is linear, say $a - b = 0$, we can of course eliminate one of the variables and solve the simpler, unconstrained problem. However, this may not be possible if the constraints are non-linear. *Lagrange multipliers* are a powerful way of dealing with the general case. We form the Lagrange function defined by

$$\Lambda(a,b,\lambda) = f(a,b) - \lambda(g(a,b) - c)$$

where λ is the Lagrange multiplier, and solve the unconstrained problem $\nabla \wedge (a, b, \lambda) = \mathbf{0}$. Since $\nabla_{a,b} \wedge (a, b, \lambda) = \nabla f(a, b) - \lambda \nabla g(a, b)$ and $\nabla_{\lambda} \wedge (a, b, \lambda) = g(a, b) - c$, this is a succinct way of requiring (*i*) that the gradients of f and g point in the same direction, and (*ii*) that the constraint is satisfied. We can include multiple equality constraints and also inequality constraints, each with their own Lagrange multiplier.

From the Lagrange function it is possible to derive a *dual* optimisation

problem where we find the optimal values of the Lagrange multipliers. In general, the solution to the dual problem is only a lower bound on the solution to the *primal* problem, but under a set of conditions known as the *Karush–Kuhn–Tucker conditions* (*KKT*) the two solutions become equal. The quadratic optimisation problem posed by support vector machines is usually solved in its dual form.

Background 7.3. Basic concepts and terminology in mathematical optimisation.

$$\frac{\partial}{\partial \mathbf{w}}\Lambda(\mathbf{w}, t, \alpha_1, \ldots, \alpha_n) = \frac{\partial}{\partial \mathbf{w}}\frac{1}{2}\mathbf{w}\cdot\mathbf{w} - \frac{\partial}{\partial \mathbf{w}}\mathbf{w}\cdot\left(\sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i\right) = \mathbf{w} - \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i$$

Since this partial derivative is 0 for an optimal weight vector we conclude $\mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i x_i$ – the same expression as we derived for the perceptron in Equation 7.9 on p.209. For the perceptron, the instance weights α_i are non-negative integers denoting the number of times an example has been misclassified in training. For a support vector machine, the α_i are non-negative reals. What they have in common is that, if $\alpha_i = 0$ for a particular example $\mathbf{x}_i$, that example could be removed from the training set without affecting the learned decision boundary. In the case of support vector machines this means that $\alpha_i > 0$ only for the support vectors: the training examples nearest to the decision boundary.

Now, by plugging the expressions $\sum_{i=1}^{n}\alpha_i y_i = 0$ and $\mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i x_i$ back into the Lagrangian we are able to eliminate $\mathbf{w}$ and t, and hence obtain the dual optimisation problem, which is entirely formulated in terms of the Lagrange multipliers:

$$\begin{aligned}\Lambda(\alpha_1, \ldots, \alpha_n) &= -\frac{1}{2}\left(\sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i\right)\cdot\left(\sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i\right) + \sum_{i=1}^{n}\alpha_i \\ &= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\cdot\mathbf{x}_j + \sum_{i=1}^{n}\alpha_i\end{aligned}$$

The dual problem is to maximise this function under positivity constraints and one equality constraint:

$$\alpha_1^*, \ldots, \alpha_n^* = \underset{\alpha_1,\ldots,\alpha_n}{\arg\max} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j + \sum_{i=1}^{n}\alpha_i$$

$$\text{subject to } \alpha_i \geq 0, 1 \leq i \leq n \text{ and } \sum_{i=1}^{n}\alpha_i y_i = 0$$

The dual form of the optimisation problem for support vector machines illustrates two important points. First, it shows that searching for the maximum-margindecision boundary is equivalent to searching for the support vectors: they are the training examples with nonzero Lagrange multipliers, and through $\mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i x_i$ they completely determine the decision boundary. Secondly, it shows that the optimisation problem is entirely defined by pairwise dot products between training instances: the entries of the Gram matrix. As we shall see in Section 7.5, this paves the way for a powerful adaptation of support vector machines that allows them to learn non-linear decision boundaries.

The following example makes these issues a bit more concrete by showing detailed calculations on some toy data.

Figure 7.8. **(left)** A maximum-margin classifier built from three examples, with $\mathbf{w} = (0, -1/2)$and margin 2. The circled examples are the support vectors: they receive nonzero Lagrange multipliers and define the decision boundary. **(right)** By adding a second positive the decision boundary is rotated to $\mathbf{w} = (3/5, -4/5)$ and the margin decreases to 1.

Example 7.5 (Two maximum-margin classifiers and their support vectors). Let the data points and labels be as follows (see Figure 7.8 (left)):

$$\mathbf{X} = \begin{pmatrix} 1 & 2 \\ -1 & 2 \\ -1 & -2 \end{pmatrix} \qquad \mathbf{y} = \begin{pmatrix} -1 \\ -1 \\ +1 \end{pmatrix} \qquad \mathbf{X}' = \begin{pmatrix} -1 & -2 \\ 1 & -2 \\ -1 & -2 \end{pmatrix}$$

The matrix $\mathbf{X}'$ on the right incorporates the class labels; i.e., the rows are y_i $\mathbf{x}_i$. The Gram matrix is (without and with class labels):

$$\mathbf{X}\mathbf{X}^{\mathrm{T}} = \begin{pmatrix} 5 & 3 & -5 \\ 3 & 5 & -3 \\ -5 & -3 & 5 \end{pmatrix} \qquad \mathbf{X}'\mathbf{X}'^{\mathrm{T}} = \begin{pmatrix} 5 & 3 & 5 \\ 3 & 5 & 3 \\ 5 & 3 & 5 \end{pmatrix}$$

The dual optimisation problem is thus

$$\begin{aligned} &\underset{\alpha_1,\alpha_2,\alpha_3}{\operatorname{argmax}} -\frac{1}{2}\left(5\alpha_1^2 + 3\alpha_1\alpha_2 + 5\alpha_1\alpha_3 + 3\alpha_2\alpha_1 + 5\alpha_2^2 + 3\alpha_2\alpha_3 + 5\alpha_3\alpha_1 \right. \\ &\qquad\qquad \left. + 3\alpha_3\alpha_2 + 5\alpha_3^2\right) + \alpha_1 + \alpha_2 + \alpha_3 \\ &= \underset{\alpha_1,\alpha_2,\alpha_3}{\operatorname{argmax}} -\frac{1}{2}\left(5\alpha_1^2 + 6\alpha_1\alpha_2 + 10\alpha_1\alpha_3 + 5\alpha_2^2 + 6\alpha_2\alpha_3 + 5\alpha_3^2\right) + \alpha_1 + \alpha_2 + \alpha_3 \end{aligned}$$

subject to $\alpha_1 \geq 0$, $\alpha_2 \geq 0$, $\alpha_3 \geq 0$ and $-\alpha_1 - \alpha_2 + \alpha_3 = 0$. While in practice such problems are solved by dedicated quadratic optimisation solvers, here we will show how to solve this toy problem by hand.

Using the equality constraint we can eliminate one of the variables, say α_3, and simplify the objective function to

$$\begin{aligned} &\underset{\alpha_1,\alpha_2,\alpha_3}{\operatorname{argmax}} -\frac{1}{2}\left(5\alpha_1^2 + 6\alpha_1\alpha_2 + 10\alpha_1(\alpha_1+\alpha_2) + 5\alpha_2^2 + 6\alpha_2(\alpha_1+\alpha_2) + 5(\alpha_1+\alpha_2)^2\right) \\ &\qquad\qquad + 2\alpha_1 + 2\alpha_2 \\ &= \underset{\alpha_1,\alpha_2,\alpha_3}{\operatorname{argmax}} -\frac{1}{2}\left(20\alpha_1^2 + 32\alpha_1\alpha_2 + 16\alpha_2^2\right) + 2\alpha_1 + 2\alpha_2 \end{aligned}$$

Setting partial derivatives to 0 we obtain $-20\alpha_1 - 16\alpha_2 + 2 = 0$ and $-16\alpha_1 - 16\alpha_2 + 2 = 0$ (notice that, because the objective function is quadratic, these equations are guaranteed to be linear). We therefore obtain the solution $\alpha_1 = 0$ and $\alpha_2 = \alpha_3 = 1/8$. We then have $\mathbf{w} = 1/8(\mathbf{x}_3 - \mathbf{x}_2) = \begin{pmatrix} 0 \\ -1/2 \end{pmatrix}$, resulting in a margin of $\mathbf{1} / \|\mathbf{w}\| = 2$. Finally, t can be obtained from any support vector, say $\mathbf{x}_2$, since $y_2(\mathbf{w} \cdot \mathbf{x}_2 - t) = 1$; this gives $-1\cdot(-1 - t) = 1$, hence $t = 0$. The

resulting maximum-margin classifier is depicted in Figure 7.8 (left). Notice that the first example $\mathbf{x}_1$ is not a support vector, even though it is on the margin: this is because removing it will not affect the decision boundary.

We now add an additional positive at (3, 1). This gives the following data matrices:

$$\mathbf{X}' = \begin{pmatrix} -1 & -2 \\ 1 & -2 \\ -1 & -2 \\ 3 & 1 \end{pmatrix} \qquad \mathbf{X}'\mathbf{X}'^{\mathrm{T}} = \begin{pmatrix} 5 & 3 & 5 & -5 \\ 3 & 5 & 3 & 1 \\ 5 & 3 & 5 & -5 \\ -5 & 1 & -5 & 10 \end{pmatrix}$$

It can be verified by similar calculations to those above that the margin decreases to 1 and the decision boundary rotates to $\mathbf{w} = \begin{pmatrix} 3/5 \\ -4/5 \end{pmatrix}$ (Figure 7.8 (right)). The Lagrange multipliers now are $\alpha_1 = 1/2$, $\alpha_2 = 0$, $\alpha_3 = 1/10$ and $\alpha_4 = 2/5$. Thus, only $\mathbf{x}_3$ is a support vector in both the original and the extended data set.

Soft margin SVM

If the data is not linearly separable, then the constraints $\mathbf{w} \cdot \mathbf{x}_i - t \geq 1$ posed by the examples are not jointly satisfiable. However, there is a very elegant way of adapting the optimisation problem such that it admits a solution even in this case. The idea is to introduce *slack variables* ξ_i, one for each example, which allow some of them to be inside the margin or even at the wrong side of the decision boundary – we will call these *margin errors*. Thus, we change the constraints to $\mathbf{w} \cdot \mathbf{x}_i - t \geq 1 - \xi_i$ and add the sum of all slack variables to the objective function to be minimised, resulting in the following *soft margin* optimisation problem:

$$\mathbf{w}^*, t^*, \xi_i^* = \underset{\mathbf{w}, t, \xi_i}{\arg\min} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \xi_i$$

$$\text{subject to } y_i(\mathbf{w} \cdot \mathbf{x}_i - t) \geq 1 - \xi_i \text{ and } \xi_i \geq 0, 1 \leq i \leq n \qquad (7.11)$$

C is a user-defined parameter trading off margin maximisation against slack

variable minimisation: a high value of C means that margin errors incur a high penalty, while a low value permits more margin errors (possibly including misclassifications) in order to achieve a large margin. If we allow more margin errors we need fewer support vectors, hence C controls to some extent the 'complexity' of the SVM and hence is often referred to as the *complexity parameter*. It can be seen as a form of regularisation similar to that discussed in the context of leastsquares regression.

The Lagrange function is then as follows:

$$\begin{aligned}\Lambda(\mathbf{w}, t, \xi_i, \alpha_i, \beta_i) &= \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i(y_i(\mathbf{w}\cdot\mathbf{x}_i - t) - (1-\xi_i)) - \sum_{i=1}^{n}\beta_i\xi_i \\ &= \frac{1}{2}\mathbf{w}\cdot\mathbf{w} - \mathbf{w}\cdot\left(\sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i\right) + t\left(\sum_{i=1}^{n}\alpha_i y_i\right) + \sum_{i=1}^{n}\alpha_i + \sum_{i=1}^{n}(C-\alpha_i-\beta_i)\xi_i \\ &= \Lambda(\mathbf{w}, t, \alpha_i) + \sum_{i=1}^{n}(C-\alpha_i-\beta_i)\xi_i\end{aligned}$$

For an optimal solution every partial derivative with respect to ξ_i should be 0, from which it follows that $C - \alpha_i - \beta_i = 0$ for all i , and hence the added term vanishes from the dual problem. Furthermore, since both α_i and β_i are positive, this means that α_i cannot be larger than C, which manifests itself as an additional upper bound on α_i in the dual problem:

$$\alpha_1^*, \ldots, \alpha_n^* = \underset{\alpha_1,\ldots,\alpha_n}{\arg\max} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\cdot\mathbf{x}_j + \sum_{i=1}^{n}\alpha_i$$

$$\text{subject to } 0 \le \alpha_i \le C \text{ and } \sum_{i=1}^{n}\alpha_i y_i = 0 \qquad (7.12)$$

This is a remarkable and beautiful result. It follows from the particular way that slack variables were added to the optimisation problem in Equation 7.11. By restricting the slack variables to be positive and adding them to the objective function to be minimised, they function as penalty terms, measuring deviations on the wrong side of the margin only. Furthermore, the fact that the β_i multipliers do not appear in the dual objective follows from the fact that the penalty term in the primal objective is linear in ξ_i. In effect, these slack variables implement what was called *hinge loss* in Figure 2.6 on p.63: a margin $z > 1$ incurs no penalty, and a margin $z = 1 - \xi \le 1$ incurs a penalty $\xi = 1 - z$.

Figure 7.9. (**left**) The soft margin classifier learned with $C = 5/16$, at which point $\mathbf{x}_2$ is about to become a support vector. (*right*) The soft margin classifier learned with $C = 1/10$: all examples contribute equally to the weight vector. The asterisks denote the class means, and the decision boundary is parallel to the one learned by the basic linear classifier.

What is the significance of the upper bound C on the α_i multipliers? Since $C - \alpha_i - \beta_i = 0$ for all i, $\alpha_i = C$ implies $\beta_i = 0$. The β_i multipliers come from the $\xi_i \geq 0$ constraint, and a multiplier of 0 means that the lower bound is not reached, i.e., $\xi_i > 0$ (analogous to the fact that $\alpha_j = 0$ means that $\mathbf{x}_j$ is not a support vector and hence $\mathbf{w}\cdot\mathbf{x}_j - t > 1$). In other words, a solution to the soft margin optimisation problem in dual form divides the training examples into three cases:

$\alpha_i = 0$	these are outside or on the margin;
$0 < \alpha_i < C$	these are the support vectors on the margin;
$\alpha_i = C$	these are on or inside the margin.

Notice that we still have $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i x_i$, and so both second and third case examples participate in spanning the decision boundary.

Example 7.6 (Soft margins). We continue Example 7.5, where we saw that adding the positive example $\mathbf{x}_4 = (3, 1)$ to the first three examples significantly reduced the margin from 2 to 1. We will now show that soft margin classifiers with larger margins are learned with sufficiently large complexity parameter C.

Recall that the Lagrange multipliers for the classifier in Figure 7.8 (right) are $\alpha_1 = 1/2$, $\alpha_2 = 0$, $\alpha_3 = 1/10$ and $\alpha_4 = 2/5$. So α_1 is the largest multiplier, and as long as $C > \alpha_1 = 1/2$ no margin errors are tolerated. For $C = 1/2$ we have $\alpha_1 = C$, and hence for $C < 1/2$ we have that $\mathbf{x}_1$ becomes a margin error and the optimal classifier is a soft margin classifier. Effectively, with decreasing C the decision boundary and the upper margin move upward, while the lower margin stays the same.

The upper margin reaches $\mathbf{x_2}$ for $C = 5/16$ (Figure 7.9 (left)), at which point we have $\mathbf{w} = \begin{pmatrix} 3/8 \\ -1/2 \end{pmatrix}$, $t = 3/8$ and the margin has increased to 1.6. Furthermore, we have $\xi_1 = 6/8$, $\alpha_1 = C = 5/16$, $\alpha_2 = 0$, $\alpha_3 = 1/16$ and $\alpha_4 = 1/4$.

If we now decrease C further, the decision boundary starts to rotate clockwise, so that $\mathbf{x}_4$ becomes a margin error as well, and only $\mathbf{x}_2$ and $\mathbf{x}_3$ are support vectors. The boundary rotates until $C = 1/10$, at which point we have $\mathbf{w} = \begin{pmatrix} 1/5 \\ -1/2 \end{pmatrix}$, $t = 1/5$ and the margin has increased to 1.86. Furthermore, we $\xi_1 = 4/10$ and $\xi_4 = 7/10$, and all multipliers have become equal to C (Figure 7.9 (right)).

Finally, when C decreases further the decision boundary stays where it is, but the norm of the weight vector gradually decreases and all points become margin errors.

Example 7.6 illustrates an important point: for low enough C, all examples receive the same multiplier C, and hence we have $\mathbf{w} = C\sum_{i=1}^{n} y_i \mathbf{x}_i = C\,(Pos \cdot \mu^{\oplus} - Neg \cdot \mu^{\ominus})$, where $\mu^{\oplus}$ and $\mu^{\ominus}$ are the means of the positive and negative examples, respectively. In other words, *a minimal-complexity soft margin classifier summarises the classes by their class means in a way very similar to the basic linear classifier*. For intermediate values of C the decision boundary is spanned by the support vectors and the per-class means of the margin errors.

In summary, support vector machines are linear classifiers that construct the unique decision boundary that maximises the distance to the nearest training examples (the support vectors). The complexity parameter C can be used to adjust the number and severity of allowed margin violations. Training an SVM

involves solving a large quadratic optimisation problem and is usually best left to a dedicated numerical solver.

7.4 Obtaining probabilities from linear classifiers

As we have seen, a linear classifier produces scores $\hat{s}(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x}_i - t$ that are thresholded at 0 in order to classify examples. Owing to the geometric nature of linear classifiers, such scores can be used to obtain the (signed) distance of $\mathbf{x}_i$ from the decision boundary. To see this, notice that the length of the projection of $\mathbf{x}_i$ onto $\mathbf{w}$ is $||\mathbf{x}_i|| \cos \theta$, where θ is the angle between $\mathbf{x}_i$ and $\mathbf{w}$. Since $\mathbf{w} \cdot \mathbf{x}_i = ||\mathbf{w}|| \, ||\mathbf{x}_i|| \cos \theta$, we can write this length as $(\mathbf{w} \cdot \mathbf{x}_i)/||\mathbf{w}||$. This gives the following signed distance:

Figure 7.10. We can think of a linear classifier as a projection onto the direction given by $\mathbf{w}$, here assumed to be a unit vector. $\mathbf{w} \cdot \mathbf{x} - t$ gives the signed distance from the decision boundary on the projection line. Also indicated are the class means $\mu^{\oplus}$ and $\mu^{\ominus}$, and the corresponding mean distances $d^{\oplus}$ and $d^{\ominus}$.

$$d(\mathbf{x}_i) = \frac{\hat{s}(\mathbf{x}_i)}{||\mathbf{w}||} = \frac{\mathbf{w} \cdot \mathbf{x}_i - t}{||\mathbf{w}||} = \mathbf{w}' \cdot \mathbf{x}_i - t'$$

with $\mathbf{w}' = \mathbf{w}/||\mathbf{w}||$ rescaled to unit length and $t' = t/||\mathbf{w}||$ the corresponding rescaled

intercept. The sign of this quantity tells us which side of the decision boundary we are on: positive distances for points on the 'positive' side of the decision boundary (the direction in which **w** points) and negative distances on the other side (Figure 7.10).

This geometric interpretation of the scores produced by linear classifiers offers an interesting possibility for turning them into probabilities, a process that was called ☞ *calibration* in Section 2.3. Let $\overline{d}^{\oplus}$ denote the mean distance of the positive examples to the decision boundary: i.e., $\overline{d}^{\oplus} = \mathbf{w} \cdot \mu^{\oplus} - t$, where $\mu^{\oplus}$ is the mean of the positive examples and **w** is unit length (although the latter assumption is not strictly necessary, as it will turn out that the weight vector will be rescaled). It would not be unreasonable to expect that the distance of positive examples to the decision boundary is normally distributed around this mean:[2] that is, when plotting a histogram of these distances, we would expect the familiar bell curve to appear. Under this assumption, the probability density function of d is $P(d|\oplus) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(d-\overline{d}^{\oplus})^2}{2\sigma^2}\right)$ (see Background 9.1 on p.267 if you need to remind yourself about the normal distribution). Similarly, the distances of negative examples to the decision boundary can be expected to be normally distributed around $\overline{d}^{\ominus} = \mathbf{w} \cdot \mu^{\ominus} - t$, with $\overline{d}^{\ominus} < 0 < \overline{d}^{\oplus}$. We will assume that both normal distributions have the same variance σ^2.

Suppose we now observe a point **x** with distance $d(\mathbf{x})$. We classify this point as positive if $d(\mathbf{x}) > 0$ and as negative if $d(\mathbf{x}) < 0$, but we want to attach a probability $\hat{P}(\mathbf{x}) = P(\oplus|d(\mathbf{x}))$ to these predictions. Using Bayes' rule we obtain

$$P(\oplus|d(\mathbf{x})) = \frac{P(d(\mathbf{x})|\oplus)P(\oplus)}{P(d(\mathbf{x})|\oplus)P(\oplus) + P(d(\mathbf{x})|\ominus)P(\ominus)} = \frac{LR}{LR + 1/clr}$$

where LR is the likelihood ratio obtained from the normal score distributions, and clr is the class ratio. We will assume for simplicity that $clr = 1$ in the derivation below. Furthermore, assume for now that $\sigma^2 = 1$ and $\overline{d}^{\oplus} = -\overline{d}^{\ominus} = 1/2$ (we will relax this in a moment). We then have

$$\begin{aligned} LR &= \frac{P(d(\mathbf{x})|\oplus)}{P(d(\mathbf{x})|\ominus)} = \frac{\exp\left(-(d(\mathbf{x}) - 1/2)^2/2\right)}{\exp\left(-(d(\mathbf{x}) + 1/2)^2/2\right)} \\ &= \exp\left(-(d(\mathbf{x}) - 1/2)^2/2 + (d(\mathbf{x}) + 1/2)^2/2\right) = \exp(d(\mathbf{x})) \end{aligned}$$

and so

$$P(\oplus|d(\mathbf{x})) = \frac{\exp(d(\mathbf{x}))}{\exp(d(\mathbf{x}))+1} = \frac{\exp(\mathbf{w}\cdot\mathbf{x}-t)}{\exp(\mathbf{w}\cdot\mathbf{x}-t)+1}$$

So, in order to obtain probability estimates from a linear classifier outputting distance scores d, we convert d into a probability by means of the mapping $d \mapsto \frac{\exp(d)}{\exp(d)+1}$ (or, equivalently, $d \mapsto \frac{1}{1+\exp(-d)}$). This S-shaped or *sigmoid* function is called the *logistic function*; it finds applications in a wide range of areas (Figure 7.11).

Suppose now that $\overline{d}^{\oplus} = -\overline{d}^{\ominus}$ as before, but we do not assume anything about the magnitude of these mean distances or of σ^2. In this case we have

$$\begin{aligned} LR &= \exp\left(\frac{-(d(\mathbf{x})-\overline{d}^{\oplus})^2 + (d(\mathbf{x})-\overline{d}^{\ominus})^2}{2\sigma^2}\right) \\ &= \exp\left(\frac{2\overline{d}^{\oplus} d(\mathbf{x}) - \left(\overline{d}^{\oplus}\right)^2 - 2\overline{d}^{\ominus} d(\mathbf{x}) + \left(\overline{d}^{\ominus}\right)^2}{2\sigma^2}\right) = \exp\left(\gamma d(\mathbf{x})\right) \end{aligned}$$

with $a = (\overline{d}^{\oplus} - \overline{d}^{\ominus})/\sigma^2$ a scaling factor that rescales the weight vector so that the mean distances per class are one unit of variance apart. In other words, by taking the scaling factor γ into account, we can drop our assumption that $\mathbf{w}$ is a unit vector.

If we also drop the assumption that $\overline{d}^{\oplus}$ and $\overline{d}^{\ominus}$ are symmetric around the decision boundary, then we obtain the most general form

Figure 7.11. The logistic function, a useful function for mapping distances from a linear decision boundary into an estimate of the positive posterior probability. The **fat** red line indicates the standard logistic function $\hat{p}(d) = \frac{1}{1+\exp(-d)}$; this function can be used to obtain probability estimates if the two classes are equally prevalent and the class means are equidistant from the decision boundary and one unit of variance apart. The steeper and flatter red lines show how the function changes if the class means are 2 and 1/2 units of variance apart, respectively. The three blue lines show how these curves change if $d_0 = 1$, which means that the positives are on average further away from the decision boundary.

$$LR = \frac{P(d(\mathbf{x})|\oplus)}{P(d(\mathbf{x})|\ominus)} = \exp\big(\gamma(d(\mathbf{x}) - d_0)\big) \tag{7.13}$$

$$\gamma = \frac{\overline{d}^{\oplus} - \overline{d}^{\ominus}}{\sigma^2} = \frac{\mathbf{w}\cdot(\boldsymbol{\mu}^{\oplus} - \boldsymbol{\mu}^{\ominus})}{\sigma^2}, \quad d_0 = \frac{\overline{d}^{\oplus} + \overline{d}^{\ominus}}{2} = \frac{\mathbf{w}\cdot(\boldsymbol{\mu}^{\oplus} + \boldsymbol{\mu}^{\ominus})}{2} - t$$

d_0 has the effect of moving the decision boundary from $\mathbf{w} \cdot \mathbf{x} = t$ to $\mathbf{x} = (\mu^{\oplus} + \mu^{\ominus})/2$, that is, halfway between the two class means. The logistic mapping thus becomes $d \mapsto \frac{1}{1+\exp(-\gamma(d-d_0))}$, and the effect of the two parameters is visualised in Figure 7.11.

Example 7.7 (Logistic calibration of a linear classifier). Logistic calibration has a particularly simple form for the basic linear classifier, which has $\mathbf{w} = \mu^{\oplus} - \mu^{\ominus}$. It follows that

$$\overline{d}^{\oplus} - \overline{d}^{\ominus} = \frac{\mathbf{w}\cdot(\boldsymbol{\mu}^{\oplus} - \boldsymbol{\mu}^{\ominus})}{\|\mathbf{w}\|} = \frac{\|\boldsymbol{\mu}^{\oplus} - \boldsymbol{\mu}^{\ominus}\|^2}{\|\boldsymbol{\mu}^{\oplus} - \boldsymbol{\mu}^{\ominus}\|} = \|\boldsymbol{\mu}^{\oplus} - \boldsymbol{\mu}^{\ominus}\|$$

and hence $\gamma = \|\mu^{\oplus} - \mu^{\ominus}\|/\sigma^2$. Furthermore, $d_0 = 0$ as $(\mu^{\oplus} + \mu^{\ominus})/2$ is already on the decision boundary. So in this case logistic calibration does not move the decision boundary, and only adjusts the steepness of the sigmoid according to the separation of the classes. Figure 7.12 illustrates this for some data sampled from two normal distributions with the same diagonal covariance matrix.

Figure 7.12. The surface shows the sigmoidal probability estimates resulting from logistic calibration of the basic linear classifier on random data satisfying the assumptions of logistic calibration.

To summarise: in order to get calibrated probability estimates out of a linear classifier, we first calculate the mean distances $\overline{d}^{\oplus}$ and $\overline{d}^{\ominus}$ and the variance σ^2, and from those the location parameter d_0 and the scaling parameter γ. The likelihood ratio is then $LR = \exp(\gamma(d(\mathbf{x}) - d_0)) = \exp(\gamma(\mathbf{w} \cdot \mathbf{x} - t - d_0))$. Since the logarithm of the likelihood ratio is linear in $\mathbf{x}$, such models are called *log-linear models*. Notice that $\gamma(\mathbf{w} \cdot \mathbf{x} - t - d_0) = \mathbf{w}' \cdot \mathbf{x} - t'$ with $\mathbf{w}' = \gamma\mathbf{w}$ and $t' = \gamma(t + d_0)$. This means that the logistic calibration procedure can change the location of the decision boundary but not its direction. However, there may be an alternative weight vector with a different direction that assigns a higher likelihood to the data. Finding the maximum-likelihood linear classifier using the logistic model is called ☞*logistic regression*, and will be discussed in Section 9.3.

As an alternative to logistic calibration, we can also use the isotonic calibration

method discussed in Section 2.3. Figure 7.13 (left) shows the ROC curve of the basic linear classifier on the data in Figure 7.12 as well as its convexhull. We can then construct a piecewise linear calibration function with plateaus corresponding to the convex hull segments, as shown in Figure 7.13 (right). In contrast with the logistic method this calibration method is non-parametric and hence does not make any assumptions about the data. In order to avoid overfitting, non-parametric methods typically need more data than parametric methods. It is interesting to note that no grading takes place on the plateaus, which are rather similar to the segments of a grouping model. In other words, convex hull calibration can potentially produce a hybrid between grouping and grading models.

Figure 7.13. (**left**) ROC curve and convex hull of the same model and data as in Figure 7.12. (**right**) The convex hull can be used as a non-parametric calibration method. Each segment of the convex hull corresponds to a plateau of the probability surface.

7.5 Going beyond linearity with kernel methods

In this chapter we have looked at linear methods for classification and regression. Starting with the leastsquares method for regression, we have seen how to adapt it to binary classification, resulting in a version of the basic linear classifier that takes feature correlation into account by constructing the matrix $(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}$ and is sensitive to unequal class distributions. We then looked at the heuristic perceptron algorithm for linearly separable data, and the support vector machine which finds the unique decision boundary with maximum margin and which can be adapted to non-separable data. In this section we show that these

techniques can be adapted to learn non-linear decision boundaries. The main idea is simple (and was already explored in Example 1.9 on p.43): to transform the data non-linearly to a feature space in which linear classification can be applied.

Figure 7.14.(**left**) Decision boundaries learned by the basic linear classifier and the perceptron using the square of the features. (**right**) Data and decision boundaries in the transformed feature space.

Example 7.8 (Learning a quadratic decision boundary). The data in Figure 7.14 (left) is not linearly separable, but both classes have a clear circular shape. Figure 7.14 (right) shows the same data with the feature values squared. In this transformed feature space the data has become linearly separable, and the perceptron is able to separate the classes. The resulting decision boundary in the original space is a near-circle. Also shown is the decision boundary learned by the basic linear classifier in the quadratic feature space, corresponding to an ellipse in the original space.

In general, mapping points back from the feature space to the instance space is non-trivial. E.g., in this example each class mean in feature space maps back to four points in the original space, owing to the quadratic mapping.

It is customary to call the transformed space the *feature space* and the original space the *input space*. The approach thus appears to be to transform the training data to feature space and learn a model there. In order to classify new data we

transform that to feature space as well and apply the model. However, the remarkable thing is that in many cases the feature space does not have to be explicitly constructed, as we can perform all necessary operations in input space.

Take the perceptron algorithm in dual form, for example (Algorithm 7.2 on p.209). The algorithm is a simple counting algorithm – the only operation that is somewhat involved is testing whether example **x**i is correctly classified by evaluating $y_i \sum_{j=1}^{|D|} \alpha_j y_j \mathbf{x}_i \cdot \mathbf{x}_j$. The key component of this calculation is the dot product $\mathbf{x}_i \cdot \mathbf{x}_j$. Assuming bivariate examples $\mathbf{x}_i = (x_i, y_i)$ and $\mathbf{x}_j = (x_j, y_j)$ for notational simplicity, the dot product can be written as $\mathbf{x}_i \cdot \mathbf{x}_j = x_i x_j + y_i y_j$. The corresponding instances in the quadratic feature space are (x_i^2, y_i^2) and $\left(x_j^2, y_j^2\right)$, and their dot product is

$$(x_i^2, y_i^2) \cdot \left(x_j^2, y_j^2\right) = x_i^2 x_j^2 + y_i^2 y_j^2$$

This is almost equal to

$$(\mathbf{x}_i \cdot \mathbf{x}_j)^2 = (x_i x_j + y_i y_j)^2 = (x_i x_j)^2 + (y_i y_j)^2 + 2x_i x_j y_i y_j$$

but not quite because of the third term of cross-products. We can capture this term by extending the feature vector with a third feature $\sqrt{2}xy$. This gives the following feature space:

$$\phi(\mathbf{x}_i) = \left(x_i^2, y_i^2, \sqrt{2}x_i y_i\right) \qquad \phi(\mathbf{x}_j) = \left(x_j^2, y_j^2, \sqrt{2}x_j y_j\right)$$
$$\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) = x_i^2 x_j^2 + y_i^2 y_j^2 + 2x_i x_j y_i y_j = (\mathbf{x}_i \cdot \mathbf{x}_j)^2$$

We now define $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^2$, and replace $\mathbf{x}_i \cdot \mathbf{x}_j$ with $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ in the dual perceptron algorithm to obtain the *kernel perceptron* (Algorithm 7.4), which is able to learn the kind of non-linear decision boundaries illustrated in Example 7.8.

The introduction of kernels opens up a whole range of possibilities. Clearly we can define a polynomial kernel of any degree p as $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^p$. This transforms a d-dimensional input space into a high-dimensional feature space, such that each new feature is a product of p terms (possibly repeated). If we include a constant, say $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^p$, we would get all lower-order terms as well. So, for example, in a bivariate input space and setting $p = 2$ the resulting feature space is

---

Algorithm 7.4: KemelPerceptron(D, κ) – perceptron training algorithm using a kernel.

```
Input  : labelled training data D in homogeneous coordinates;
         kernel function κ.
Output : coefficients α_i defining non-linear decision boundary.
α_i ← 0 for 1 ≤ i ≤ |D|;
converged←false;
while converged = false do
    converged←true;
    for i = 1 to |D| do
        if y_i Σ_{j=1}^{|D|} α_j y_j κ(x_i, x_j) ≤ 0 then
            α_i ← α_i + 1;
            converged←false;
        end
    end
end
```

$$\phi(\mathbf{x}) = \left(x^2, y^2, \sqrt{2}xy, \sqrt{2}x, \sqrt{2}y, 1\right)$$

with linear as well as quadratic features.

But we are not restricted to polynomial kernels. An often-used kernel is the *Gaussian kernel*, defined as

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right) \tag{7.14}$$

where σ is a parameter known as the *bandwidth*. To understand the Gaussian kernel a bit better, notice that $\kappa(\mathbf{x}, \mathbf{x}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}) = ||\phi(\mathbf{x})||^2$ for any kernel obeying a number of standard properties referred to as 'positive semi-definiteness'. In this case we have $\kappa(\mathbf{x}, \mathbf{x}) = 1$, which means that all points $\phi(\mathbf{x})$ lie on a hypersphere around the feature space origin – which is however of infinite dimension, so geometric considerations don't help us much here. It is more helpful to think of a Gaussian kernel as imposing a Gaussian (i.e., multivariate normal, see Background 9.1 on p.267) surface on each support vector in instance space, so that the decision boundary is defined in terms of those Gaussian surfaces.

Kernel methods are best known in combination with support vector machines. Notice that the soft margin optimisation problem (Equation 7.12 on p.217) is defined in terms of dot products between training instances and hence the 'kernel trick' can be applied:

$$\alpha_1^*, \ldots, \alpha_n^* = \underset{\alpha_1,\ldots,\alpha_n}{\operatorname{argmax}} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{n}\alpha_i$$

$$\text{subject to } 0 \le \alpha_i \le C \text{ and } \sum_{i=1}^{n}\alpha_i y_i = 0$$

One thing to keep in mind is that the decision boundary learned with a non-linear kernel cannot be represented by a simple weight vector in input space. Thus, in order to classify a new example **x** we need to evaluate $y_i \sum_{j=1}^{n} \alpha_j y_j \kappa(\mathbf{x}, \mathbf{x}_j)$ which is an $O(n)$ computation involving all training examples, or at least the ones with nonzero multipliers α_j. This is why support vector machines are a popular choice as a kernel method, since they naturally promote sparsity in the support vectors. Although we have restricted attention to numerical features here, it is worth stressing that kernels can be defined over discrete structures, including trees, graphs, and logical formulae, and thus open the way to extending geometric models to non-numerical data.

7.6 Linear models: Summary and further reading

After considering logical models in the previous three chapters we had a good look at linear models in this chapter. Logical models are inherently non-numerical, and so deal with numerical features by using thresholds to convert them into two or more intervals. Linear models are almost diametrically opposite in that they can deal with numerical features directly but need to pre-process non-numerical features.[3] Geometrically, linear models use lines and planes to build the model, which essentially means that a certain increase or decrease in one of the features has the same effect, regardless of that feature's value or any of the other features. They are simple and robust to variations in the training data, but sometimes suffer from underfitting as a consequence.

☞ In Section 7.1 we considered the leastsquares method that was originally conceived to solve a regression problem. This classical method, which derives its name from minimising the sum of squared residuals between predicted and actual function values, is described in innumerable introductory mathematics and engineering texts (and was one of the example programs I remember running on my father's Texas Instruments

TI-58 programmable calculator). We first had a look at the problem in univariate form, and then derived the general solution as $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathrm{X}^T\mathbf{y}$ where $(\mathbf{X}^T\mathbf{X})^{-1}$ is a transformation that decorrelates, centres and normalise the features. We then discussed regularised versions of linear regression: ridge regression was introduced by Hoerl and Kennard (1970), and the lasso which naturally leads to sparse solutions was introduced by Tibshira (1996). We saw how the leastsquares method could be applied to binary classification by encoding the classes by +1 and –1, leading to the solutior $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}(Pos\ \mu^{\oplus} - Neg\ \mu^{\ominus})$. This generalises the basic linear classifier by taking feature correlation and unequal class prevalence into account, bu at a considerably increased computational cost (quadratic in the number o instances and cubic in the number of features).

Section 7.2 presented another classical linear model, the perceptron. Unlik the leastsquares method, which always finds the optimal solution in terms of sum of squared residuals, the perceptron is a heuristic algorithm that depends, for one thing, on the order in which the examples are presented. Invented by Rosenblatt (1958), its convergence for linearly separable data was proved by Novikoff (1962), who gave an upper bound on the number of mistakes made before the perceptron converged. Minsky and Papert (1969) proved further formal properties of the perceptron, but also demonstrated the limitations of a linear classifier. These were overcome with the development, over an extended period of time and with contributions from many people, of the multilayer perceptron and its back-propagation training algorithm (Rumelhart, Hinton and Williams, 1986). I this section we also learned about the dual, instance-based view of linear classification in which we are learning instance weights rather than feature weights. For the perceptron these weights are the number of times the example has been misclassified during training.

Maximum-margin classification with support vector machines was the top of Section 7.3. The approach was proposed by Boser, Guyon and Vapnik (1992). Using the dual formulation, the instance weights are nonzero only for the support vectors, which are the training instances on the margin. Th soft-margin generalisation is due to Cortes and Vapnik (1995). Margin errors are allowed, but the total margin error is added as a regularisation term to the objective function to be minimised, weighted by the complexit

parameter C; all instances inside the margin receive instance weight C. As we have seen, by making C sufficiently small the support vector machine summarises the classes by their unweighted class means and hence is very similar to the basic linear classifier. A general introduction to SVMs is provided by Cristianini and Shawe-Taylor (2000). The sequential minimal optimisation algorithm is an often-used solver which iteratively selects pairs of multipliers to optimise analytically and is due to Platt (1998).

In Section 7.4 we considered two methods to turn linear classifiers into probability estimators by converting the signed distance from the decision boundary into class probabilities. One well-known method is to use the logistic function, either straight out of the box or by fitting location and spread parameters to the data. Although this is often presented as a simple trick, we saw how it can be justified by assuming that the distances per class are normally distributed with the same variance; this latter assumptic is needed to make the transformation monotonie. A non-parametric alternative is to use the ROC convex hull to obtain calibrated probability estimates. As was already mentioned in the summary of Chapter 2, the approach has its roots in isotonic regression (Best and Chakravarti, 1990) and was introduced to the machine learning community by Zadrozny and Elkan (2002). Fawcett and Niculescu-Mizil (2007) and Flach and Matsubara (2007) show its equivalence to calibration by means of the RO convex hull.

Finally, Section 7.5 discussed briefly how to go beyond linearity with kernel methods. The 'kernel trick' can be applied to any learning algorithr that can be entirely described in terms of dot products, which includes mo approaches discussed in this chapter. The beauty is that we are implicitly classifying in a high-dimensional feature space, without having to constru the space explicitly. I gave the kernel perceptron as a simple example of a kernelised algorithm; in the next chapter we will see another example. Shawe-Taylor and Cristianini (2004) provide an excellent reference bringing together a wealth of material on the use of kernels in machine learning, and Gärtner (2009) discusses how kernel methods can be applied to structured, non-numerical data.

[1] A more sophisticated algorithm can achieve $O(d^{2.8})$, but this is probably the best we can do.

[2] For instance, with sufficiently many examples this could be justified by the *central limit theorem*: the sum of a large number of identically distributed independent random variables is approximately normally distributed.

[3] Ways to pre-process non-numerical features for use in linear models are discussed in Chapter 10.

CHAPTER 8

Distance-based models

MANY FORMS OF LEARNING are based on generalising from training data to unseen data by exploiting the similarities between the two. With grouping models such as decision trees these similarities take the form of an equivalence relation or partition of the instance space: two instances are similar whenever they end up in the same segment of this partition. In this chapter we consider learning methods that utilise more graded forms of similarity. There are many different ways in which similarity can be measured, and in Section 8.1 we take a look at the most important of them. Section 8.2 is devoted to a discussion of two key concepts in distance-based machine learning: neighbours and exemplars. In Section 8.3 we consider what is perhaps the best-known distance-based learning method: the nearest-neighbour classifier. Section 8.4 investigates K-means clustering and close relatives, and Section 8.5 looks at hierarchical clustering by constructing dendrograms. Finally, in Section 8.6 we discuss how several of these methods can be extended using the kind of kernels that we saw in the previous chapter.

8.1 So many roads…

It may seem odd at first that there should be many ways to measure distance. I am not referring to the fact that distance can be measured on different scales (kilometres, miles, nautical miles, and so on), as such changes of scale are simple monotonic transformations and do not fundamentally alter the distance measure. A better intuition is obtained by taking the mode of travel into account. Clearly, when travelling from Bristol to Amsterdam by train you travel a larger

distance then when travelling by plane, because planes are less restricted in their paths than trains. We will explore this a bit further by considering the game of chess.

Figure 8.1. (left) Distance as experienced by a King on a chessboard: green squares are one move away, orange ones two moves and red ones three moves. The shape formed by equidistant squares from the current position is itself a square. (middle) A Rook can travel any number of squares in one move, but only horizontally or vertically. No square is further than two moves away. (right) The (fictional) KRook combines the restrictions of King and Rook: it can move only one square at a time, and only horizontally or vertically. Equidistant squares now form a lozenge.

In chess, each piece is governed by a set of rules that restrict its possible moves. These restrictions can be directional: for instance, King and Queen can move horizontally, vertically and diagonally, while a Bishop can only move diagonally, a Rook only horizontally and vertically, and pawns only upwards. King and pawn are further restricted by the fact that they can move only one square at a time, whereas Queen, Rook and Bishop can move any number of squares in a single allowed direction. Finally, a Knight moves according to a very specific pattern (one diagonal step and one horizontal or vertical step in a single move).

Although these pieces move around on the same board, they experience distances in very different ways. For example, the next square down is one move away for King, Queen and Rook; three moves away for a Knight; and unreachable for Bishop or pawn. This is, of course, very similar to our experience in the real world. Trains and cars can only move along tracks or roads, like a Bishop, which leaves remote places unreachable. A mountain range can mean large detours when travelling by car, train or on foot, but is easy to cross when flying. On an underground, two stations a few streets away may be only reachable with several changes of line, not unlike the way a Knight can reach a nearby square only in two or three moves. And on foot we are most flexible but also slow, like a King.

Figure 8.2. (left) The Bishop's world: squares are either one or two moves away, or else unreachable. **(right)** The fictional Bing combines the restrictions of King and Bishop: it can only move one square at a time, and only diagonally. Equidistant squares now form a punctuated square.

Figure 8.1 visualises the distances experienced by King and Rook. Both can reach all parts of the chessboard, but a Rook can travel much faster. In fact, a Rook can reach any square in either one or two moves (assuming no other pieces are in its way). All squares one move away form a cross, and the remaining squares are one additional move away. A King will often have to travel more than two moves to reach a particular square (although there are also squares that the King can reach in one move while a Rook needs two). The squares one move away form a small square shape around the current position; those two moves away form a larger square around the smaller square; and so on. Figure 8.1 (right) shows a piece that doesn't exist in chess, but could. It combines the restrictions of King and Rook, and I therefore call it a KRook. Like a King, it can only move one square at a time; and like a Rook, it can only move horizontally and vertically. For the KRook, equidistant squares form a sort of lozenge around the current position.

Figure 8.2 (left) visualises the Bishop's moves. The Bishop is somewhat similar to the Rook in that some squares (those of the same colour as its current square) are never more than two moves away; however, the remaining squares of the other colour are unreachable. Combining the restrictions of the Bishop (only diagonal moves) with those of the King (one square per move) we obtain another fictional piece, the Bing (Figure 8.2 (right)). We could say that the world of Bishops and Bings is rotated 45 degrees, compared with the world of Rooks and KRooks.

What's the relevance of all this when trying to understand distance-based machine learning, you may ask? Well, the rank (row) and file (column) on a chessboard is not unlike a discrete or categorical feature in machine learning (in fact, since ranks and files are ordered, they are ☞*ordinal features*, as we will further discuss in Chapter 10). We can switch to real-valued features by imagining a 'continuous' chessboard with infinitely many, infinitesimally narrow

ranks and files. Squares now become points, and distances are not expressed as the number of squares travelled, but simply as a real number on some scale. If we now look at the shapes obtained by connecting equidistant points, we see that many of these carry over from the discrete to the continuous case. For a King, for example, all points a given fixed distance away still form a square around the current position; and for a KRook they still form a square rotated 45 degrees. As it happens, these are special cases of the following generic concept.

Definition 8.1 (Minkowski distance). *If* $\mathscr{X} = \mathbb{R}^d$, *the* Minkowski distance *of order* $p > 0$ *is defined as*

$$\mathrm{Dis}_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{j=1}^{d} |x_j - y_j|^p \right)^{1/p} = ||\mathbf{x} - \mathbf{y}||_p$$

where $||\mathbf{z}||_p = \left(\sum_{j=1}^{d} |z_j|^p \right)^{1/p}$ *is the* p-norm *(sometimes denoted* L_p *norm) of the vector* $\mathbf{z}$. *We will often refer to* Dis_p *simply as the p-norm.*

So, the 2-*norm* refers to the familiar *Euclidean distance*

$$\mathrm{Dis}_2(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{j=1}^{d} (x_j - y_j)^2} = \sqrt{(\mathbf{x} - \mathbf{y})^{\mathrm{T}} (\mathbf{x} - \mathbf{y})}$$

which measures distance 'as the crow flies'. Two other values of p can be related back to the chess example. The 1-*norm* denotes *Manhattan distance*, also called *cityblock distance*:

$$\mathrm{Dis}_1(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{d} |x_j - y_j|$$

This is the distance if we can only travel along coordinate axes: similar to a taxi in Manhattan or other cities whose streets follow a regular grid pattern, but also the distance experienced by our fictional KRook piece. If we now let p grow larger, the distance will be more and more dominated by the largest coordinate-wise distance, from which we can infer that $\mathrm{Dis}_\infty(\mathbf{x}, \mathbf{y}) = \max_j |x_j - y_j|$. This is the distance experienced by the King on a chessboard, who can move diagonally as well as horizontally and vertically but only one step at a time; it is also called *Chebyshev distance*. Figure 8.3 (left) visualises equidistant points from the origin using Minkowski distances of various orders. It can be seen that

Euclidean distance is the only Minkowski distance that is rotationinvariant – in other words, special significance is given to the directions of the coordinate axes whenever $p \neq 2$. Minkowski distances do not refer to a particular choice of origin and are therefore translation-invariant, but none of them are scaling-invariant.

You will sometimes see references to the 0-*norm* (or L_0 norm) which counts the number of non-zero elements in a vector. The corresponding distance then counts the number of positions in which vectors **x** and **y** differ. This is not strictly a Minkowski distance; however, we can define it as

Figure 8.3. (left) Lines connecting points at order-p Minkowski distance 1 from the origin for (from inside) $p = 0.8$; $p = 1$ (Manhattan distance, the rotated square in red); $p = 1.5$; $p = 2$ (Euclidean distance, the violet circle); $p = 4$; $p = 8$; and $p = \infty$ (Chebyshev distance, the blue rectangle). Notice that for points on the coordinate axes all distances agree. For the other points, our reach increases with p; however, if we require a rotationinvariant distance metric then Euclidean distance is our only choice. (right) The rotated ellipse $\mathbf{x}^T\mathbf{R}^T\mathbf{S}^2\mathbf{R}\mathbf{x} = 1/4$; the axis-parallel ellipse $\mathbf{x}^T\mathbf{S}^2\mathbf{x} = 1/4$; and the circle $\mathbf{x}^T\mathbf{x} = 1/4$ (**R** and **S** as in Example 8.1).

$$\mathrm{Dis}_0(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{d} (x_j - y_j)^0 = \sum_{j=1}^{d} I[x_j = y_j]$$

under the understanding that $x^0 = 0$ for $x = 0$ and 1 otherwise. This is actually the distance experienced by a Rook on the chessboard: if both rank and file are different the square is two moves away, if only one of them is different the square is one move away. If **x** and **y** are binary strings, this is also called the *Hamming distance*. Alternatively, we can see the Hamming distance as the number of bits that need to be flipped to change **x** into **y;** for non-binary strings of unequal length this can be generalised to the notion of *edit distance* or *Levenshtein distance*.

Do all of these mathematical constructs make sense as a notion of distance? In

order to answer that question we can draw up a list of properties that a proper distance measure should have, such as non-negativity and symmetry. The generally agreedupon list defines what is known as a metric.

Definition 8.2 (Distance metric). *Given an instance space $\mathscr{X}$, a* distance metric *is a function* Dis $: \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ *such that for any* $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathscr{X}$:

1. *distances between a point and itself are zero:* $\mathrm{Dis}(x, x) = 0$;
2. *all other distances are larger than zero: if* $x \neq y$ *then* $\mathrm{Dis}(x, y) > 0$;
3. *distances are symmetric:* $\mathrm{Dis}(y, x) = \mathrm{Dis}(x, y)$;
4. *detours can not shorten the distance:* $\mathrm{Dis}(x, z) \leq \mathrm{Dis}(x, y) + \mathrm{Dis}(y, z)$.

If the second condition is weakened to a non-strict inequality – i.e., $\mathrm{Dis}(x, y)$ *may be zero even if* $x \neq y$ *– the function* Dis *is called a* pseudo-metric.

Figure 8.4. (left) The green circle connects points the same Euclidean distance (i.e., Minkowski distance of order $p = 2$) away from the origin as A. The orange circle shows that B and C are equidistant from A. The red circle demonstrates that C is closer to the origin than B, which conforms to the triangle inequality. **(middle)** With Manhattan distance ($p = 1$), B and C are equally close to the origin and also equidistant from A. **(right)** With $p < 1$ (here, $p = 0.8$) C is further away from the origin than B; since both are again equidistant from A, it follows that travelling from the origin to C via A is quicker than going there directly, which violates the triangle inequality.

The last condition is called the *triangle inequality* (or sub-additivity, as it really concerns the interaction between distance and addition). Figure 8.4 investigates this for Minkowski distances of various orders. The triangle inequality dictates that the distance from the origin to C is no more than the sum of the distances from the origin to A (Dis(O, A)) and from A to C (Dis(A, C)). B is at the same distance from A as C, regardless of the distance measure used; so Dis(O, A) + Dis(A, C) is equal to the distance from the origin to B. So, if we draw a circle around the origin through B, the triangle inequality dictates that C not be outside that circle. As we see in the left figure for Euclidean distance, B is the only point where the circles around the origin and around A intersect, so everywhere else the triangle inequality is a strict inequality.

The middle figure shows the same situation for Manhattan distance ($p = 1$). Now, B and C are in fact equidistant from the origin, and so travelling via A to C is no longer a detour, but just one of the many shortest routes. However, if we now decrease p further, we see that C ends up outside the red shape, and is thus further away than B when seen from the origin, whereas of course the sum of the distances from the origin to A and from A to C is still equal to the distance from the origin to B. At this point, our intuition breaks down: Minkowski distances with $p < 1$ are simply not very useful as distances since they all violate the triangle inequality.

Sometimes it is useful to use different scales for different coordinates if they are traversed with different speeds. For instance, for people horizontal distances can be traversed more easily than vertical differences, and consequently it is more realistic to use an ellipse rather than a circle to identify points that can be reached in a fixed amount of time, with the major axis of the ellipse indicating directions that can be traversed at larger speed. The ellipse can also be rotated, so that the major axis is not aligned with any of the coordinates: for instance, this could be the direction of a motorway, or the wind direction. Mathematically, while hyper-spheres (circles in $d \geq 2$ dimensions) of radius r can be defined by the equation $\mathbf{x}^T\mathbf{x} = r^2$, hyper-ellipses are defined by $\mathbf{x}^T\mathbf{M}\mathbf{x} = r^2$, where $\mathbf{M}$ is a matrix describing the appropriate rotation and scaling.

Example 8.1 (Elliptical distance). Consider the following matrices

$$\mathbf{R} = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix} \qquad \mathbf{S} = \begin{pmatrix} 1/2 & 0 \\ 0 & 1 \end{pmatrix} \qquad \mathbf{M} = \begin{pmatrix} 5/8 & -3/8 \\ -3/8 & 5/8 \end{pmatrix}$$

The matrix $\mathbf{R}$ describes a clockwise rotation of 45 degrees, and the diagonal matrix $\mathbf{S}$ scales the x-axis by a factor 1/2. The equation

$$(\mathbf{SRx})^T(\mathbf{SRx}) = \mathbf{x}^T\mathbf{R}^T\mathbf{S}^T\mathbf{SRx} = \mathbf{x}^T\mathbf{R}^T\mathbf{S}^2\mathbf{Rx} = \mathbf{x}^T\mathbf{Mx} = 1/4$$

describes a shape which, after clockwise rotation of 45 degrees and scaling of the x-axis by a factor 1/2, is a circle with radius 1/2 – i.e., the 'ascending' ellipse in Figure 8.3 (right). The ellipse equation is $(5/8)x^2 + (5/8)y^2 - (3/4)xy = 1/2$.

Often, the shape of the ellipse is estimated from data as the inverse of the covariance matrix: $\mathbf{M} = \boldsymbol{\Sigma}^{-1}$. This leads to the definition of the *Mahalanobis distance*

$$\mathrm{Dis}_M(\mathbf{x}, \mathbf{y}|\boldsymbol{\Sigma}) = \sqrt{(\mathbf{x}-\mathbf{y})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\mathbf{y})} \tag{8.1}$$

Using the covariance matrix in this way has the effect of decorrelating and normalising the features, as we saw in Section 7.1. Clearly, Euclidean distance is a special case of Mahalanobis distance with the identity matrix $\mathbf{I}$ as covariance matrix: $\mathrm{Dis}_2(\mathbf{x}, \mathbf{y}) = \mathrm{Dis}_M(\mathbf{x}, \mathbf{y}|\mathbf{I})$.

8.2 Neighbours and exemplars

Now that we understand the basics of measuring distance in instance space, we proceed to consider the key ideas underlying distance-based models. The two most important of these are: formulating the model in terms of a number of prototypical instances or *exemplars*, and defining the decision rule in terms of the nearest exemplars or *neighbours*. We can understand these concepts by revisiting our old friend, the basic linear classifier. This classifier uses the two class means $\boldsymbol{\mu}^{\oplus}$ and $\boldsymbol{\mu}^{\ominus}$ as exemplars, as a summary of all we need to know about the training data in order to build the classifier. A fundamental property of the mean of a set of vectors is that it minimises the sum of squared Euclidean distances to those vectors.

Theorem 8.1 (The arithmetic mean minimises squared Euclidean distance). *The arithmetic mean $\boldsymbol{\mu}$ of a set of data points D in a Euclidean space is the unique point that minimises the sum of squared Euclidean distances to those data points.*

Proof. We will show that $\mathrm{argmin}_{\mathbf{y}} \sum_{\mathbf{x}\in D} ||\mathbf{x}-\mathbf{y}||^2 = \boldsymbol{\mu}$, where $||.||$ denotes the 2-norm. We find this minimum by taking the gradient (the vector of partial derivatives with respect to y_i) of the sum and setting it to the zero vector:

$$\nabla_{\mathbf{y}} \sum_{\mathbf{x}\in D} ||\mathbf{x}-\mathbf{y}||^2 = -2\sum_{\mathbf{x}\in D}(\mathbf{x}-\mathbf{y}) = -2\sum_{\mathbf{x}\in D}\mathbf{x} + 2|D|\mathbf{y} = \mathbf{0}$$

from which we derive $\mathbf{y} = \frac{1}{|D|}\sum_{\mathbf{x}\in D}\mathbf{x} = \boldsymbol{\mu}$. ∎

Notice that minimising the sum of squared Euclidean distances of a given set

of points is the same as minimising the *average* squared Euclidean distance. You may wonder what happens if we drop the square here: wouldn't it be more natural to take the point that minimises total Euclidean distance as exemplar? This point is known as the *geometric median*, as for univariate data it corresponds to the median or 'middle value' of a set of numbers. However, for multivariate data there is no closed-form expression for the geometric median, which needs to be calculated by successive approximation. This computational advantage is the main reason why distance-based methods tend to use squared Euclidean distance.

In certain situations it makes sense to restrict an exemplar to be one of the given data points. In that case, we speak of a *medoid*, to distinguish it from a *centroid* which is an exemplar that doesn't have to occur in the data. Finding a medoid requires us to calculate, for each data point, the total distance to all other data points, in order to choose the point that minimises it. Regardless of the distance metric used, this is an $O(n^2)$ operation for n points, so for medoids there is no compuational reason to prefer one distance metric over another. Figure 8.5 shows a set of 10 data points where the different ways of determining exemplars all give different results. In particular, the mean and squared 2-norm medoid can be overly sensitive to outliers.

Once we have determined the exemplars, the basic linear classifier constructs the decision boundary as the perpendicular bisector of the line segment connecting the two exemplars. An alternative, distance-based way to classify instances without direct reference to a decision boundary is by the following decision rule: if $\mathbf{x}$ is nearest to $\boldsymbol{\mu}^{\oplus}$ then classify it as positive, otherwise as negative; or equivalently, classify an instance to the class of the *nearest* exemplar. If we use Euclidean distance as our closeness measure, simple geometry tells us we get exactly the same decision boundary (Figure 8.6 (left)).

Figure 8.5. A small data set of 10 points, with circles indicating centroids and squares indicating medoids (the latter must be data points), for different distance metrics. Notice how the outlier on the bottom-right 'pulls' the mean away from the geometric median; as a result the corresponding medoid changes as well.

So *the basic linear classifier can be interpreted from a distance-based perspective as constructing exemplars that minimise squared Euclidean distance within each class, and then applying a nearest-exemplar decision rule*. This change of perspective opens up many new possibilities. For example, we can investigate what the decision boundary looks like if we use Manhattan distance for the decision rule (Figure 8.6 (right)). It turns out that the decision boundary can only run along a number of fixed angles: in two dimensions these are horizontal, vertical and at (plus or minus) 45 degrees. This can be understood as follows. Suppose the two exemplars have different x- and y- coordinates, then they span a rectangle (I'll assume a tall rectangle, as in the figure). Imagine yourself in the centre of that rectangle, then clearly you are at equal distances from both exemplars (in fact, that same point is part of the 2-norm decision boundary). Now, imagine that you move one horizontal step, then you will move closer to one exemplar and away from the other; in order to compensate for that, you will also need to make a vertical step. So, within the rectangle, you maintain equal distance from the exemplars by moving at a 45 degree angle. Once you reach the perimeter of the rectangle you will walk away from both exemplars by making horizontal steps, so from there the decision boundary runs horizontally.

Figure 8.6. (left) For two exemplars the nearest-exemplar decision rule with Euclidean distance results in a linear decision boundary coinciding with the perpendicular bisector of the line connecting the two exemplars. The crosses denote different locations on the decision boundary, and the circles centred at those locations demonstrate that the exemplars are equidistant from each of them. When travelling along the decision boundary from bottom-left to top-right, these circles first shrink then grow again after passing the location halfway between the two exemplars. **(right)** Using Manhattan distance the circles are replaced by diamonds. Travelling from left to right, the diamonds shrink along the left-most horizontal segment of the decision boundary, then stay the same size along the 45-degree segment, and then grow again along the right-most horizontal segment.

Figure 8.7. (left) Decision regions defined by the 2-norm nearest-exemplar decision rule for three exemplars. **(right)** With Manhattan distance the decision regions become non-convex.

Figure 8.8. (left) Voronoi tesselation for five exemplars. **(middle)** Taking the two nearest exemplars into account leads to a further subdivision of each Voronoi cell. **(right)** The shading indicates which exemplars contribute to which cell.

Another useful consequence of switching to the distance-based perspective is that the nearest-exemplar decision rule works equally well for more than two exemplars, which gives us a multi-class version of the basic linear classifier.[1] Figure 8.7 (left) illustrates this for three exemplars. Each decision region is now bounded by two line segments. As you would expect, the 2-norm decision boundaries are more regular than the 1-norm ones: mathematicians say that the 2-norm decision regions are *convex*, which means that linear interpolation between any two points in the region can never go outside it. Clearly, this doesn't hold for 1-norm decision regions (Figure 8.7 (right)).

Increasing the number of exemplars further means that some of the regions become closed convex 'cells' (we are assuming Euclidean distance for the remainder of this section), giving rise to what is known as a *Voronoi tesselation*. Since the number of classes is typically much lower than the number of exemplars, decision rules often take more than one nearest exemplar into account. This increases the number of decision regions further.

Example 8.2 (Two neighbours know more than one). Figure 8.8 (left) gives a Voronoi tesselation for five exemplars. Each line segment is part of the perpendicular bisector of two exemplars. There are $\binom{5}{2} = 10$ pairs of exemplars, but two of these pairs are too far away from each other so we observe only eight line segments in the Voronoi tesselation.

If we now also take the second-nearest exemplars into account, each Voronoi cell is further subdivided: for instance, since the central point has four neighbours, the central cell is divided into four subregions (Figure 8.8 (middle)). You can think of those additional line segments as being part of the Voronoi tesselation that results when the central point is removed. The

other exemplars have only three immediate neighbours and so their cells are divided into three subregions. We thus obtain 16 '2-nearest exemplar' decision regions, each of which is defined by a different pair of nearest and second-nearest exemplars.

Figure 8.8 (right) shades each of these regions according to the two nearest exemplars spanning it. Notice that we gave each of the two exemplars the same weight, and so there are pairs of adjacent regions (across each of the original Voronoi boundaries) receiving the same shading, resulting in eight different shadings in all. This will be relevant later on, when we discuss the refinement of nearest-neighbour classifiers.

To summarise, the main ingredients of distance-based models are

- distance metrics, which can be Euclidean, Manhattan, Minkowski or Mahalanobis, among many others;
- exemplars: centroids that find a centre of mass according to a chosen distance metric, or medoids that find the most centrally located data point; and
- distance-based decision rules, which take a vote among the k nearest exemplars.

In the next sections these ingredients are combined in various ways to obtain supervised and unsupervised learning algorithms.

8.3 Nearest-neighbour classification

In the previous section we saw how to generalise the basic linear classifier to more than two classes, by learning an exemplar for each class and using the nearest-exemplar decision rule to classify new data. In fact, the most commonly used distance-based classifier is even more straightforward than that: it simply uses each training instance as an exemplar. Consequently, 'training' this classifier requires nothing more than memorising the training data. This extremely simple classifier is known as the *nearest-neighbour classifier*. Its decision regions are made up of the cells of a Voronoi tesselation, with piecewise

linear decision boundaries selected from the Voronoi boundaries (since adjacent cells may be labelled with the same class).

What are the properties of the nearest-neighbour classifier? First, notice that, unless the training set contains identical instances from different classes, we will be able to separate the classes perfectly on the training set – not really a surprise, as we memorised all training examples! Furthermore, by choosing the right exemplars we can more or less represent any decision boundary, or at least an arbitrarily close piecewise linear approximation. It follows that the nearest-neighbour classifier has low bias, but also high variance: move any of the exemplars spanning part of the decision boundary, and you will also change the boundary. This suggests a risk of overfitting if the training data is limited, noisy or unrepresentative.

From an algorithmic point of view, training the nearest-neighbour classifier is very fast, taking only $O(n)$ time for storing n exemplars. The downside is that classifying a single instance also takes $O(n)$ time, as the instance will need to be compared with every exemplar to determine which one is the nearest. It is possible to reduce classification time at the expense of increased training time by storing the exemplars in a more elaborate data structure, but this tends not to scale well to large numbers of features.

In fact, high-dimensional instance spaces can be problematic for another reason: the infamous *curse of dimensionality*. High-dimensional spaces tend to be extremely sparse, which means that every point is far away from virtually every other point, and hence pairwise distances tend to be uninformative. However, whether or not you are hit by the curse of dimensionality is not simply a matter of counting the number of features, as there are several reasons why the effective dimensionality of the instance space may be much smaller than the number of features. For example, some of the features may be irrelevant and drown out the relevant features' signal in the distance calculations. In such a case it would be a good idea, before building a distance-based model, to reduce dimensionality by performing ☞*feature selection*, as will be discussed in Chapter 10. Alternatively, the data may live on a *manifold* of lower dimension than the instance space (e.g., the surface of a sphere is a two-dimensional manifold wrapped around a three-dimensional object), which allows other dimensionality-reduction techniques such as ☞*principal component analysis*, which will be explained in the same chapter. In any case, before applying nearest-neighbour classification it is a good idea to plot a histogram of pairwise distances of a sample to see if they are sufficiently varied.

Notice that the nearest-neighbour method can easily be applied to regression problems with a real-valued target variable. In fact, the method is completely oblivious to the type of target variable and can be used to output text documents, images and videos. It is also possible to output the exemplar itself instead of a separate target, in which case we usually speak of *nearest-neighbour retrieval*. Of course we can only output targets (or exemplars) stored in the exemplar database, but if we have a way of aggregating these we can go beyond this restriction by applying the *k-nearest neighbour* method. In its simplest form, the k-nearest neighbour classifier takes a vote between the $k \geq 1$ nearest exemplars of the instance to be classified, and predicts the majority class. We can easily turn this into a probability estimator by returning the normalised class counts as a probability distribution over classes.

Figure 8.9. (left) Decision regions of a 3-nearest neighbour classifier; the shading represents the predicted probability distribution over the five classes. **(middle)** 5-nearest neighbour.**(right)** 7-nearest neighbour.

Figure 8.9 illustrates this on a small data set of 20 exemplars from five different classes, for $k = 3, 5, 7$. The class distribution is visualised by assigning each test point the class of a uniformly sampled neighbour: so, in a region where two of $k = 3$ neighbours are red and one is orange, the shading is a mix of two-thirds red and one-third orange. While for $k = 3$ the decision regions are still mostly discernible, this is much less so for $k = 5$ and $k = 7$. This may seem at odds with our earlier demonstration of the increase in the number of decision regions with increasing k in Example 8.2. However, this increase is countered by the fact that the probability vectors become more similar to each other. To take an extreme example: if k is equal to the number of exemplars n, every test instance will have the same number of neighbours and will receive the same probability vector which is equal to the prior distribution over the exemplars. If $k = n - 1$ we can reduce one of the class counts by 1, which can be done in c ways: the same number of possibilities as with $k = 1$!

We conclude that the refinement of k-nearest neighbour – the number of

different predictions it can make – initially increases with increasing k, then decreases again. Furthermore, we can say that the bias increases and the variance decreases with increasing k. There is no easy recipe to decide what value of k is appropriate for a given data set. However, it is possible to sidestep this question to some extent by applying *distance weighting* to the votes: that is, the closer an exemplar is to the instance to be classified, the more its vote counts. Figure 8.10 demonstrates this, using the reciprocal of the distance to an exemplar as the weight of its vote. This blurs the decision boundaries, as the model now applies a combination of grouping by means of the Voronoi boundaries, and grading by means of distance weighting. Furthermore, since the weights decrease quickly for larger distances, the effect of increasing k is much smaller than with unweighted voting. In fact, with distance weighting we can simply put $k = n$ and still obtain a model that makes different predictions in different parts of the instance space. One could say that distance weighting makes k-nearest neighbour more of a global model, while without it (and for small k) it is more like an aggregation of local models.

Figure 8.10. (left) 3-nearest neighbour with distance weighting on the data from Figure 8.9. **(middle)** 5-nearest neighbour. **(right)** 7-nearest neighbour.

If k-nearest neighbour is used for regression problems, the obvious way to aggregate the predictions from the k neighbours is by taking the mean value, which can again be distance-weighted. This would lend the model additional predictive power by predicting values that aren't observed among the stored exemplars. More generally, we can apply k-means to any learning problem where we have an appropriate 'aggregator' for multiple target values.

8.4 Distance-based clustering

In a distance-based context, unsupervised learning is usually taken to refer to

clustering, and we will now review a number of distance-based clustering methods. The ones considered in this section are all exemplar-based and hence predictive: they naturally generalise to unseen instances (see Section 3.3 for the distinction between predictive and descriptive clustering). In the next section we consider a clustering method that is not exemplar-based and hence descriptive.

Predictive distance-based clustering methods use the same ingredients as distance-based classifiers: a distance metric, a way to construct exemplars and a distance-based decision rule. In the absence of an explicit target variable, the assumption is that the distance metric indirectly encodes the learning target, so that we aim to find clusters that are *compact* with respect to the distance metric. This requires a notion of cluster compactness that can serve as our optimisation criterion. To that end, we refer back to the scatter matrix introduced in Background 7.2 on p.200.

Definition 8.3 (Scatter). *Given a data matrix* $\mathbf{X}$, *the* scatter matrix *is the matrix*

$$\mathbf{S} = (\mathbf{X} - \mathbf{1}\boldsymbol{\mu})^{\mathrm{T}}(\mathbf{X} - \mathbf{1}\boldsymbol{\mu}) = \sum_{i=1}^{n} (\mathbf{X}_{i\cdot} - \boldsymbol{\mu})^{\mathrm{T}}(\mathbf{X}_{i\cdot} - \boldsymbol{\mu})$$

where $\boldsymbol{\mu}$ *is a row vector containing all column means of* $\mathbf{X}$. *The* scatter *of* $\mathbf{X}$ *is defined as* $\mathrm{Scat}(\mathbf{X}) = \sum_{i=1}^{n} ||\mathbf{X}_{i\cdot} - \boldsymbol{\mu}||^2$, *which is equal to the trace of the scatter matrix (i.e., the sum of its diagonal elements).*

Imagine now that we partition D into K subsets $D_1 \uplus \ldots \uplus D_K = D$, and let $\boldsymbol{\mu}_j$ denote the mean of D_j . Let $\mathbf{S}$ be the scatter matrix of D, and $\mathbf{S}_j$ be the scatter matrices of D_j. These scatter matrices then have the following relationship:

$$\mathbf{S} = \sum_{j=1}^{K} \mathbf{S}_j + \mathbf{B} \tag{8.2}$$

Here, $\mathbf{B}$ is the scatter matrix that results by replacing each point in D with the corresponding $\boldsymbol{\mu}_j$. Each $\mathbf{S}_j$ is called a *within-cluster scatter matrix* and describes the compactness of the j-th cluster. $\mathbf{B}$ is the *between-cluster scatter matrix* and describes the spread of the cluster centroids. It follows that the traces of these matrices can be decomposed similarly, which gives

$$\mathrm{Scat}(D) = \sum_{j=1}^{K} \mathrm{Scat}(D_j) + \sum_{j=1}^{K} |D_j|\,||\boldsymbol{\mu}_j - \boldsymbol{\mu}||^2 \tag{8.3}$$

What this tells us is that minimising the total scatter over all clusters is

equivalent to maximising the (weighted) scatter of the centroids. The *K-means problem* is to find a partition that minimises the total within-cluster scatter.

Example 8.3 (Reducing scatter by partitioning data). Consider the following five points: (0, 3), (3, 3), (3, 0), (–2, –4) and (–4, –2). These points are, conveniently, centred around (0, 0). The scatter matrix is

$$\mathbf{S} = \begin{pmatrix} 0 & 3 & 3 & -2 & -4 \\ 3 & 3 & 0 & -4 & -2 \end{pmatrix} \begin{pmatrix} 0 & 3 \\ 3 & 3 \\ 3 & 0 \\ -2 & -4 \\ -4 & -2 \end{pmatrix} = \begin{pmatrix} 38 & 25 \\ 25 & 38 \end{pmatrix}$$

with trace Scat(D) = 76. If we cluster the first two points together in one cluster and the remaining three in another, then we obtain cluster means $\boldsymbol{\mu}_1$ = (1.5, 3) and $\boldsymbol{\mu}_2$ = (–1, –2) and within-cluster scatter matrices

$$\mathbf{S}_1 = \begin{pmatrix} 0-1.5 & 3-1.5 \\ 3-3 & 3-3 \end{pmatrix} \begin{pmatrix} 0-1.5 & 3-3 \\ 3-1.5 & 3-3 \end{pmatrix} = \begin{pmatrix} 4.5 & 0 \\ 0 & 0 \end{pmatrix}$$

$$\mathbf{S}_2 = \begin{pmatrix} 3-(-1) & -2-(-1) & -4-(-1) \\ 0-(-2) & -4-(-2) & -2-(-2) \end{pmatrix} \begin{pmatrix} 3-(-1) & 0-(-2) \\ -2-(-1) & -4-(-2) \\ -4-(-1) & -2-(-2) \end{pmatrix} = \begin{pmatrix} 26 & 10 \\ 10 & 8 \end{pmatrix}$$

with traces Scat(D_1) = 4.5 and Scat(D_2) = 34. Two copies of $\boldsymbol{\mu}_1$ and three copies of $\boldsymbol{\mu}_2$ have, by definition, the same centre of gravity as the complete data set: (0, 0) in this case. We thus calculate the between-cluster scatter matrix as

$$\mathbf{B} = \begin{pmatrix} 1.5 & 1.5 & -1 & -1 & -1 \\ 3 & 3 & -2 & -2 & -2 \end{pmatrix} \begin{pmatrix} 1.5 & 3 \\ 1.5 & 3 \\ -1 & -2 \\ -1 & -2 \\ -1 & -2 \end{pmatrix} = \begin{pmatrix} 7.5 & 15 \\ 15 & 30 \end{pmatrix}$$

with trace 37.5.

Alternatively, if we treat the first three points as a cluster and put the other two in a second cluster, then we obtain cluster means $\boldsymbol{\mu}'_1$ = (2, 2) and $\boldsymbol{\mu}'_2$ =

(–3, –3), and within-cluster scatter matrices

$$\mathbf{S}_1' = \begin{pmatrix} 0-2 & 3-2 & 3-2 \\ 3-2 & 3-2 & 0-2 \end{pmatrix} \begin{pmatrix} 0-2 & 3-2 \\ 3-2 & 3-2 \\ 3-2 & 0-2 \end{pmatrix} = \begin{pmatrix} 6 & -3 \\ -3 & 6 \end{pmatrix}$$

$$\mathbf{S}_2' = \begin{pmatrix} -2-(-3) & -4-(-3) \\ -4-(-3) & -2-(-3) \end{pmatrix} \begin{pmatrix} -2-(-3) & -4-(-3) \\ -4-(-3) & -2-(-3) \end{pmatrix} = \begin{pmatrix} 2 & -2 \\ -2 & 2 \end{pmatrix}$$

with traces $\mathrm{Scat}(D_1') = 12$ and $\mathrm{Scat}(D_2') = 4$. The between-cluster scatter matrix is

$$\mathbf{B}' = \begin{pmatrix} 2 & 2 & 2 & -3 & -3 \\ 2 & 2 & 2 & -3 & -3 \end{pmatrix} \begin{pmatrix} 2 & 2 \\ 2 & 2 \\ 2 & 2 \\ -3 & -3 \\ -3 & -3 \end{pmatrix} = \begin{pmatrix} 30 & 30 \\ 30 & 30 \end{pmatrix}$$

with trace 60. Clearly, the second clustering produces tighter clusters whose centroids are further apart.

K-means algorithm

The *K*-means problem is NP-complete, which means that there is no efficient solution to find the global minimum and we need to resort to a heuristic algorithm. The best-known algorithm is usually also called *K*-means, although the name 'Lloyd's algorithm' is also used. The outline of the algorithm is given in Algorithm 8.1. The algorithm iterates between partitioning the data using the nearest-centroid decision rule, and recalculating centroids from a partition. Figure 8.11 demonstrates the algorithm on a small data set with three clusters, and Example 8.4 gives the result on our example data set describing properties of different machine learning methods.

Figure 8.11. (left) First iteration of 3-means on Gaussian mixture data. The dotted lines are the Voronoi boundaries resulting from randomly initialised centroids; the violet solid lines are the result of the recalculated means. **(middle)** Second iteration, taking the previous partition as starting point (dotted line). **(right)** Third iteration with stable clustering.

Example 8.4 (Clustering MLM data). Refer back to the MLM data set in Table 1.4 on p.39 (it is also helpful to look at its two-dimensional approximation in Figure 1.7 on p.37). When we run K-means on this data with $K = 3$, we obtain the clusters {Associations, Trees, Rules}, {GMM, naive Bayes}, and a larger cluster with the remaining data points. When we run it with $K = 4$, we get that the large cluster splits into two: {kNN, Linear Classifier, Linear Regression} and

Algorithm 8.1: KMeans(D,K) – K-means clustering using Euclidean distance Dis_2.

Input : data $D \subseteq \mathbb{R}^d$; number of clusters $K \in \mathbb{N}$.
Output : K cluster means $\mu_1, \ldots, \mu_K \in \mathbb{R}^d$.

```
randomly initialise K vectors μ_1,...,μ_K ∈ ℝ^d;
repeat
    assign each x ∈ D to argmin_j Dis_2(x, μ_j);
    for j = 1 to K do
        D_j ← {x ∈ D | x assigned to cluster j};
        μ_j = (1/|D_j|) Σ_{x∈D_j} x;
    end
until no change in μ_1,...,μ_K;
return μ_1,...,μ_K;
```

Figure 8.12. (left) First iteration of 3-means on the same data as Figure 8.11 with differently initialised centroids. **(right)** 3-means has converged to a sub-optimal clustering.

{Kmeans, Logistic Regression, SVM}; but also that GMM gets reallocated to the latter cluster, and naive Bayes ends up as a singleton.

It can be shown that one iteration of K-means can never increase the within-cluster scatter, from which it follows that the algorithm will reach a *stationary point*: a point where no further improvement is possible. It is worth noting that even the simplest data set will have many stationary points.

Example 8.5 (Stationary points in clustering). Consider the task of dividing the set of numbers {8, 44, 50, 58, 84} into two clusters. There are four possible partitions that 2-means can find: {8}, {44, 50, 58, 84}; {8, 44}, {50, 58, 84}; {8, 44, 50}, {58, 84}; and {8, 44, 50, 58}, {84}. It is easy to verify that each of these establishes a stationary point for 2-means, and hence will be found with a suitable initialisation. Only the first clustering is optimal; i.e., it minimises the total within-cluster scatter.

In general, while K-means converges to a stationary point in finite time, no guarantees can be given about whether the convergence point is in fact the global minimum, or if not, how far we are from it. Figure 8.12 shows how an unfortunate initialisation of the centroids can lead to a sub-optimal solution. In

practice it is advisable to run the algorithm a number of times and select the solution with the smallest within-cluster scatter.

Clustering around medoids

It is straightforward to adapt the K-means algorithm to use a different distance metric; note that this will also change the objective function being minimised. Algorithm 8.2 gives the *K-medoids* algorithm, which additionally requires the exemplars to be data points. Notice that calculating the medoid of a cluster requires examining all pairs of points – whereas calculating the mean requires just a single pass through the points – which can be prohibitive for large data sets. Algorithm 8.3 gives an alternative algorithm called *partitioning around medoids* (*PAM*) that tries to improve a clustering locally by swapping medoids with other data points. The quality of a clustering Q is calculated as the total distance over all points to their nearest medoid. Notice that there are $k(n-k)$ pairs of one medoid and one non-medoid, and evaluating Q requires iterating over $n-k$ data points, so the computational cost of one iteration is quadratic in the number of data points. For large data sets one can run PAM on a small sample but evaluate Q on the whole data set, and repeat this a number of times for different samples.

An important limitation of the clustering methods discussed in this section is that they represent clusters only by means of exemplars. This disregards the shape of the clusters, and sometimes leads to counter-intuitive results. The two data sets in Figure 8.13 are identical, except for a rescaling of the y-axis. Nevertheless, K-means finds entirely different clusterings. This is not actually a shortcoming of the K-means algorithm as such, as in Figure 8.13 (right) the two centroids are further away than in the intended solution, and hence this represents a better solution in terms of Equation 8.3. The real issue is that in this case we want to estimate the 'shape' of the clusters as well as the cluster centroids, and hence take account of more than just the trace of the scatter matrices. We will discuss this further in the next chapter.

Algorithm 8 A.2: KMedoids(D, K, Dis) – K-medoids clustering using arbitrary distance metric Dis.

Input : data $D \subseteq \mathscr{X}$; number of clusters $K \in \mathbb{N}$;
distance metric $\mathrm{Dis} : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$.

Output : K medoids $\mu_1, \ldots, \mu_K \in D$, representing a predictive clustering of $\mathscr{X}$.

```
randomly pick K data points μ_1, ..., μ_K ∈ D;
repeat
    assign each x ∈ D to argmin_j Dis(x, μ_j);
    for j = 1 to k do
        D_j ← {x ∈ D | x assigned to cluster j};
        μ_j = argmin_{x ∈ D_j} Σ_{x' ∈ D_j} Dis(x, x');
    end
until no change in μ_1, ..., μ_K;
return μ_1, ..., μ_K;
```

Figure 8.13. (left) On this data 2-means detects the right clusters. (**right**) After rescaling the y-axis, this configuration has a higher between-cluster scatter than the intended one.

Algorithm 8.3: PAM(D, K, Dis) – Partitioning around medoids clustering using arbitrary distance metric Dis.

Input : data $D \subseteq \mathscr{X}$; number of clusters $K \in \mathbb{N}$;
distance metric $\mathrm{Dis} : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$.
Output : K medoids $\mu_1, \ldots, \mu_K \in D$, representing a predictive clustering of $\mathscr{X}$.

```
randomly pick K data points μ_1,...,μ_K ∈ D;
repeat
    assign each x ∈ D to argmin_j Dis(x, μ_j);
    for j = 1 to k do
        D_j ← {x ∈ D | x assigned to cluster j};
    end
    Q ← Σ_j Σ_{x∈D_j} Dis(x, μ_j);
    for each medoid m and each non-medoid o do
        calculate the improvement in Q resulting from swapping m with o;
    end
    select the pair with maximum improvement and swap;
until no further improvement possible;
return μ_1,...,μ_K;
```

Figure 8.14. (left) Silhouette for the clustering in Figure 8.13 (left), using squared Euclidean distance. Almost all points have a high $s(\mathbf{x})$, which means that they are much closer, on average, to the other members of their cluster than to the members of the neighbouring cluster. (**right**) The silhouette for the clustering in Figure 8.13 (right) is much less convincing.

Silhouettes

How could we detect the poor quality of the clustering in Figure 8.13 (right)? An interesting technique is the use of silhouettes. For any data point $\mathbf{x}_i$, let $d(\mathbf{x}_i, D_j)$ denote the average distance of $\mathbf{x}_i$ to the data points in cluster D_j, and let $j(i)$ denote the index of the cluster that $\mathbf{x}_i$ belongs to. Furthermore, let $a(\mathbf{x}_i) = d(\mathbf{x}_i,$

$D_{j(i)}$) be the average distance of $\mathbf{x}_i$ to the points in its own cluster $D_{j(i)}$, and $b(\mathbf{x}_i) = \min_{k \neq j(i)} d(\mathbf{x}_i, D_k)$ be the average distance to the points in its neighbouring cluster. We would expect $a(\mathbf{x}_i)$to be considerably smaller than $b(\mathbf{x}_i)$, but this cannot be guaranteed. So we can take the difference $b(\mathbf{x}_i) - a(\mathbf{x}_i)$ as an indication of how 'well-clustered' $\mathbf{x}_i$ is, and divide this by $b(\mathbf{x}_i)$ to obtain a number less than or equal to 1.

It is, however, conceivable that $a(\mathbf{x}_i) > b(\mathbf{x}_i)$, in which case the difference $b(\mathbf{x}_i) - a(\mathbf{x}_i)$ is negative. This describes the situation that, on average, the members of the neighbouring cluster are closer to $\mathbf{x}_i$ than the members of its own cluster. In order to get a normalised value we divide by $a(\mathbf{x}_i)$ in this case. This leads to the following definition:

$$s(\mathbf{x}_i) = \frac{b(\mathbf{x}_i) - a(\mathbf{x}_i)}{\max(a(\mathbf{x}_i), b(\mathbf{x}_i))} \tag{8.4}$$

A *silhouette* then sorts and plots $s(\mathbf{x})$ for each instance, grouped by cluster. Examples are shown in Figure 8.14 for the two clusterings in Figure 8.13. In this particular case we have used squared Euclidean distance in the construction of the silhouette, but the method can be applied to other distance metrics. We can clearly see that the first clustering is much better than the second. In addition to the graphical representation, we can compute average silhouette values per cluster and over the whole data set.

Figure 8.15. A dendrogram (printed left to right to improve readability) constructed by hierarchical clustering from the data in Table 1.4 on p.39.

8.5 Hierarchical clustering

The clustering methods discussed in the previous section use exemplars to represent a predictive clustering: a partition of the entire instance space. In this section we take a look at methods that represent clusters using trees. We previously encountered *clustering trees* in Section 5.3: those trees use features to navigate the instance space, similar to decision trees, and aren't distance-based as such. Here we consider trees called dendrograms, which are purely defined in terms of a distance measure. Because dendrograms use features only indirectly, as the basis on which the distance measure is calculated, they partition the given data rather than the entire instance space, and hence represent a descriptive clustering rather than a predictive one.

Example 8.6 (Hierarchical clustering of MLM data). We continue Example 8.4 on p.248. A hierarchical clustering of the MLM data is given in Figure 8.15. The tree shows that the three logical methods at the top form a strong cluster. If we wanted three clusters, we get the logical cluster, a second small cluster {GMM, naive Bayes}, and the remainder. If we wanted four clusters, we would separate GMM and naive Bayes, as the tree indicates this cluster is the least tight of the three (notice that this is slightly different from the solution found by 4-means). If we wanted five clusters, we would construct {Linear Regression, LinearClassifier} as a separate cluster. This illustrates the key advantage of hierarchical clustering: it doesn't require fixing the number of clusters in advance.

A precise definition of a dendrogram is as follows.

> **Definition 8.4 (Dendrogram).** *Given a data set D, a* dendrogram *is a binary tree with the elements of D at its leaves. An internal node of the tree represents the subset of elements in the leaves of the subtree rooted at that node. The level of a node is the distance between the two clusters represented by the children of the node. Leaves have level 0.*

For this definition to work, we need a way to measure how close two clusters are. You might think that this is straightforward: just calculate the distance between the two cluster means. However, this occasionally leads to problems, as

discussed later in this section. Furthermore, taking cluster means as exemplars assumes Euclidean distance, and we may want to use one of the other distance metrics discussed earlier. This has led to the introduction of the so-called linkage function, which is a general way to turn pairwise point distances into pairwise cluster distances.

Definition 8.5 (Linkage function). *A* linkage function $L : 2^{\mathscr{X}} \times 2^{\mathscr{X}} \to \mathbb{R}$ calculates the distance between arbitrary subsets of the instance space, given a distance metric $\mathrm{Dis}: \mathscr{X} \times \mathscr{X} \to \mathbb{R}$

The most common linkage functions are as follows:

Single linkage	defines the distance between two clusters as the *smallest* pairwise distance between elements from each cluster.
Complete linkage	defines the distance between two clusters as the *largest* pointwise distance.
Average linkage	defines the cluster distance as the *average* pointwise distance.
Centroid linkage	defines the cluster distance as the point distance between the cluster means.

These linkage functions can be defined mathematically as follows:

$$L_{\text{single}}(A, B) = \min_{x \in A, y \in B} \mathrm{Dis}(x, y)$$

$$L_{\text{complete}}(A, B) = \max_{x \in A, y \in B} \mathrm{Dis}(x, y)$$

$$L_{\text{average}}(A, B) = \frac{\sum_{x \in A, y \in B} \mathrm{Dis}(x, y)}{|A| \cdot |B|}$$

$$L_{\text{centroid}}(A, B) = \mathrm{Dis}\left(\frac{\sum_{x \in A} x}{|A|}, \frac{\sum_{y \in B} y}{|B|}\right)$$

Clearly, all these linkage functions coincide for singleton clusters: $L(\{x\}, \{y\}) = \mathrm{Dis}(x, y)$. However, for larger clusters they start to diverge. For example, suppose $\mathrm{Dis}(x, y) < \mathrm{Dis}(x, z)$, then the linkage between $\{x\}$ and $\{y, z\}$ is different in all four cases:

$$L_{\text{single}}(\{x\}, \{y, z\}) = \mathrm{Dis}(x, y)$$

$$L_{\text{complete}}(\{x\}, \{y, z\}) = \mathrm{Dis}(x, z)$$

$$L_{\text{average}}(\{x\}, \{y, z\}) = \big(\mathrm{Dis}(x, y) + \mathrm{Dis}(x, z)\big)/2$$

$$L_{\text{centroid}}(\{x\}, \{y, z\}) = \mathrm{Dis}(x, (y + z)/2)$$

The general algorithm to build a dendrogram is given in Algorithm 8.4. The tree is built from the data points upwards and is hence a bottom–up or *agglomerative* algorithm. At each iteration the algorithm constructs a new partition of the data by merging the two nearest clusters together. In general, the HAC algorithm gives different results when different linkage functions are used. Single linkage is the easiest case to understand, as it effectively builds a graph by adding increasingly longer links between points, one at a time, such that ultimately there is a path between any pair of points (hence the term 'linkage'). At any point during this process, the connected components are the clusters found at that iteration, and the linkage of the most recently found cluster is the length of the most recently added link. Hierarchical clustering using single linkage can essentially be done by calculating and sorting all pairwise distances between data points, which requires $O(n^2)$ time for n points. The other linkage functions require at least $O(n^2 \log n)$. Notice that the unoptimised algorithm in Algorithm 8.4 has time complexity $O(n^3)$.

Algorithm 8.4: HAC(D, L) – Hierarchical agglomerative clustering.

Input : data $D \subseteq \mathscr{X}$; linkage function $L : 2^{\mathscr{X}} \times 2^{\mathscr{X}} \to \mathbb{R}$ defined in terms of distance metric.

Output : a dendrogram representing a descriptive clustering of D.

```
initialise clusters to singleton data points;
create a leaf at level 0 for every singleton cluster;
repeat
    find the pair of clusters X, Y with lowest linkage l, and merge;
    create a parent of X, Y at level l;
until all data points are in one cluster;
return the constructed binary tree with linkage levels;
```

Figure 8.16. (left) Complete linkage defines cluster distance as the largest pairwise distance between elements from each cluster, indicated by the coloured lines between data points. The clustering found can be represented as nested partitions (bottom) or a dendrogram (top); the level of a horizontal connection between clusters in the dendrogram corresponds to the length of a linkage line. The example assumes that ties are broken by small irregularities in the grid. (**middle**) Centroid linkage defines the distance between clusters as the distance between their means. Notice that E obtains the same linkage as A and B, and so the latter clusters effectively disappear. (**right**) Single linkage defines the distance between clusters as the smallest pairwise distance. The dendrogramall but collapses, which means that no meaningful clusters are found in the given grid configuration.

Example 8.7 (Linkage matters). We consider a regular grid of 8 points in two rows of four (Figure 8.16). We assume that ties are broken by small irregularities. Each linkage function merges the same clusters in the same order, but the linkages are quite different in each case. Complete linkage gives the impression that D is far removed from the rest, whereas by moving D very slightly to the right it would have been added to E before C. With centroid linkage we see that E has in fact the same linkage as A and B, which means that A and B are not really discernible as separate clusters, even though they are found first. Single linkage seems preferable in this case, as it most clearly demonstrates that there is no meaningful cluster structure in this set of points.

Single and complete linkage both define the distance between clusters in terms of a particular pair of points. Consequently, they cannot take the shape of the cluster into account, which is why average and centroid linkage can offer an advantage. However, centroid linkage can lead to non-intuitive dendrograms, as

illustrated in Figure 8.17. The issue here is that we have $L(\{1\}, \{2\}) < L(\{1\}, \{3\})$ and $L(\{1\}, \{2\}) < L(\{2\}, \{3\})$ but $L(\{1\}, \{2\}) > L(\{1, 2\}, \{3\})$. The first two inequalities mean that 1 and 2 are the first to be merged into a cluster; but the second inequality means that the level of cluster {1, 2, 3} in the dendrogram drops below the level of {1, 2}. Centroid linkage violates the requirement of *monotonicity*, which stipulates that $L(A, B) < L(A, C)$ and $L(A, B) < L(B, C)$ implies $L(A, B) < L(A \cup B, C)$ for any clusters A, B and C. The other three linkage functions are monotonic (the example also serves as an illustration why average linkage and centroid linkage are not the same).

Figure 8.17. (left) Points 1 and 2 are closer to each other than to point 3. However, the distance between point 3 to the centroid of the other two points is less than any of the pairwise distances. **(right)** This results in a decrease in linkage when adding point 3 to cluster {1, 2}, and hence a non-monotonic dendrogram.

Another thing to keep in mind when constructing dendrograms is that the hierarchical clustering method is deterministic and will always construct a clustering. Consider Figure 8.18, which shows a data set of 20 uniformly randomly sampled points. One would be hard-pressed to find any cluster structure in this data; yet a dendrogram constructed with complete linkage and Euclidean distance appears to indicate that there are three or four clearly discernible clusters. But if we look closer, we see that the linkage levels are very close together in the bottom of the tree, and the fact that linkages are higher towards the top comes primarily from the use of complete linkage, which concentrates on maximal pairwise distances. The silhouette in Figure 8.18 (right) confirms that the cluster structure is not very strong. Effectively, we are witnessing here a particular, clustering-related kind of overfitting, already familiar from other tree-based models discussed in Chapter 5. Furthermore, dendrograms – like other tree models – have high variance in that small changes in the data points can lead to large changes in the dendrogram.

In conclusion, hierarchical clustering methods have the distinct advantage that the number of clusters does not need to be fixed in advance. However, this advantage comes at considerable computational cost. Furthermore, we now need to choose not just the distance measure used, but also the linkage function.

Figure 8.18. (left) 20 data points, generated by uniform random sampling. (**middle**) The dendrogram generated from complete linkage. The three clusters suggested by the dendrogram are spurious as they cannot be observed in the data. (**right**) The rapidly decreasing silhouette values in each cluster confirm the absence of a strong cluster structure. Point 18 has a negative silhouette value as it is on average closer to the green points than to the other red points.

8.6 From kernels to distances

In Section 7.5 we discussed how kernels can be used to extend the power of linear models considerably. Recall that a kernel is a function $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i).\phi(\mathbf{x}_j)$ that calculates a dot product in some feature space, but without constructing the feature vectors $\phi(\mathbf{x})$ explicitly. Any learning method that can be defined purely in terms of dot products of data points is amenable to such 'kernelisation'. Because of the close connection between Euclidean distance and dot products we can apply the same 'kernel trick' to many distance-based learning methods.

The key insight is that Euclidean distance can be rewritten in terms of dot products:

$$\mathrm{Dis}_2(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||_2 = \sqrt{(\mathbf{x}-\mathbf{y})\cdot(\mathbf{x}-\mathbf{y})} = \sqrt{\mathbf{x}\cdot\mathbf{x} - 2\mathbf{x}\cdot\mathbf{y} + \mathbf{y}\cdot\mathbf{y}}$$

This formula clearly shows that the distance between $\mathbf{x}$ and $\mathbf{y}$ decreases whenever the dot product $\mathbf{x} \cdot \mathbf{y}$ increases, which suggests that the dot product itself is a kind of similarity measure. However, it is not translation-invariant, because it depends on the location of the origin. The two terms $\mathbf{x} \cdot \mathbf{x}$ and $\mathbf{y} \cdot \mathbf{y}$ have the effect of making the overall expression translation-invariant. Replacing the dot product with a kernel function κ, we can construct the following

kernelised distance:

$$\mathrm{Dis}_{\kappa}(\mathbf{x},\mathbf{y}) = \sqrt{\kappa(\mathbf{x},\mathbf{x}) - 2\kappa(\mathbf{x},\mathbf{y}) + \kappa(\mathbf{y},\mathbf{y})} \tag{8.5}$$

It turns out that Dis_{κ} defines a pseudo-metric (see Definition 8.2 on p.235) whenever κ is a positive semi-definite kernel.[2]

As an illustration, Algorithm 8.5 adapts the ☞*K-means* algorithm (Algorithm 8.1 on p. 248) to use a kernelised distance. So, the algorithm clusters according to a nonlinear distance in instance space, corresponding to Euclidean distance in an implicit feature space. However, one complication arises, which is that Theorem 8.1 doesn't apply to nonlinear distances, and so we cannot construct cluster means in instance space. For this reason Algorithm 8.5 treats the clustering as a partition rather than a set of exemplars. Consequently, assigning each data point **x** to its nearest cluster (step 3) is now of quadratic complexity, since for each cluster we need to sum up the distances of all its members to **x**. In contrast, this step is linear in $|D|$ for the K-means algorithm.

There is an alternative way to turn dot products into distances. Since the dot product can be written as $||\mathbf{x}|| \cdot ||\mathbf{y}|| \cos\theta$, where θ is the angle between the vectors **x** and **y**, we define the *cosine similarity* as

$$\cos\theta = \frac{\mathbf{x}\cdot\mathbf{y}}{||\mathbf{x}||\cdot||\mathbf{y}||} = \frac{\mathbf{x}\cdot\mathbf{y}}{\sqrt{(\mathbf{x}\cdot\mathbf{x})(\mathbf{y}\cdot\mathbf{y})}} \tag{8.6}$$

Cosine similarity differs from Euclidean distance in that it doesn't depend on the length of the vectors **x** and **y**. On the other hand, it is not translation-independent, but assigns special status to the origin: one way to think of it is as a projection onto a unit sphere around the origin, and measuring distance on that sphere. Cosine similarity is usually turned into a distance metric by taking $1 - \cos\theta$. Being defined entirely in terms of dot products, it is as easily kernelised as Euclidean distance.

Algorithm 8.5: Kernel-KMeans(D, K) – K-means clustering using kernelised distance Dis_{κ}.

Input : data $D \subseteq \mathscr{X}$; number of clusters $K \in \mathbb{N}$.
Output : K-fold partition $D_1 \uplus \ldots \uplus D_K = D$.

```
randomly initialise K clusters D_1,...,D_K;
repeat
    assign each x ∈ D to argmin_j (1/|D_j|) Σ_{y∈D_j} Dis_κ(x,y);
    for j = 1 to K do
        D_j ← {x ∈ D | x assigned to cluster j};
    end
until no change in D_1,...,D_K;
return D_1,...,D_K;
```

8.7 Distance-based models: Summary and further reading

Along with linear models, distance-based models are the second group of models with strong geometric intuitions. The literature on distance-based models is rich and diverse; in this chapter I've concentrated on getting the main intuitions across.

- In Section 8.1 we reviewed the most commonly used distance metrics: the Minkowski distance or p-norm with special cases Euclidean distance ($p = 2$) and Manhattan distance ($p = 1$); the Hamming distance, which counts the number of bits or literals that are different; and the Mahalanobis distance, which decorrelates and normalises the features (Mahalanobis, 1936). Other distances can be taken into account, as long as they satisfy the requirements of a distance metric listed in Definition 8.2.

- Section 8.2 investigated the key concepts of neighbours and exemplars. Exemplars are either centroids that find a centre of mass according to a chosen distance metric, or medoids that find the most centrally located data point. The most commonly used centroid is the arithmetic mean, which minimises squared Euclidean distance to all other points. Other definitions of centroids are possible but harder to compute: e.g., the geometric median is the point minimising Euclidean distance, but does not admit a closed-

form solution. The complexity of finding a medoid is always quadratic regardless of the distance metric. We then considered nearest-neighbour decision rules, and looked in particular at the difference between 2-norm and 1-norm nearest-exemplar decision boundaries, and how these get refined by switching to a 2-nearest-exemplars decision rule.

- In Section 8.3 we discussed nearest-neighbour models which simply use the training data as exemplars. This is a very widely used model for classification, the origins of which can be traced back to Fix and Hodges (1951). Despite its simplicity, it can be shown that with sufficient training data the error rate is at most twice the optimal error rate (Cover and Hart, 1967). The 1-nearest neighbour classifier has low bias but high variance; by increasing the number of neighbours over which we aggregate we can reduce the variance but at the same time increase the bias. The nearest-neighbour decision rule can also be applied to real-valued target variables, and more generally to any task where we have an appropriate aggregator for multiple target values.

- Section 8.4 considered a number of algorithms for distance-based clustering using either arithmetic means or medoids. The *K*-means algorithm is a simple heuristic approach to solve the *K*-means problem that was originally proposed in 1957 and is sometimes referred to as Lloyd's algorithm (Lloyd, 1982). It is dependent on the initial configuration and can easily converge to the wrong stationary point. We also looked at the *K*-medoids and partitioning around medoids algorithms, the latter due to Kaufman and Rousseeuw (1990). These are computationally more expensive due to the use of medoids. Silhouettes (Rousseeuw, 1987) are a useful technique to check whether points are on average closer to the other members of their cluster than they are to the members of the neighbouring cluster. Much more detail about these and other clustering methods is provided by Jain, Murty and Flynn (1999).

- Whereas the previous clustering methods all result in a partition of the instance space and are therefore predictive, hierarchical clustering discussed in Section 8.5 applies only to the given data and is hence descriptive. A distinct advantage is that the clustering is constructed in the

form of a dendrogram, which means that the number of clusters does not need to be specified in advance and can be chosen by inspecting the dendrogram. However, the method is computationally expensive and infeasible for large data sets. Furthermore, it is not always obvious which of the possible linkage functions to choose.

Finally, in Section 8.6 we briefly considered how distances can be 'kernelised', and we gave one example in the form of kernel *K*-means. The use of a non-Euclidean distance metric leads to quadratic complexity of recalculating the clusters in each iteration.

---

[1] In information retrieval this is often called the *Rocchio classifier*.

[2] It is only a metric if the feature mapping ϕ is injective: suppose not, then some distinct $\mathbf{x}$ and $\mathbf{y}$ are mapped to the same feature vector $\phi(\mathbf{x}) = \phi(\mathbf{y})$, from which we derive $\kappa(\mathbf{x}, \mathbf{x}) - 2\kappa(\mathbf{x}, \mathbf{y}) + \kappa(\mathbf{y}, \mathbf{y}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}) - 2\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) + \phi(\mathbf{y}) \cdot \phi(\mathbf{y}) = 0$.

CHAPTER 9

Probabilistic models

THE THIRD AND FINAL FAMILY of machine learning models considered in this book are probabilistic models. We have already seen how probabilities can be useful to express a model's expectation about the class of a given instance. For example, a ☞ *probability estimation tree* (Section 5.2) attaches a class probability distribution to each leaf of the tree, and each instance that gets filtered down to a particular leaf in a tree model is labelled with that particular class distribution. Similarly, a calibrated linear model translates the distance from the decision boundary into a class probability (Section 7.4). These are examples of what are called *discriminative* probabilistic models. They model the posterior probability distribution $P(Y|X)$, where Y is the target variable and X are the features. That is, given X they return a probability distribution over Y.

The other main class of probabilistic models are called *generative* models. They model the joint distribution $P(Y, X)$ of the target Y and the feature vector X. Once we have access to this joint distribution we can derive any conditional or marginal distribution involving the same variables. In particular, since $P(X) = \sum_y P(Y = y, X)$ it follows that the posterior distribution can be obtained as

$$P(Y|X) = \frac{P(Y, X)}{\sum_y P(Y = y, X)}$$

Alternatively, generative models can be described by the likelihood function $P(X|Y)$, since $P(Y, X) = P(X|Y)\,P(Y)$ and the target or prior distribution (usually abbreviated to 'prior') can be easily estimated or postulated. Such models are called 'generative' because we can sample from the joint distribution to obtain new data points together with their labels. Alternatively, we can use $P(Y)$ to sample a class and $P(X|Y)$ to sample an instance for that class – this was illustrated for the spam e-mail example on p.29. In contrast, a discriminative model such as a probability estimation tree or a linear classifier models $P(Y|X)$ but not $P(X)$, and hence can be used to label data but not generate it.

Since generative models can do anything that discriminative models do, they may seem preferable. However, they have a number of drawbacks as well. First of all, note that storing the joint distribution requires space exponential in the

number of features. This necessitates simplifying assumptions such as independence between features, which may lead to inaccuracies if they are not valid in a particular domain. The most common criticism levied against generative models is that accuracy in modelling $P(X)$ may actually be achieved at the expense of less accurate modelling of $P(Y|X)$. However, the issue is not yet fully understood, and there are certainly situations where knowledge of $P(X)$ provides welcome additional understanding of the domain. For example, we may be less concerned about misclassifying certain instances if they are unlikely according to $P(X)$.

One of the most attractive features of the probabilistic perspective is that it allows us to view learning as a process of reducing uncertainty. For instance, a uniform class prior tells us that, before knowing anything about the instance to be classified, we are maximally uncertain about which class to assign. If the posterior distribution after observing the instance is less uniform, we have reduced our uncertainty in favour of one class or the other. We can repeat this process every time we receive new information, using the posterior obtained in the previous step as the prior for the next step. This process can be applied, in principle, to any unknown quantity that we come across.

Example 9.1 (Spam or not?). Suppose we want to estimate the probability θ that an arbitrary e-mail is spam, so that we can use the appropriate prior distribution. The natural thing to do is to inspect n e-mails, determine the number of spam emails d, and set $\hat{\theta} = d/n$] we don't really need any complicated statistics to tell us that. However, while this is the most likely estimate of θ – the maximum a posteriori (MAP) estimate, using the terminology introduced on p.28 – this doesn't mean that other values of θ are completely ruled out. We model this by a probability distribution over θ which is updated each time new information comes in. This is further illustrated in Figure 9.1 for a distribution that is more and more skewed towards spam.

Figure 9.1. Each time we inspect an e-mail, we are reducing our uncertainty regarding the prior spam probability θ. After we inspect two e-mails and observe one spam, the possible θ values are characterised by a symmetric distribution around 1/2. If we inspect a third, fourth, …, tenth e-mail and each time (except the first one) it is spam, then this distribution narrows and shifts a little bit to the right each time. As you would expect, the distribution for n e-mails reaches its maximum at $\hat{\theta}_{\text{MAP}} = \frac{n-1}{n}$ (e.g., $\hat{\theta}_{\text{MAP}} = 0.8$ for $n = 5$); however, asymmetric distributions like these contain information that cannot be conveyed by single numbers such as the mean or the maximum.

Explicitly modelling the posterior distribution over the parameter θ has a number of advantages that are usually associated with the 'Bayesian' perspective:

- We can precisely characterise the uncertainty that remains about our estimate by quantifying the spread of the posterior distribution.
- We can obtain a generative model for the parameter by sampling from the posterior distribution, which contains much more information than a summary statistic such as the MAP estimate can convey – so, rather than using a single e-mail with $\theta = \theta_{\text{MAP}}$, our generative model can contain a number of e-mails with θ sampled from the posterior distribution.
- We can quantify the probability of statements such as 'e-mails are biased towards ham' (the tiny shaded area in Figure 9.1 demonstrates that after observing one ham and nine spam e-mails this probability is very small, about 0.6%).
- We can use one of these distributions to encode our prior beliefs: e.g., if w believe that the proportions of spam and ham are typically 50–50, we can take the distribution for $n = 2$ (the lowest, symmetric one in Figure 9.1) as

our prior.[1]

The key point is that *probabilities do not have to be interpreted as estimates of relative frequencies, but can carry the more general meaning of (possibly subjective) degrees of belief*. Consequently, we can attach a probability distribution to almost anything: not just features and targets, but also model parameters and even models. For instance, in the example just given we were considering the distribution $P(\theta|D)$, where D represents the data (i.e., the classes of the inspected e-mails).

An important concept related to probabilistic models is *Bayes-optimality*. A classifier is Bayes-optimal if it always assigns argmax_y P* $(Y = x|X = x)$ to an instance x, where P^* denotes the true posterior distribution. Even if we almost never know the true distribution in a practical situation, there are several ways in which we can make this concrete. For example, we can perform experiments with artificially generated data for which we have chosen the true distribution ourselves: this allows us to experimentally evaluate how close the performance of a model is to being Bayes-optimal. Alternatively, the derivation of a probabilistic learning method usually makes certain assumptions about the true distribution, which allows us to prove theoretically that the model will be Bayes-optimal provided these assumptions are met. For example, later on in this chapter we will state the conditions under which the basic linear classifier is Bayes-optimal. The property is therefore best understood as a yardstick by which we measure the performance of probabilistic models.

Since many models discussed in previous chapters are able to estimate class probabilities and hence are discriminative probabilistic models, it is worth pointing out that the choice of a single model, often referred to as *model selection*, does not necessarily lead to Bayes-optimality – even if the model chosen is the one that performs best under the true distribution. To illustrate this, let m^* be the best probability estimation tree we have learned from a sufficient amount of data. Using m^* we would predict $\text{argmax}_y\ P\ (Y = y|M = m^*, X = x)$ for an instance x, where M is a random variable ranging over the model class m^* was chosen from. However, these predictions are not necessarily Bayes-optimal since

$$
\begin{aligned}
P(Y|X=x) &= \sum_{m \in M} P(Y, M=m|X=x) && \text{by marginalising over } M \\
&= \sum_{m \in M} P(Y|M=m, X=x)P(M=m|X=x) && \text{by the chain rule} \\
&= \sum_{m \in M} P(Y|M=m, X=x)P(M=m) && \text{by independence of } M \text{ and } X
\end{aligned}
$$

Here, $P(M)$ can be interpreted as a posterior distribution over models after seeing the training data (the MAP model is therefore $m^* = \text{argmax}_m P(M = m)$). The final expression in the preceding derivation tells us to average the predictions of all models, weighted by their posterior probabilities. Clearly, this distribution is only equal to $P(Y|M = m^*, X = x)$ if $P(M)$ is zero for all models other than m^*, i.e., if we have seen sufficient training data to rule out all but one remaining model. This is obviously unrealistic.[2]

The outline of the chapter is as follows. In Section 9.1 we will see some useful connections between the geometric perspective and the probabilistic viewpoint, which come about when features are normally distributed. This allows us, as already mentioned, to state the conditions under which the basic linear classifier is Bayes-optimal. In Section 9.2 we consider the case of categorical features, leading to the well-known naive Bayes classifier. Section 9.3 revisits the linear classifier from a probabilistic perspective, which results in a new training algorithm explicitly aimed at optimising the posterior probability of the examples. Section 9.4 discusses ways to deal with hidden variables. Finally, in Section 9.5 we briefly look at compression-based learning methods, which can be given a probabilistic interpretation by means of information-theoretic notions.

9.1 The normal distribution and its geometric interpretations

We can draw a connection between probabilistic and geometric models by considering probability distributions defined over Euclidean spaces. The most common such distributions are *normal distributions*, also called *Gaussians;* Background 9.1 recalls the most important facts concerning univariate and multivariate normal distributions. We start by considering the univariate, two-class case. Suppose the values of $x \in \mathbb{R}$ follow a *mixture model:* i.e., each class has its own probability distribution (a *component* of the mixture model). We will assume a Gaussian mixture model, which means that the components of the mixture are both Gaussians. We thus have

$$P(x|\oplus) = \frac{1}{\sqrt{2\pi}\sigma^{\oplus}} \exp\left(-\frac{1}{2}\left[\frac{x-\mu^{\oplus}}{\sigma^{\oplus}}\right]^2\right) \qquad P(x|\ominus) = \frac{1}{\sqrt{2\pi}\sigma^{\ominus}} \exp\left(-\frac{1}{2}\left[\frac{x-\mu^{\ominus}}{\sigma^{\ominus}}\right]^2\right)$$

where $\mu^{\oplus}$ and $\sigma^{\oplus}$ are the mean and standard deviation for the positive class, and

$\mu^{\ominus}$ and $\sigma^{\ominus}$ are the mean and standard deviation for the negative class. This gives the following likelihood ratio:

$$\mathrm{LR}(x) = \frac{P(x|\oplus)}{P(x|\ominus)} = \frac{\sigma^{\ominus}}{\sigma^{\oplus}} \exp\left(-\frac{1}{2}\left[\left(\frac{x-\mu^{\oplus}}{\sigma^{\oplus}}\right)^2 - \left(\frac{x-\mu^{\ominus}}{\sigma^{\ominus}}\right)^2\right]\right) \tag{9.1}$$

The univariate normal or Gaussian distribution has the following probability density function:

$$P(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) = \frac{1}{E}\exp\left(-\frac{1}{2}\left[\frac{x-\mu}{\sigma}\right]^2\right) = \frac{1}{E}\exp\left(-z^2/2\right), \quad E = \sqrt{2\pi}\sigma$$

The distribution has two parameters: μ, which is the mean or expected value, as well as the median (i.e., the point where the area under the density function is split in half) and the mode (i.e., the point where the density function reaches its maximum); and σ, which is the standard deviation and determines the width of the bell-shaped curve.
$z = (x - \mu)/\sigma$ is the *z-score* associated with x; it measures the number of standard deviations between x and the mean (it has itself mean 0 and standard deviation 1). It follows that $P(x|\mu, \sigma) = \frac{1}{\sigma} P(z|0, 1)$, where $P(z|0, 1)$ denotes the *standard normal distribution*. In other words, any normal distribution can be obtained from the standard normal distribution by scaling the x-axis with a factor σ, scaling the y-axis with a factor $1/\sigma$ (so the area under the curve remains 1), and translating the origin over μ. The *multivariate normal distribution* over d-vectors $\mathbf{x} = (x_1, \ldots, x_d)^{\mathrm{T}} \in \mathbb{R}^d$ is

$$P(\mathbf{x}|\mu,\Sigma) = \frac{1}{E_d}\exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\mu)\right), \quad E_d = (2\pi)^{d/2}\sqrt{|\Sigma|} \tag{9.2}$$

The parameters are the mean vector $\mu = (\mu_1, \ldots, \mu_d)^{\mathrm{T}}$ and the d-by-d covariance matrix $\mathbf{\Sigma}$ (see Background 7.2 on p.200). $\mathbf{\Sigma}^{-1}$ is the inverse of the covariance matrix, and $|\Sigma|$ is its determinant. The components of $\mathbf{x}$ may be thought of as d features that are possibly correlated.
If $d = 1$, then $\mathbf{\Sigma} = \sigma^2 = |\Sigma|$ and $\mathbf{\Sigma}^{-1} = 1/\sigma^2$, which gives us the univariate Gaussian as a special case. For $d = 2$ we have $\mathbf{\Sigma} = \begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{pmatrix}$, $|\mathbf{\Sigma}| = \sigma_1^2\sigma_2^2 - (\sigma_{12})^2$ and $\mathbf{\Sigma}^{-1} = \frac{1}{|\mathbf{\Sigma}|}\begin{pmatrix} \sigma_2^2 & -\sigma_{12} \\ -\sigma_{12} & \sigma_1^2 \end{pmatrix}$. Using z-

scores we derive the following expression for the bivariate normal distribution:

$$P(x_1, x_2|\mu_1, \mu_2, \sigma_1, \sigma_2, \rho) = \frac{1}{E_2} \exp\left(-\frac{1}{2(1-\rho^2)}(z_1^2 + z_2^2 - 2\rho z_1 z_2)\right), \quad E_2 = 2\pi\sigma_1\sigma_2\sqrt{1-\rho^2} \tag{9.3}$$

where $z_i = (x_i - \mu_i)/\sigma_t$ for $i = 1,2$, and $\rho = \sigma_{12}/\sigma_1\sigma_2$ is the *correlation coefficient* between the two features.

The *multivariate standard normal distribution* has $\boldsymbol{\mu} = \mathbf{0}$ (a d-vector with all 0s) and $\boldsymbol{\Sigma} = \mathbf{I}$ (the d-by-d identity matrix), and thus $P(\mathbf{x}|\mathbf{0}, \mathbf{I}) = \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}\mathbf{x}\cdot\mathbf{x}\right)$.

Background 9.1. The normal distribution.

Figure 9.2. If positive examples are drawn froma Gaussian with mean and standard deviation 1 and negatives from a Gaussian with mean and standard deviation 2, then the two distributions cross at $x = \pm 1.85$. This means that the maximum-likelihood region for positives is the closed interval [– 1.85, 1.85], and hence the negative region is non-contiguous.

Let's first consider the case that both components have the same standard deviation, i.e., $\sigma^{\oplus} = \sigma^{\ominus} = \sigma$. We can then simplify the exponent in Equation 9.1 as follows:

$$\begin{aligned}
-\frac{1}{2\sigma^2}\left[(x-\mu^{\oplus})^2 - (x-\mu^{\ominus})^2\right] &= -\frac{1}{2\sigma^2}\left[x^2 - 2\mu^{\oplus}x + \mu^{\oplus 2} - (x^2 - 2\mu^{\ominus}x + \mu^{\ominus 2})\right] \\
&= -\frac{1}{2\sigma^2}\left[-2(\mu^{\oplus} - \mu^{\ominus})x + (\mu^{\oplus 2} - \mu^{\ominus 2})\right] \\
&= \frac{\mu^{\oplus} - \mu^{\ominus}}{\sigma^2}\left[x - \frac{\mu^{\oplus} + \mu^{\ominus}}{2}\right]
\end{aligned}$$

The likelihood ratio can thus be written as $\mathrm{LR}(x) = \exp\,[\gamma(x - \mu))$, with two parameters: $\gamma = (\mu^{\oplus} - \mu^{\ominus})/\sigma^2$ is the difference between the means in proportion to the variance, and $\mu = (\mu^{\oplus} + \mu^{\ominus})/2$ is the midpoint between the two class means. It follows that the maximum-likelihood decision threshold (the value of x such that $\mathrm{LR}(x) = 1$) is $\boldsymbol{X}_{\mathrm{ML}} = \mu$.

If $\sigma^{\oplus} \neq \sigma^{\ominus}$, the x^2 terms in Equation 9.1 do not cancel. This results in two decision boundaries and a non-contiguous decision region for one of the classes.

Example 9.2 (Univariate mixture model with unequal variances). Suppose $\boldsymbol{\mu}^{\oplus} = 1$, $\mu^{\ominus} = 2$ and $\sigma^{\ominus} = 2\sigma^{\oplus} = 2$, then $\mathrm{LR}(x) = 2\exp(-[(x-1)^2 - (x-2)^2/4]/2) = 2\exp(3x^2/8)$. It follows that the ML decision boundaries are $x = \pm(8/3)\ln 2 = \pm 1.85$. As can be observed in Figure 9.2, these are the points where the two Gaussians cross. In contrast, if $\sigma^{\ominus} = \sigma^{\oplus}$ then we get a single ML decision boundary at $x = 1.5$.

Figure 9.3. (left) If the features are uncorrelated and have the same variance, maximum-likelihood classification leads to the basic linear classifier, whose decision boundary is orthogonal to the line connecting the means. (middle) As long as the per-class covariance matrices are identical, the Bayes-optimal decision boundary is linear – if we were to decorrelate the features by rotation and scaling, we would again obtain the basic linear classifier. (right) Unequal covariance matrices lead to hyperbolic decision boundaries, which means that one of the decision regions is non-contiguous.

Non-contiguous decision regions can also occur in higher-dimensional spaces. The following example demonstrates this for $m = 2$.

Example 9.3 (Bivariate Gaussian mixture). We use Equation 9.3 on p.267

to obtain explicit expressions for the ML decision boundary in the bivariate case. Throughout the example we assume $\mu_1^{\oplus} = \mu_2^{\oplus} = 1$ and $\mu_1^{\ominus} = \mu_2^{\ominus} = -1$.

(*i*) If all variances are 1 and both correlations are 0, then the ML decision boundary is given by $(x_1 - 1)^2 + (x_2 - 1)^2 - (x_1 + 1)^2 - (x_2 + 1)^2 = -2x_1 - 2x_2 - 2x_1 - 2x_2 = 0$, i.e., $x_1 + x_2 = 0$ (Figure 9.3 (left)).

(*ii*) If $\sigma_1^{\oplus} = \sigma_1^{\ominus} = 1, \sigma_2^{\oplus} = \sigma_2^{\ominus} = \sqrt{2}$ and $\rho^{\oplus} = \rho^{\ominus} = \sqrt{2}/2$, then the ML decision boundary is $(x_1 - 1)^2 + (x_2 - 1)^2/2 - \sqrt{2}(x_1 - 1)(x_2 - 1)/\sqrt{2} - (x_1 + 1)^2 - (x_2 + 1)^2/2 + \sqrt{2}(x_1 + 1)(x_2 + 1)/\sqrt{2} = -2x_1 = 0$ (Figure 9.3 (middle)).

(*iii*) If all variances are 1 and $\boldsymbol{\rho}^{\oplus} = -\rho^{\ominus} = \boldsymbol{\rho}$, then the ML decision boundary is given by $(x_1 - 1)^2 + (x_2 - 1)^2 - 2\rho(x_1 - 1)(x_2 - 1) - (x_1 + 1)^2 - (x_2 + 1)^2 - 2\rho(x_1 + 1)(x_2 + 1) = -4x_1 - 4x_2 - 4\rho x_1 x_2 - 4\rho = 0$, i.e., $x_1 + x_2 + \rho x_1 x_2 + \rho = 0$, which is a hyperbole. Figure 9.3 (right) illustrates this for $\rho = 0.7$. Notice that the bottom left of the instance space is a positive decision region, even though it contains no training examples and it is closer to the negative mean than to the positive mean.

Notice the circles and ellipses in Figure 9.3, which provide a visual summary of the covariance matrix. By projecting the shape for the positive class down to the x-axis we obtain the interval $[\mu_1^{\oplus} - \sigma_1^{\oplus}, \mu_1^{\oplus} + \sigma_1^{\oplus}]$ – i.e., one standard deviation around the mean – and similar for the negative class and the y-axis. Three cases can be distinguished: (*i*) both x and y standard deviations are equal and the correlation coefficient is zero, in which case the shape is a circle; (*ii*) the standard deviations are different and the correlation coefficient is zero, which means the shape is an ellipse parallel to the axis with the largest standard deviation; (*iii*) the correlation coefficient is non-zero: the orientation of the ellipse gives the sign of the correlation coefficient, and its width varies with the magnitude of the correlation coefficient.[3] Mathematically, these shapes are defined by setting $f(\mathbf{x})$ in $\frac{1}{E_d}\exp\left(-\frac{1}{2}f(\mathbf{x})\right)$ to 1 and solving for $\mathbf{x}$, in order to capture the points that are one standard deviation away from the mean. For the bivariate case this leads to $(z_1^2 + z_2^2 - 2\rho z_1 z_2) = 1 - \rho^2$, which can be translated into an elliptic equation for x_1 and x_2 by expanding the z-scores. Notice that for $\rho = 0$ this is a circle around the origin, and when $\rho \to 1$ this approaches the line $z_2 = z_1$ (we can't put $\rho = 1$ because this leads to a singular covariance matrix).

In the general multivariate case the condition $(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \mu) = 1$ defines a

hyperellipse, because $\boldsymbol{\Sigma}^{-1}$ satisfies certain properties.[4] For a standard normal distribution, one-standard-deviation contours lie on a hyper-sphere (a circle in d dimensions) defined by $\mathbf{x} \cdot \mathrm{x} = 1$. A very useful geometric intuition is that, just as hyper-spheres can be turned into arbitrary hyperellipses by scaling and rotation, any multivariate Gaussian can be obtained from the standard Gaussian by scaling and rotation (to obtain the desired covariance matrix) and translation (to obtain the desired mean). Conversely, we can turn an arbitrary multivariate Gaussian into a standard normal distribution by translation, rotation and scaling, as was already suggested in Background 1.2 on p.24. This results in decorrelated and normalised features.

The general form of the likelihood ratio can be derived from Equation 9.2 on p.267 as

$$\mathrm{LR}(\mathbf{x}) = \sqrt{\frac{|\boldsymbol{\Sigma}^{\ominus}|}{|\boldsymbol{\Sigma}^{\oplus}|}} \exp\left(-\frac{1}{2}\left[(\mathbf{x}-\boldsymbol{\mu}^{\oplus})^{\mathrm{T}}(\boldsymbol{\Sigma}^{\oplus})^{-1}(\mathbf{x}-\boldsymbol{\mu}^{\oplus}) - (\mathbf{x}-\boldsymbol{\mu}^{\ominus})^{\mathrm{T}}(\boldsymbol{\Sigma}^{\ominus})^{-1}(\mathbf{x}-\boldsymbol{\mu}^{\ominus})\right]\right)$$

where $\boldsymbol{\mu}^{\oplus}$ and $\mu^{\ominus}$ are the class means, and $\boldsymbol{\Sigma}^{\oplus}$ $\boldsymbol{\Sigma}^{\ominus}$ are the covariance matrices for each class. To understand this a bit better, assume that $\boldsymbol{\Sigma}^{\oplus} = \boldsymbol{\Sigma}^{\ominus} = \mathbf{I}$ (i.e., in each class the features are uncorrelated and have unit variance), then we have

$$\begin{aligned}\mathrm{LR}(\mathbf{x}) &= \exp\left(-\frac{1}{2}\left[(\mathbf{x}-\boldsymbol{\mu}^{\oplus})^{\mathrm{T}}(\mathbf{x}-\boldsymbol{\mu}^{\oplus}) - (\mathbf{x}-\boldsymbol{\mu}^{\ominus})^{\mathrm{T}}(\mathbf{x}-\boldsymbol{\mu}^{\ominus})\right]\right)\\ &= \exp\left(-\frac{1}{2}\left[\|\mathbf{x}-\boldsymbol{\mu}^{\oplus}\|^2 - \|\mathbf{x}-\boldsymbol{\mu}^{\ominus}\|^2\right]\right)\end{aligned}$$

It follows that $\mathrm{LR}(\mathbf{x}) = 1$ for any $\mathbf{x}$ equidistant from $\boldsymbol{\mu}^{\oplus}$ and $\mu^{\ominus}$. But this means that the ML decision boundary is a straight line at equal distances from the class means – in which we recognise our old friend, the basic linear classifier! In other words, *for uncorrelated, unit-variance Gaussian features, the basic linear classifier is Bayes-optimal*. This is a good example of how a probabilistic viewpoint can justify particular models.

More generally, as long as the per-class covariance matrices are equal, the ML decision boundary will be linear, intersecting $\boldsymbol{\mu}^{\oplus} - \mu^{\ominus}$ in the middle, but not at right angles if the features are correlated. This means that the basic linear classifier is only Bayes-optimal in this case if we first decorrelate and normalise the features. With non-equal class covariances the decision boundary will be hyperbolic. So, the three cases in Figure 9.3 generalise to the multivariate case.

We have now seen several examples of how the normal distribution links the probabilistic and geometric viewpoints. The multivariate normal distribution essentially translates distances into probabilities. This becomes obvious when we

plug the definition of ☞*Mahalanobis distance* (Equation 8.1 on p.237) into Equation 9.2:

$$P(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{E_d}\exp\left(-\frac{1}{2}\left(\mathrm{Dis}_M(\mathbf{x},\boldsymbol{\mu}|\boldsymbol{\Sigma})\right)^2\right) \tag{9.4}$$

Similarly, the standard normal distribution translates Euclidean distances into probabilities:

$$P(\mathbf{x}|\mathbf{0},\mathbf{I}) = \frac{1}{(2\pi)^{d/2}}\exp\left(-\frac{1}{2}\left(\mathrm{Dis}_2(\mathbf{x},\mathbf{0})\right)^2\right)$$

Conversely, we see that *the negative logarithm of the Gaussian likelihood can be interpreted as a squared distance*:

$$-\ln P(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \ln E_d + \frac{1}{2}\left(\mathrm{Dis}_M(\mathbf{x},\boldsymbol{\mu}|\boldsymbol{\Sigma})\right)^2$$

The intuition is that the logarithm transforms the multiplicative probability scale into an additive scale (which, in the case of Gaussian distributions, corresponds to a squared distance). Since additive scales are often easier to handle, log-likelihoods are a common concept in statistics.

Another example of the link between the geometric and the probabilistic perspective occurs when we consider the question of estimating the parameters of a normal distribution. For example, suppose we want to estimate the mean $\boldsymbol{\mu}$ of a multivariate Gaussian distribution with given covariance matrix $\boldsymbol{\Sigma}$ from a set of data points X. The principle of *maximum-likelihood estimation* states that we should find the value of $\boldsymbol{\mu}$ that maximises the joint likelihood of X. Assuming that the elements of X were independently sampled, the joint likelihood decomposes into a product over the individual data points in X, and the maximum-likelihood estimate can be found as follows:

$$\begin{aligned}
\hat{\boldsymbol{\mu}} &= \underset{\boldsymbol{\mu}}{\arg\max} \prod_{\mathbf{x}\in X} P(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) \\
&= \underset{\boldsymbol{\mu}}{\arg\max} \prod_{\mathbf{x}\in X} \frac{1}{E_d}\exp\left(-\frac{1}{2}\left(\mathrm{Dis}_M(\mathbf{x},\boldsymbol{\mu}|\boldsymbol{\Sigma})\right)^2\right) && \text{using Equation 9.4} \\
&= \underset{\boldsymbol{\mu}}{\arg\min} \sum_{\mathbf{x}\in X} \left[\ln E_d + \frac{1}{2}\left(\mathrm{Dis}_M(\mathbf{x},\boldsymbol{\mu}|\boldsymbol{\Sigma})\right)^2\right] && \text{taking negative logarithms} \\
&= \underset{\boldsymbol{\mu}}{\arg\min} \sum_{\mathbf{x}\in X} \left(\mathrm{Dis}_M(\mathbf{x},\boldsymbol{\mu}|\boldsymbol{\Sigma})\right)^2 && \text{dropping constant term and factor}
\end{aligned}$$

We thus find that the maximum-likelihood estimate of the mean of a multivariate

distribution is the point that minimises the total squared Mahalanobis distance to all points in X. For the identity covariance matrix $\mathbf{\Sigma} = \mathbf{I}$ we can replace Mahalanobis distance with Euclidean distance, and by Theorem 8.1 the point minimising total squared Euclidean distance to all points in X is the arithmetic mean $\frac{1}{|X|}\sum_{\mathbf{x}\in X}\mathbf{x}$.

As a final example of how geometric and probabilistic views of the same problem can be strongly connected I will now demonstrate how the ☞ *least-squares solution to a linear regression problem* (Section 7.1) can be derived as a maximum-likelihood estimate. For ease of notation we will look at the univariate case discussed in Example 7.1. The starting point is the assumption that our training examples (h_i, y_i) are noisy measurements of true function points $(x_i, f(x_i))$: i.e., $y_i = f(x_i) + \epsilon_i$, where the ϵ_i are independently and identically distributed errors. (Notice the slight change of notation as y_i is now no longer the true function value.) We want to derive the maximum-likelihood estimates $\hat{y}_i$ of $f(x_i)$. We can derive this if we assume a particular noise distribution, for example Gaussian with variance σ_2. It then follows that each y_i is normally distributed with mean $a + bx_i$ and variance σ^2, and thus

$$P(y_i|a,b,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(y_i-(a+bx_i))^2}{2\sigma^2}\right)$$

Since the noise terms ϵ_i are independent for different i, so are the y_i and so the joint probability over all i is simply the product of n of these Gaussians:

$$\begin{aligned}P(y_1,\ldots,y_n|a,b,\sigma^2) &= \prod_{i=1}^{n}\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(y_i-(a+bx_i))^2}{2\sigma^2}\right)\\ &= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n\exp\left(-\frac{\sum_{i=1}^{n}(y_i-(a+bx_i))^2}{2\sigma^2}\right)\end{aligned}$$

For ease of algebraic manipulation we take the negative natural logarithm:

$$-\ln P(y_1,\ldots,y_n|a,b,\sigma^2) = \frac{n}{2}\ln 2\pi + \frac{n}{2}\ln\sigma^2 + \frac{\sum_{i=1}^{n}(y_i-(a+bx_i))^2}{2\sigma^2}$$

Taking the partial derivatives with respect to a, b and σ^2 and setting to zero in order to maximise the negative log likelihood gives the following three equations:

$$\sum_{i=1}^{n} y_i - (a + bx_i) = 0$$

$$\sum_{i=1}^{n} \left(y_i - (a + bx_i)\right) x_i = 0$$

$$\frac{n}{2}\frac{1}{\sigma^2} - \frac{\sum_{i=1}^{n} \left(y_i - (a + bx_i)\right)^2}{2(\sigma^2)^2} = 0$$

The first two equations are essentially the same as derived in Example 7.1 and give us $\hat{a} = \bar{y} - \hat{b}\,\bar{x}$ and $\hat{b} = \sigma_{xy}/\sigma_{xx}$, respectively. The third equation tells us that the sum of squared residuals is equal to $n\sigma^2$ and gives the maximum-likelihood estimate of the noise variance as $\left(\sum_{i=1}^{n} \left(y_i - (a + bx_i)\right)^2\right)/n$.

It is reassuring that the probabilistic viewpoint allows us to derive (ordinary) east-squares regression from first principles. On the other hand, a full treatment would require noise on the x-values as well (total least squares), but this complicates the mathematics and does not necessarily have a unique solution. This illustrates that *a good probabilistic treatment of a machine learning problem achieves a balance between solid theoretical foundations and the pragmatism required to obtain a workable solution*.

9.2 Probabilistic models for categorical data

To kill time during long drives to some faraway holiday destination, my sisters and I would often play games involving passing cars. For example, we would ask each other to look out for cars that had a particular colour, were from a particular country or had a particular letter on the numberplate. A binary question such as 'is the car blue?' is called a *Bernoulli trial* by statisticians. They are modelled as a binary random variable whose probability of success is fixed over each independent trial. We used a Bernoulli distribution to model the event of an e-mail being ham in Example 9.1. On top of such a random variable, other probability distributions can be built. For example, we may want to guess how many of the next n cars are blue: this is governed by the binomial distribution. Or the task may be to estimate how many cars we need to see until the first Dutch one: this number follows a geometric definition. Background 9.2 will help to refresh your memory regarding the main definitions.

Categorical variables or features (also called discrete or nominal) are ubiquitous in machine learning. Perhaps the most common form of the Bernoulli distribution models whether or not a word occurs in a document. That is, for the

i-th word in our vocabulary we have a random variable X_i governed by a Bernoulli distribution. The joint distribution over the *bit vector* $X = (X_1, \ldots, X_k)$ is called a *multivariate Bernoulli distribution*. Variables with more than two outcomes are also common: for example, every word position in an e-mail corresponds to a categorical variable with k outcomes, where k is the size of the vocabulary. The multinomial distribution manifests itself as a *count vector:* a histogram of the number of occurrences of all vocabulary words in a document. This establishes an alternative way of modelling text documents that allows the number of occurrences of a word to influence the classification of a document.

The *Bernoulli distribution*, named after the Swiss seventeenth century mathematician Jacob Bernoulli, concerns Boolean or binary events with two possible outcomes: success or 1, and failure or 0. A Bernoulli distribution has a single parameter θ which gives the probability of success: hence $P(X = 1) = \theta$ and $P(X = 0) = 1 - \theta$. The Bernoulli distribution has expected value $\mathbb{E}[X] = \theta$ and variance $\mathbb{E}[(X - \mathbb{E}[X])^2] = \theta(1 - \theta)$.
The *binomial distribution* arises when counting the number of successes S in n independent Bernoulli trials with the same parameter θ. It is described by

$$P(S = s) = \binom{n}{s}\theta^s(1-\theta)^{n-s} \text{ for } s \in \{0, \ldots, n\}$$

This distribution has expected value $\mathbb{E}[S] = n\theta$ and variance $\mathbb{E}[(S - \mathbb{E}[S])^2] = n\theta(1 - \theta)$. The *categorical distribution* generalises the Bernoulli distribution to $k \geq 2$ outcomes. The parameter of the distribution is a k-vector $\theta = (\theta_1, \ldots, \theta_k)$ such that $\sum_{i=1}^{k} \theta_i = 1$.
Finally, the *multinomial distribution* tabulates the outcomes of n independent and identically distributed (i.i.d.) categorical trials. That is, $\mathbf{X} = (X_1, \ldots, X_k)$ is a k-vector of integer counts, and

$$P(\mathbf{X} = (x_1, \ldots, x_k)) = n!\frac{\theta_1^{x_1}}{x_1!} \cdots \frac{\theta_k^{x_k}}{x_k!}$$

with $\sum_{i=1}^{k} x_i = n$. Notice that setting $n = 1$ gives us an alternative way of stating the categorical distribution as $P(\mathbf{X} = (x_1, \ldots, x_k)) = \theta_1^{x_1} \cdots \theta_k^{x_k}$, with

exactly one of the x_i equal to 1 and the rest set to 0. Furthermore, setting $k = 2$ gives an alternative expression for the Bernoulli distribution as $P(X = x) = \theta^x (1 - \theta)^{1-x}$ for $x \in \{0,1\}$. It is also useful to note that if **X** follows a multinomial distribution, then each component X_i follows a binomial distribution with parameter θ_i.

We can estimate the parameters of these distributions by counting in a straightforward way. Suppose *a b a c c b a a b c* is a sequence of words. We might be interested in individual words being *a* or not, and interpret the data as coming from 10 i.i.d. Bernoulli trials, which would allow us to estimate $\hat{\theta}_a = 4/10 = 0.4$. This same parameter generates a binomial distribution of the number of occurrences of the word *a* in similar sequences. Alternatively, we can estimate the parameters of the categorical (word occurrences) and multinomial (word counts) distributions as $\hat{\theta} = (0.4, 0.3, 0.3)$.

It is almost always a good idea to smooth these distributions by including *pseudo-counts*. Imagine our vocabulary includes the word *d* but we haven't yet observed it, then a maximum-likelihood estimate would set $\hat{\theta}_d = 0$. We can smooth this by adding a virtual occurrence of each word to our observations, leading to $\hat{\theta}' = (5/14, 4/14, 4/14, 1/14)$. In the case of a binomial this is the Laplace correction.

Background 9.2. Probability distributions for categorical data.

Both these document models are in common use. Despite their differences, they both assume independence between word occurrences, generally referred to as the *naive Bayes assumption*. In the multinomial document model, this follows from the very use of the multinomial distribution, which assumes that words at different word positions are drawn independently from the same categorical distribution. In the multivariate Bernoulli model we assume that the bits in a bit vector are statistically independent, which allows us to compute the joint probability of a particular bit vector $(x_1,\ldots,x_k)$ as the product of the probabilities of each component $P(X_i = x_i)$. In practice, such word independence assumptions are often not true: if we know that an email contains the word 'Viagra', we can be quite sure that it will also contain the word 'pill'. In any case, the experience is that, while the naive Bayes assumption almost certainly leads to poor probability estimates, it often doesn't harm ranking performance. This means that, provided the classification threshold is chosen with some care, we can

usually get good classification performance too.

Using a naive Bayes model for classification

Assume that we have chosen one of the possible distributions to model our data X. In a classification context, we furthermore assume that the distribution depends on the class, so that $P(X|Y = \text{spam})$and $P(X|Y = \text{ham})$ are different distributions. The more different these two distributions are, the more useful the features X are for classification. Thus, for a specific e-mail x we calculate both $P(X = x|Y = \text{spam})$and $P(X = x|Y = \text{ham})$, and apply one of several possible decision rules:

maximum likelihood (ML)	– predict $\text{argmax}_y\, P(X = x|Y = y)$;
maximum a posteriori (MAP)	– predict $\text{argmax}_y\, P(X = x|Y = y)P(Y = y)$;
recalibrated likelihood	– predict $\text{argmax}_y\, WyP(X = x|Y = y)$.

The relation between the first two decision rules is that ML classification is equivalent to MAP classification with a uniform class distribution. The third decision rule generalises the first two in that it replaces the class distribution with a set of weights learned from the data: this makes it possible to correct for estimation errors in the likelihoods, as we shall see later.

Example 9.4 (Prediction using a naive Bayes model). Suppose our vocabulary contains three words a, b and c, and we use a multivariate Bernoulli model for our e-mails, with parameters

$$\boldsymbol{\theta}^{\oplus} = (0.5, 0.67, 0.33) \qquad \boldsymbol{\theta}^{\ominus} = (0.67, 0.33, 0.33)$$

This means, for example, that the presence of b is twice as likely in spam (+), compared with ham.

The e-mail to be classified contains words a and b but not c, and hence is described by the bit vector $\mathbf{x} = (1, 1, 0)$. We obtain likelihoods

$$P(\mathbf{x}|\oplus) = 0.5 \cdot 0.67 \cdot (1 - 0.33) = 0.222 \qquad P(\mathbf{x}|\ominus) = 0.67 \cdot 0.33 \cdot (1 - 0.33) = 0.148$$

The ML classification of $\mathbf{x}$ is thus spam. In the case of two classes it is often

convenient to work with likelihood ratios and odds. The likelihood ratio can be calculated as $\frac{P(\mathbf{x}|\oplus)}{P(\mathbf{x}|\ominus)} = \frac{0.5}{0.67}\frac{0.67}{0.33}\frac{1-0.33}{1-0.33} = 3/2 > 1$. This means that the MAP classification of **x** is also spam if the prior odds are more than 2/3, but ham if they are less than that. For example, with 33% spam and 67% ham the prior odds are $\frac{P(\oplus)}{P(\ominus)} = \frac{0.33}{0.67} = 1/2$, resulting in a posterior odds of $\frac{P(\oplus|\mathbf{x})}{P(\ominus|\mathbf{x})} = \frac{P(\mathbf{x}|\oplus)}{P(\mathbf{x}|\ominus)}\frac{P(\oplus)}{P(\ominus)} = 3/2 \cdot 1/2 = 3/4 < 1$. In this case the likelihood ratio for **x** is not strong enough to push the decision away from the prior.

Alternatively, we can employ a multinomial model. The parameters of a multinomial establish a distribution over the words in the vocabulary, say

$$\boldsymbol{\theta}^{\oplus} = (0.3, 0.5, 0.2) \qquad \boldsymbol{\theta}^{\ominus} = (0.6, 0.2, 0.2)$$

The e-mail to be classified contains three occurrences of word *a*, one single occurrence of word *b* and no occurrences of word *c*, and hence is described by the count vector **x** = (3, 1, 0). The total number of vocabulary word occurrences is $n = 4$. We obtain likelihoods

$$P(\mathbf{x}|\oplus) = 4!\frac{0.3^3}{3!}\frac{0.5^1}{1!}\frac{0.2^0}{0!} = 0.054 \qquad P(\mathbf{x}|\ominus) = 4!\frac{0.6^3}{3!}\frac{0.2^1}{1!}\frac{0.2^0}{0!} = 0.1728.$$

The likelihood ratio is $\left(\frac{0.3}{0.6}\right)^3\left(\frac{0.5}{0.2}\right)^1\left(\frac{0.2}{0.2}\right)^0 = 5/16$. The ML classification of **x** is thus ham, the opposite of the multivariate Bernoulli model. This is mainly because of the three occurrences of word *a*, which provide strong evidence for ham.

Notice how the likelihood ratio for the multivariate Bernoulli model is a product of factors $\theta_i^{\oplus}/\theta_i^{\ominus}$ if $x_i = 1$ in the bit vector to be classified, and $(1-\theta_i^{\oplus})/(1-\theta_i^{\ominus})$ if $x_i = 0$. For the multinomial model the factors are $(\theta_i^{\oplus}/\theta_i^{\ominus})^{x_i}$. One consequence of this is that the multinomial model only takes the presence of words into account, whereas in the multivariate Bernoulli model absent words can make a difference. In the previous example, not containing word *b* corresponds to a factor of (1 – 0.67)/(1 – 0.33) = 1/2 in the likelihood ratio. The other main difference between the two models is that multiple occurrences of words are treated like duplicated features in the multinomial model, through the exponential 'weight' x_i. This becomes clearer by taking the logarithm of the likelihood ratio, which is $\sum_i x_i(\ln\theta_i^{\oplus} - \ln\theta_i^{\ominus})$: this expression is linear in $\ln\theta_i^{\oplus}$ and

$\theta_i^{\ominus}$ with x_i as weights. Notice that this does not mean that naive Bayes classifiers are linear in the sense discussed in Chapter 7 unless we can demonstrate a linear relationship between ln θ and the corresponding feature value. But we can say that naive Bayes models are linear in a particular space (the 'log-odds' space) obtained by applying a well-defined transformation to the features. We will return to this point when we discuss ☞ *feature calibration* in Section 10.2.

The fact that the joint likelihood ratio of a naive Bayes model factorises as a product of likelihood ratios of individual words is a direct consequence of the naive Bayes assumption. In other words, the learning task decomposes into univariate tasks, one for each word in the vocabulary. We have encountered such a decomposition before when we discussed ☞ *multivariate linear regression* in Section 7.1. There, we saw an example of how ignoring feature correlation could be harmful. Can we come up with similar examples for naive Bayes classifiers? Consider the situation when a particular word occurs twice in the vocabulary. In that case, we have the same factor occurring twice in the product for the likelihood ratio, and are effectively giving the word in question twice the weight of other words. While this is an extreme example, such double-counting does have noticeable effects in practice. I previously gave the example that if a spam e-mail contains the word 'Viagra', it is also expected to contain the word 'pill', so seeing the two words together should not give much more evidence for spam than seeing the first word on its own, and the likelihood ratio for the two words should not be much higher than that of the first word. However, multiplying two likelihood ratios larger than 1 will result in an even larger likelihood ratio. As a result, the probability estimates of a naive Bayes classifier are often pushed too far towards 0 or 1.

This may not seem such a big deal if we are only interested in classification, and not in the probability estimates as such. However, *an often overlooked consequence of having uncalibrated probability estimates such as those produced by naive Bayes is that both the ML and MAP decision rules become inadequate*. Unless we have evidence that the model assumptions are satisfied, the only sensible thing to do in this case is to invoke the *recalibrated likelihood decision rule*, which requires one to learn a weight vector over the classes, in order to correct for the estimation errors in the likelihoods. Specifically, we want to find weights w_i such that predicting $\text{argmax}_y\, w_y P(X = x|Y = y)$ results in the smallest possible loss – e.g., the number of misclassified examples – over a test set. For two classes this can be solved by the same procedure we considered for

☞ *turning rankers into classifiers* in Section 2.2. To see this, notice that for two classes the recalibrated likelihood decision rule can be rewritten as

- ☞ predict positive if $w^{\oplus}\ P(X = x|Y = \oplus) > w^{\ominus}\ P(X = x|Y = \ominus)$ and negative otherwise; which is equivalent to
- ☞ predict positive if $P(X = x|Y = \oplus)/P(X = x|Y = \ominus) > w^{\ominus}\ /\ w^{\oplus}$ and negative otherwise

Figure 9.4. (left) ROC curves produced by two naive Bayes classifiers (solid line: a variant of the multivariate Bernoulli model; dashed line: a variant of the multinomial model). Both models have similar ranking performance and yield almost the same – more or less optimal – MAP decision threshold. (right) On a different data set from the same domain, the multinomial model's MAP threshold is slightly better, hinting at somewhat better calibrated probability estimates. But since the slope of the accuracy isometrics indicates that there are about four positives for every negative, the optimal decision rule is in fact to always predict positive.

This demonstrates that in the two-class case we really have just one degree of freedom, as multiplying the weights by a constant does not affect the decisions. In other words, what we are interested in is finding the best threshold $t = w^{\ominus}/w^{\oplus}$ on the likelihood ratio, which is essentially the same problem as finding the best operating point on an ROC curve. The solution is given by the point on the highest accuracy isometric. Figure 9.4 illustrates this on two real-life data sets: in the left figure we see that the MAP decision threshold is more or less optimal, whereas in the right figure the optimal point is in the top right-hand corner.

For more than two classes, finding a globally optimal weight vector is computationally intractable, which means that we need to resort to a heuristic method. In Section 3.1 such a method was demonstrated for three classes. The idea is to fix the weights one by one, using some ordering of the classes. That is,

we use the two-class procedure to optimally separate the i-th class from the previous $i - 1$ classes.

Training a naive Bayes model

Training a probabilistic model usually involves estimating the parameters of the distributions used in the model. The parameter of a Bernoulli distribution can be estimated by counting the number of successes d in n trials and setting $\hat{\theta} = d/n$. In other words, we count, for each class, how many e-mails contain the word in question. Such relative frequency estimates are usually smoothed by including *pseudo-counts*, representing the outcome of virtual trials according to some fixed distributions. In the case of a Bernoulli distribution the most common smoothing operation is the Laplace correction, which involves two virtual trials, one of which results in success and the other in failure. Consequently, the relative frequency estimate is changed to $(d + 1)/(n + 2)$. From a Bayesian perspective this amounts to adopting a uniform prior, representing our initial belief that success and failure are equally likely. If appropriate, we can strengthen the influence of the prior by including a larger number of virtual trials, which means that more data is needed to move the estimate away from the prior. For a categorical distribution smoothing adds one pseudo-count to each of the k categories, leading to the smoothed estimate $(d + 1)/(n + k)$. The *m-estimate* generalises this further by making both the total number of pseudo-counts m and the way they are distributed over the categories into parameters. The estimate for the i-th category is defined as $(d + p_i\, m)/(n + m)$, where p_i is a distribution over the categories (i.e., $\Sigma_{i=1}^{k} = p_i = 1$). Notice that smoothed relative frequency estimates – and hence products of such estimates – can never attain the extreme values $\hat{\theta} = 0$ or $\hat{\theta} = 1$.

Example 9.5 (Training a naive Bayes model). We now show how the parameter vectors in the previous example might have been obtained. Consider the following e-mails consisting of five words a, b, c, d, e :

e_1: $b\ d\ e\ b\ b\ d\ e$

e_2: $b\ c\ e\ b\ b\ d\ d\ e\ c\ c$

e_5: $a\ b\ a\ b\ a\ b\ a\ e\ d$

e_6: $a\ c\ a\ c\ a\ c\ a\ e\ d$

e_3: *a d a d e a e e* e_7: *e a e d a e a*

e_4: *b a d b e d a b* e_8: *d e d e d*

We are told that the e-mails on the left are spam and those on the right are ham, and so we use them as a small training set to train our Bayesian classifier. First, we decide that *d* and *e* are so-called *stop words* that are too common to convey class information. The remaining words, *a*, *b* and *c*, constitute our vocabulary.

For the multinomial model, we represent each e-mail as a count vector, as in Table 9.1 (left). In order to estimate the parameters of the multinomial, we sum up the count vectors for each class, which gives (5, 9, 3) for spam and (11, 3, 3) for ham. To smooth these probability estimates we add one pseudo-count for each vocabulary word, which brings the total number of occurrences of vocabulary words to 20 for each class. The estimated parameter vectors are thus $\hat{\boldsymbol{\theta}}^{\oplus}$ = (6/20, 10/20, 4/20) = (0.3, 0.5, 0.2) for spam and $\hat{\boldsymbol{\theta}}^{\ominus}$ = (12/20, 4/20, 4/20) = (0.6, 0.2, 0.2) for ham.

In the multivariate Bernoulli model e-mails are represented by bit vectors, as in Table 9.1 (right). Adding the bit vectors for each class results in (2, 3, 1) for spam and (3, 1, 1) for ham. Each count is to be divided by the number of documents in a class, in order to get an estimate of the probability of a document containing a particular vocabulary word. Probability smoothing now means adding two pseudo-documents, one containing each word and one containing none of them. This results in the estimated parameter vectors $\hat{\boldsymbol{\theta}}^{\oplus}$ = (3/6, 4/6, 2/6) = (0.5, 0.67, 0.33) for spam and $\hat{\boldsymbol{\theta}}^{\ominus}$ = (4/6, 2/6, 2/6) = (0.67, 0.33, 0.33) for ham.

E-mail	#*a*	#*b*	#*c*	Class
e_1	0	3	0	+
e_2	0	3	3	+
e_3	3	0	0	+
e_4	2	3	0	+
e_5	4	3	0	–
e_6	4	0	3	–
e_7	3	0	0	–
e_8	0	0	0	–

E-mail	*a*?	*b*?	*c*?	Class
e_1	0	1	0	+
e_2	0	1	1	+
e_3	1	0	0	+
e_4	1	1	0	+
e_5	1	1	0	–
e_6	1	0	1	–
e_7	1	0	0	–
e_8	0	0	0	–

Table 9.1. (left) A small e-mail data set described by count vectors. (right) The same data set described by bit vectors.

Many other variations of the naive Bayes classifier exist. In fact, what is normally understood as 'the' naive Bayes classifier employs neither a multinomial nor a multivariate Bernoulli model, but rather a multivariate categorical model. This means that features are categorical, and the probability of the i-th feature taking on its l-th value for class c examples is given by $\theta_{il}^{(c)}$, under the constraint that $\sum_{l=1}^{k_i} \theta_{il}^{(c)} = 1$, where k_i is the number of values of the i-th feature. These parameters can be estimated by smoothed relative frequencies in the training set, as in the multivariate Bernoulli case. We again have that the joint probability of the feature vector is the product of the individual feature probabilities, and hence $P(F_i, F_j|C) = P(F_i|C)P(F_j|C)$ for all pairs of features and for all classes.

Notice, by the way, that conditional independence is quite different from unconditional independence: neither implies the other. To see that conditional independence does not imply unconditional independence, imagine two words that are very likely to occur in spam, but they are independent (i.e., the probability of both of them occurring in a spam e-mail is the product of the marginal probabilities). Imagine further that they are very unlikely – but also independent – in ham. Suppose I tell you an unclassified e-mail contains one of the words: you would probably guess that it is a spam e-mail, from which you would further guess that it also contains the other word – demonstrating that the words are not unconditionally independent. To see that unconditional independence does not imply conditional independence, consider two different independent words, and let an e-mail be spam if it contains at least one of the words and ham otherwise, then among spam e-mails the two words are dependent (since if I know that a spam e-mail doesn't contain one of the words, then it must contain the other).

Another extension of the naive Bayes model is required when some of the features are real-valued. One option is to discretise the real-valued features in a pre-processing stage: this will be discussed in Chapter 10. Another option is to assume that the feature values are normally distributed within each class, as discussed in the previous section. In this context it is worth noting that the naive Bayes assumption boils down to assuming a diagonal covariance matrix within each class, so that each feature can be treated independently. A third option that

is also used in practice is to model the class-conditional likelihood of each feature by a non-parametric density estimator. These three options are illustrated in Figure 9.5.

In summary, the naive Bayes model is a popular model for dealing with textual, categorical and mixed categorical/real-valued data. Its main shortcoming as a probabilistic model – poorly calibrated probability estimates – are outweighed by generally good ranking performance. Another apparent paradox with naive Bayes is that it isn't particularly Bayesian at all! For one thing, we have seen that the poor probability estimates necessitate the use of reweighted likelihoods, which avoids using Bayes' rule altogether. Secondly, in training a naive Bayes model we use maximum-likelihood parameter estimation, whereas a fully fledged Bayesian approach would not commit to a particular parameter value, but rather employ a full posterior distribution. Personally, I think the essence of naive Bayes is the decomposition of joint likelihoods into marginal likelihoods. This decomposition is evocatively visualised by the Scottish tartan pattern in Figure 1.3 on p.31, which is why I like to call naive Bayes the 'Scottish classifier'.

Figure 9.5. (left) Examples of three density estimators on 20 points sampled from a normal distribution with zero mean and unit variance (dotted line)). A histogram is a simple non-parametric method which employs a fixed number of equal-width intervals. A kernel density estimator (in red) applies interpolation to obtain a smooth density function. The solid bell curve (in blue) is obtained by estimating the sample mean and variance, assuming the true distribution is normal. **(right)** Here, the 20 points are sampled uniformly from [–2, 2], and the non-parametric methods generally do better.

9.3 Discriminative learning by optimising conditional likelihood

In the introduction to this chapter we distinguished between generative and discriminative probabilistic models. Naive Bayes models are generative: after training they can be used to generate data. In this section we look at one of the most commonly used discriminative models: *logistic regression*.[5] The easiest way to understand logistic regression is as a linear classifier whose probability estimates have been logistically calibrated using the method described in Section 7.4, but with one crucial difference: calibration is an integral part of the training algorithm, rather than a post-processing step. While in generative models the decision boundary is a by-product of modelling the distributions of each class, logistic regression models the decision boundary directly. For example, if the classes are overlapping then logistic regression will tend to locate the decision boundary in an area where classes are maximally overlapping, regardless of the 'shapes' of the samples of each class. This results in decision boundaries that are noticeably different from those learned by generative classifiers (Figure 9.6).

Equation 7.13 on p.222 expresses the likelihood ratio as $\exp(\gamma(d(\mathbf{x}) - d_0))$ with $d(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} - t$. Since we are learning the parameters all at once in discriminative learning, we can absorb γ and d_0 into $\mathbf{w}$ and t. So the logistic regression model is simply given by

$$\hat{p}(\mathbf{x}) = \frac{\exp(\mathbf{w}\cdot\mathbf{x} - t)}{\exp(\mathbf{w}\cdot\mathbf{x} - t) + 1} = \frac{1}{1 + \exp(-(\mathbf{w}\cdot\mathbf{x} - t))}$$

Figure 9.6. (left) On this data set, logistic regression (in blue) outperforms the basic linear classifier (in red) and the least squares classifier (in orange) because the latter two are more sensitive to the shape of the classes, while logistic regression concentrates on where the classes overlap. **(right)** On this slightly different set of points, logistic regression is outperformed by the other two methods because it concentrates too much on tracking the transition from mostly positive to mostly negative.

Assuming the class labels are $y = 1$ for positives and $y = 0$ for negatives, this

defines a Bernoulli distribution for each training example:

$$P(y_l|\mathbf{x}_l) = \hat{p}(\mathbf{x}_l)^{y_l}(1 - \hat{p}(\mathbf{x}_l))^{(1-y_l)}$$

It is important to note that the parameters of these Bernoulli distributions are linked through **w** and *t*, and consequently there is one parameter for every feature dimension, rather than for every training instance.

The likelihood function is

$$\mathrm{CL}(\mathbf{w}, t) = \prod_i P(y_i|\mathbf{x}_i) = \prod_i \hat{p}(\mathbf{x}_i)^{y_i}(1-\hat{p}(\mathbf{x}_i))^{(1-y_i)}$$

This is called *conditional likelihood* to stress that it gives us the *conditional* probability $P(y_i|\mathbf{x}_i)$ rather than $P(\mathbf{x}_i)$ as in a generative model. Notice that our use of the product requires the assumption that the y-values are independent given $\mathbf{x}$; but this is an entirely reasonable assumption and not nearly as strong as the naive Bayes assumption of $\mathbf{x}$ being independent within each class. As usual, the logarithm of the likelihood function is easier to work with:

$$\mathrm{LCL}(\mathbf{w}, t) = \sum_i y_i \ln \hat{p}(\mathbf{x}_i) + (1-y_i)\ln(1-\hat{p}(\mathbf{x}_i)) = \sum_{\mathbf{x}^{\oplus} \in Tr^{\oplus}} \ln \hat{p}(\mathbf{x}^{\oplus}) + \sum_{\mathbf{x}^{\ominus} \in Tr^{\ominus}} \ln(1-\hat{p}(\mathbf{x}^{\ominus}))$$

We want to maximise the log-conditional likelihood with respect to these parameters, which means that all partial derivatives must be zero:

$$\nabla_{\mathbf{w}} \mathrm{LCL}(\mathbf{w}, t) = \mathbf{0}$$
$$\frac{\partial}{\partial t} \mathrm{LCL}(\mathbf{w}, t) = 0$$

Although these equations do not yield an analytic solution, they can be used to obtain further insight into the nature of logistic regression. Concentrating on t, we first need to do some algebraic groundwork.

$$\begin{aligned}
\ln \hat{p}(\mathbf{x}) &= \ln \frac{\exp(\mathbf{w}\cdot\mathbf{x} - t)}{\exp(\mathbf{w}\cdot\mathbf{x} - t) + 1} \\
&= \mathbf{w}\cdot\mathbf{x} - t - \ln(\exp(\mathbf{w}\cdot\mathbf{x} - t) + 1) \\
\frac{\partial}{\partial t} \ln \hat{p}(\mathbf{x}) &= -1 - \frac{\partial}{\partial t} \ln(\exp(\mathbf{w}\cdot\mathbf{x} - t) + 1) \\
&= -1 - \frac{1}{\exp(\mathbf{w}\cdot\mathbf{x} - t) + 1} \exp(\mathbf{w}\cdot\mathbf{x} - t)\cdot(-1) \\
&= \hat{p}(\mathbf{x}) - 1
\end{aligned}$$

Similarly for the negatives:

$$
\begin{aligned}
\ln(1-\hat{p}(\mathbf{x})) &= \ln \frac{1}{\exp(\mathbf{w}\cdot\mathbf{x}-t)+1} \\
&= -\ln(\exp(\mathbf{w}\cdot\mathbf{x}-t)+1) \\
\frac{\partial}{\partial t}\ln(1-\hat{p}(\mathbf{x})) &= \frac{\partial}{\partial t} -\ln(\exp(\mathbf{w}\cdot\mathbf{x}-t)+1) \\
&= \frac{-1}{\exp(\mathbf{w}\cdot\mathbf{x}-t)+1}\exp(\mathbf{w}\cdot\mathbf{x}-t)\cdot(-1) \\
&= \hat{p}(\mathbf{x})
\end{aligned}
$$

It follows that the partial derivative of LCL with respect to t has a simple form:

$$
\begin{aligned}
\frac{\partial}{\partial t}\mathrm{LCL}(\mathbf{w},t) &= \sum_{\mathbf{x}^{\oplus}\in Tr^{\oplus}}(\hat{p}(\mathbf{x})-1) + \sum_{\mathbf{x}^{\ominus}\in Tr^{\ominus}}\hat{p}(\mathbf{x}^{\ominus}) \\
&= \sum_{\mathbf{x}_i\in Tr}(\hat{p}(\mathbf{x}_i)-y_i)
\end{aligned}
$$

For the optimal solution this partial derivative is zero. What this means is that, on average, the predicted probability should be equal to the proportion of positives *pos*. This is a satisfying result, as it is clearly a desirable global property of a calibrated classifier. Notice that grouping models such as probability estimating trees have this property by construction, as they set the predicted probability equal to the empirical probability in a segment.

A very similar derivation leads to the partial derivative of the log-conditional likelihood with respect to the j-th weight w_j. The point to note here is that, where as $\frac{\partial}{\partial t}(\mathbf{w}\ .\ \mathbf{x} - t) = -1$, we have $\frac{\partial}{\partial w_j}(\mathbf{w}\ .\ \mathbf{x} - t) = \frac{\partial}{\partial w_j}(\Sigma_j\ w_j\ x_j - t) = x_j$, the instance's j-th feature value. This then leads to

$$
\frac{\partial}{\partial w_j}\mathrm{LCL}(\mathbf{w},t) = \sum_{\mathbf{x}_i\in Tr}(y_i-\hat{p}(\mathbf{x}_i))x_{ij} \tag{9.5}
$$

Setting this partial derivative to zero expresses another, feature-wise calibration property. For example, if the j-th feature is a sparse Boolean feature that is mostly zero, then this calibration property only involves the instances $\mathbf{x}_i$ for which $x_{ij} = 1$: on average, those instances should have their predicted probability equal the proportion of positives among them.

Example 9.6 (Univariate logistic regression). Consider the data in Figure 9.7 with 20 points in each class. Although both classes were generated from normal distributions, class overlap in this particular sample is less than what

could be expected on the basis of the class means. Logistic regression is able to take advantage of this and gives a much steeper sigmoid than the basic linear classifier with logistic calibration (explained in Example 7.7 on p.222), which is entirely formulated in terms of class means and variance. Also shown are the probability estimates obtained from the convex hull of the ROC curve (see Figure 7.13 on p.224);this calibration procedure is non-parametric and hence better able to detect the limited class overlap.

In terms of statistics, logistic regression has better mean squared error (0.040) than the logistically calibrated classifier (0.057). Isotonic calibration leads to the lowest error (0.021), but note that no probability smoothing has been applied to mitigate the risk of overfitting. The sum of predicted probabilities is 18.7 for the logistically calibrated classifier and 20 for the other two – i.e., equal to the number of examples, which is a necessary condition for full calibration. Finally, $\sum_{\mathbf{x}_i \in Tr}(y_i - \hat{p}(\mathbf{x}_i))x_i$ is 2.6 for the logistically calibrated classifier, 4.7 for the ROC-calibrated classifier, and 0 for logistic regression as expected from Equation 9.5.

In order to train a logistic regression model we need to find

$$\mathbf{w}^*, t^* = \underset{\mathbf{w},t}{\arg\max}\,\mathrm{CL}(\mathbf{w}, t) = \underset{\mathbf{w},t}{\arg\max}\,\mathrm{LCL}(\mathbf{w}, t)$$

Figure 9.7. Logistic regression (in red) compared with probability estimates obtained by logistic calibration (in blue) and isotonic calibration (in green); the latter two are applied to the basic linear classifier (estimated class means are indicated by circles). The corresponding three decision boundaries are shown as vertical dotted lines.

This can be shown to be a convex optimisation problem, which means that there is only one maximum. A range of optimisation techniques can be applied. One simple approach is inspired by the perceptron algorithm and iterates over examples, using the following update rule:

$$\mathbf{w} = \mathbf{w} + \eta(y_i - \hat{p}_i)\mathbf{x}_i$$

where η is the learning rate. Notice the relationship with the partial derivative in Equation 9.5. Essentially, we are using single examples to approximate the direction of steepest ascent.

9.4 Probabilistic models with hidden variables

Suppose you are dealing with a four-class classification problem with classes A, B, C and D. If you have a sufficiently large and representative training sample of size n, you can use the relative frequencies in the sample n_A, …, n_D to estimate the class prior $\hat{p}_A = n_A/n, \ldots, \hat{p}_D = n_D/n$, as we have done many times before.[6] Conversely, if you know the prior and want to know the most likely class distribution in a random sample of n instances, you would use the prior to calculate expected values $\mathbb{E}[n_A] = P_A \cdot n, \ldots, \mathbb{E}[n_D] = p_D \cdot n$. So, complete knowledge of one allows us to estimate or infer the other. However, sometimes we have a bit of knowledge about both. For example, we may know that $P_A = 1/2$ and that C is twice as likely as B, without knowing the complete prior. And we may know that the sample we saw last week was evenly split between $A \cup B$ and $C \cup D$, and that C and D were equally large, but we can't remember the size of A and B separately. What should we do?

Formalising what we know about the prior, we have $p_A = 1/2$; $p_B = \beta$, as yet unknown; $pc = 2\beta$, since it is twice p_B; and $p_D = 1/2 - 3\beta$, since the four cases need to add up to 1. Furthermore: $n_A + n_B = a + b = s$, $n_c = c$ and $n_D = d$, with s, c and d known. We want to infer a, b and β: however, it seems we are stuck in a chicken-and-egg problem. If we knew β we would have full knowledge about the prior and we could use that to infer expected values for a and b:

$$\frac{\mathbb{E}[a]}{\mathbb{E}[b]} = \frac{1/2}{\beta} \qquad \mathbb{E}[a] + \mathbb{E}[b] = s$$

from which we could derive

$$\mathbb{E}[a] = \frac{1}{1+2\beta}s \qquad \mathbb{E}[b] = \frac{2\beta}{1+2\beta}s \tag{9.6}$$

So, for example, if $s = 20$ and $\beta = 1/10$, then $\mathbb{E}[a] = 16\frac{2}{3}$ and $\mathbb{E}[b] = 3\frac{1}{3}$.

Conversely, if we knew a and b, then we could estimate β by maximum-likelihood estimation, using a multinomial distribution for a, b, c and d:

$$P(a,b,c,d|\beta) = K(1/2)^a \beta^b (2\beta)^c (1/2-3\beta)^d$$
$$\ln P(a,b,c,d|\beta) = \ln K + a\ln(1/2) + b\ln\beta + c\ln(2\beta) + d\ln(1/2-3\beta)$$

Here, K is a combinatorial constant that doesn't affect the value of β which maximises the likelihood. Taking the partial derivative with respect to β gives

$$\frac{\partial}{\partial\beta}\ln P(a,b,c,d|\beta) = \frac{b}{\beta} + \frac{2c}{2\beta} - \frac{3d}{1/2-3\beta}$$

Setting to 0 and solving for β finally gives

$$\hat{\beta} = \frac{b+c}{6(b+c+d)} \tag{9.7}$$

So, for example, if $b = 5$ and $c = d = 10$, then $\hat{\beta} = 1/10$.

The way out of this chicken-and-egg problem is to iterate the following two steps: (*i*) calculate an expected value of the missing frequencies a and b from an assumed or previously estimated value of the parameter β; and (*ii*) calculate a maximum-likelihood estimate of the parameter β from assumed or expected values of the missing frequencies a and b. These two steps are iterated until a stationary configuration is reached. So, if we start with $a = 15$, $b = 5$ and $c = d = 10$, then we have just seen that $\hat{\beta} = 1/10$. Plugging this value of β into Equation 9.6 gives us $\mathbb{E}[a] = 16\frac{2}{3}$ and $\mathbb{E}[b] = 3\frac{1}{3}$. Plugging these values back into Equation 9.7 yields $\hat{\beta} = 2/21$, which in turn gives $\mathbb{E}[a] = 16.8$ and $\mathbb{E}[b] = 3.2$, and so on. A stationary configuration with $\beta = 0.0948$, $a = 16.813$ and $b = 3.187$ is reached in fewer than 10 iterations. In this simple case this is a global optimum that is reached regardless of the starting point, essentially because the relationship between b and β is monotonic ($\mathbb{E}[b]$ increases with β according to Equation 9.6 and $\hat{\beta}$ increases with b according to Equation 9.7). However, this is not normally the case: we will return to this point later.

Expectation-Maximisation

The problem that we have just discussed is an example of a problem with missing data, where the full data Y separates into observed variables X and *hidden variables* Z (also called *latent variables*). In the example, the observed variables are c, d and s, and the hidden variables are a and b. We also have model parameter(s) θ, which is β in the example.[7] Denote the estimate of θ in the t-th iteration as θ^t. We have two relevant quantities:

- the expectation $\mathbb{E}[Z|X, \theta^t]$ of the hidden variables given the observed variables and the current estimate of the parameters (so in Equation 9.6 the expectations of a and b depend on s and β);

- the likelihood $P(Y|\theta)$, which is used to find the maximising value of θ.

In the likelihood function we need values for $Y = X \cup Z$. We obviously use the observed values for X, but we need to use previously calculated expectations for Z. This means that we really want to maximise $P(X \cup \mathbb{E}[Z|X, \theta^t]|\theta)$, or equivalently, the logarithm of that function. We now make the assumption that the logarithm of the likelihood function is linear in Y : notice that this assumption is valid in the example above. For any linear function f, $f(\mathbb{E}[Z]) = \mathbb{E}[f(Z)]$ and thus we can bring the expectation outside in our objective function:

$$\ln P(X \cup \mathbb{E}[Z|X,\theta^t]|\theta) = \mathbb{E}[\ln P(X \cup Z|\theta)|X,\theta^t] = \mathbb{E}[\ln P(Y|\theta)|X,\theta^t] \tag{9.8}$$

This last expression is usually denoted as $Q(\theta|\theta^t)$, as it essentially tells us how to calculate the next value of θ from the current one:

$$\theta^{t+1} = \arg\max_{\theta} Q(\theta|\theta^t) = \arg\max_{\theta} \mathbb{E}[\ln P(Y|\theta)|X,\theta^t] \tag{9.9}$$

This, then, is the general form of the celebrated *Expectation-Maximisation* (*EM*) algorithm, which is a powerful approach to probabilistic modelling with hidden variables or missing data. Similar to the example above, we iterate over assigning an expected value to the hidden variables given our current estimates of the parameters, and re-estimating the parameters from these updated expectations, until a stationary configuration is reached. We can start the iteration by initialising either the parameters or the hidden variables in some way. The algorithm bears a striking resemblance to the *K-means* algorithm (Algorithm 8.1 on p.248), which also iterates over assigning data points to current cluster means, and re-estimating the cluster means from the new assignments. This resemblance is not accidental, as we shall see in a moment.

Like the K-means algorithm, EM can be proved to always converge to a stationary configuration for a wide class of probabilistic models. However, EM can get trapped in a local optimum that is dependent on the initial configuration.

Gaussian mixture models

A common application of Expectation-Maximisation is to estimate the parameters of a *Gaussian mixture model* from data. In such a model the data points are generated by K normal distributions, each with their own mean $\boldsymbol{\mu}_j$ and covariance matrix $\boldsymbol{\Sigma}_j$, and the proportion of points coming from each Gaussian is governed by a prior $\tau = (\tau_1,\ldots,\tau_K)$. If each data point in a sample were labelled with the index of the Gaussian it came from this would be a straightforward classification problem, which could be solved easily by estimating each Gaussian's $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ separately from the data points belonging to class j. However, we are now considering the much harder predictive clustering problem in which the class labels are hidden and need to be reconstructed from the observed feature values.

A convenient way to model this is to have for each data point $\mathbf{x}_i$ a Boolean vector $\mathbf{z}_i = (z_{i1}, \ldots, z_{iK})$ such that exactly one bit z_{ij} is set to 1 and the rest set to 0, signalling that the i-th data point comes from the j-th Gaussian. Using this notation we can adapt the expression for the ☞ *multivariate normal distribution* (Equation 9.2 on p.267) to obtain a general expression for a Gaussian mixture model:

$$P(\mathbf{x}_l, \mathbf{z}_l|\theta) = \sum_{j=1}^{K} z_{lj}\tau_j \frac{1}{(2\pi)^{d/2}\sqrt{|\Sigma_j|}} \exp\left(-\frac{1}{2}(\mathbf{x}_l - \boldsymbol{\mu}_j)^{\mathrm{T}}\Sigma_j^{-1}(\mathbf{x}_l - \boldsymbol{\mu}_j)\right) \tag{9.10}$$

Here, θ collects all the parameters τ, $\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_1,\ldots,\boldsymbol{\Sigma}k$. The interpretation as a generative model is as follows: we first randomly select a Gaussian using the prior $\boldsymbol{\tau}$, and then we invoke the corresponding Gaussian using the indicator variables z_{ij}.

In order to apply Expectation-Maximisation we form the Q function:

$$
\begin{aligned}
Q(\theta|\theta^t) &= \mathbb{E}\left[\ln P(\mathbf{X}\cup\mathbf{Z}|\theta)\,|\,\mathbf{X},\theta^t\right] \\
&= \mathbb{E}\left[\ln\prod_{i=1}^{n} P(\mathbf{x}_i\cup\mathbf{z}_i|\theta)\,\middle|\,\mathbf{X},\theta^t\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{n}\ln P(\mathbf{x}_i\cup\mathbf{z}_i|\theta)\,\middle|\,\mathbf{X},\theta^t\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{n}\ln\sum_{j=1}^{K} z_{ij}\tau_j\frac{1}{(2\pi)^{d/2}\sqrt{|\Sigma_j|}}\exp\left(-\frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu}_j)^{\mathrm{T}}\Sigma_j^{-1}(\mathbf{x}_i-\boldsymbol{\mu}_j)\right)\,\middle|\,\mathbf{X},\theta^t\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{n}\sum_{j=1}^{K} z_{ij}\ln\left(\tau_j\frac{1}{(2\pi)^{d/2}\sqrt{|\Sigma_j|}}\exp\left(-\frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu}_j)^{\mathrm{T}}\Sigma_j^{-1}(\mathbf{x}_i-\boldsymbol{\mu}_j)\right)\right)\,\middle|\,\mathbf{X},\theta^t\right] \qquad (*) \\
&= \mathbb{E}\left[\sum_{i=1}^{n}\sum_{j=1}^{K} z_{ij}\left(\ln\tau_j-\frac{d}{2}\ln(2\pi)-\frac{1}{2}\ln|\Sigma_j|-\frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu}_j)^{\mathrm{T}}\Sigma_j^{-1}(\mathbf{x}_i-\boldsymbol{\mu}_j)\right)\,\middle|\,\mathbf{X},\theta^t\right] \\
&= \sum_{i=1}^{n}\sum_{j=1}^{K}\mathbb{E}\left[z_{ij}|\mathbf{X},\theta^t\right]\left(\ln\tau_j-\frac{d}{2}\ln(2\pi)-\frac{1}{2}\ln|\Sigma_j|-\frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu}_j)^{\mathrm{T}}\Sigma_j^{-1}(\mathbf{x}_i-\boldsymbol{\mu}_j)\right)
\end{aligned}
\tag{9.11}
$$

The step marked (*) is possible because for a given i only one z_{ij} is switched on, hence we can bring the indicator variables outside the logarithm. The last line shows the Q function in the desired form, involving on the one hand expectations over the hidden variables conditioned on the observable data $\mathbf{X}$ and the previously estimated parameters θ^t, and on the other hand expressions in θ that allow us to find θ^{t+1} by maximisation.

The Expectation step of the EM algorithm is thus the calculation of the expected values of the indicator variables $\mathbb{E}[z_{ij}|\mathbf{X},\theta^t]$. Notice that expectations of Boolean variables take values on the entire interval [0, 1], under the constraint that $\sum_{j=1}^{K} z_{ij} = 1$ for all i. In effect, the hard cluster assignment of K-means is changed into a soft assignment – one of the ways in which Gaussian mixture models generalise K-means. Now, suppose that $K = 2$ and we expect both clusters to be of equal size and with equal covariances. If a given data point $\mathbf{x}_i$ is equidistant from the two cluster means (or rather, our current estimates of these), then clearly $\mathbb{E}[z_{i1}|\mathbf{X}, \theta^t = \mathbb{E}[z_{i2}|\mathbf{X}, \theta^t] = 1/2$. In the general case these expectations are apportioned proportionally to the probability mass assigned to the point by each Gaussian:

$$
\mathbb{E}\left[z_{ij}|\mathbf{X},\theta^t\right] = \frac{\tau_j^t f(\mathbf{x}_i|\boldsymbol{\mu}_j^t,\Sigma_j^t)}{\sum_{k=1}^{K}\tau_k^t f(\mathbf{x}_i|\boldsymbol{\mu}_k^t,\Sigma_k^t)} \tag{9.12}
$$

where $f(\mathrm{x}|\boldsymbol{\mu}, \Sigma)$ stands for the multivariate Gaussian density function.

For the Maximisation step we optimise the parameters in Equation 9.11. Notice

there is no interaction between the terms containing τ_j and the terms containing the other parameters, and so the prior distribution τ can be optimised separately:

$$\begin{aligned}\boldsymbol{\tau}^{t+1} &= \underset{\boldsymbol{\tau}}{\operatorname{argmax}} \sum_{i=1}^{n} \sum_{j=1}^{K} \mathbb{E}\left[z_{ij} \middle| \mathbf{X}, \theta^t\right] \ln \tau_j \\ &= \underset{\boldsymbol{\tau}}{\operatorname{argmax}} \sum_{j=1}^{K} E_j \ln \tau_j \qquad \text{under the constraint } \sum_{j=1}^{K} \tau_j = 1\end{aligned}$$

where I have written E_j for $\sum_{i=1}^{n} \mathbb{E}\left[z_{ij} \middle| \mathbf{X}, \theta^t\right]$, which is the total (partial) membership of the j-th cluster – notice that $\sum_{j=1}^{K} E_j = n$. For simplicity we assume $K = 2$, so that $\tau_2 = 1 - \tau_1$: then

$$\tau_1^{t+1} = \underset{\tau_1}{\operatorname{argmax}}\, E_1 \ln \tau_1 + E_2 \ln(1 - \tau_1)$$

Setting the derivative with respect to τ_1 to zero and solving for τ_1, it can be easily verified that $\tau_1^{t+1} = E_1/(E_1 + E_2) = E_1/n$ and thus $\tau_2^{t+1} = E_2/n$. In the general case of K clusters we have analogously

$$\tau_j^{t+1} = \frac{E_j}{\sum_{k=1}^{K} E_k} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}\left[z_{ij} \middle| \mathbf{X}, \theta^t\right] \tag{9.13}$$

The means and covariance matrices can be optimised for each cluster separately:

$$\begin{aligned}\boldsymbol{\mu}_j^{t+1}, \boldsymbol{\Sigma}_j^{t+1} &= \underset{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j}{\operatorname{argmax}} \sum_{i=1}^{n} \mathbb{E}\left[z_{ij} \middle| \mathbf{X}, \theta^t\right] \left(-\frac{1}{2} \ln |\boldsymbol{\Sigma}_j| - \frac{1}{2} (\mathbf{x}_i - \boldsymbol{\mu}_j)^{\mathrm{T}} \boldsymbol{\Sigma}_j^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_j)\right) \\ &= \underset{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j}{\operatorname{argmin}} \sum_{i=1}^{n} \mathbb{E}\left[z_{ij} \middle| \mathbf{X}, \theta^t\right] \left(\frac{1}{2} \ln |\boldsymbol{\Sigma}_j| + \frac{1}{2} (\mathbf{x}_i - \boldsymbol{\mu}_j)^{\mathrm{T}} \boldsymbol{\Sigma}_j^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_j)\right)\end{aligned}$$

Notice that the term between brackets is a squared-distance term with the expectations functioning as instance weights on each instance. This describes a generalised version of the problem of finding the point that ☞ *minimises the sum of squared Euclidean distances* to a set of points (Theorem 8.1 on p.238). While that problem is solved by the arithmetic mean, here we simply take the *weighted* average over all the points:

$$\boldsymbol{\mu}_j^{t+1} = \frac{1}{E_j} \sum_{i=1}^{n} \mathbb{E}\left[z_{ij} \middle| \mathbf{X}, \theta^t\right] \mathbf{x}_i = \frac{\sum_{i=1}^{n} \mathbb{E}\left[z_{ij} \middle| \mathbf{X}, \theta^t\right] \mathbf{x}_i}{\sum_{i=1}^{n} \mathbb{E}\left[z_{ij} \middle| \mathbf{X}, \theta^t\right]} \tag{9.14}$$

Similarly, the covariance matrix is computed as a weighted average of

covariance matrices obtained from each data point, taking into account the newly estimated mean:

$$\begin{aligned}\boldsymbol{\Sigma}_j^{t+1} &= \frac{1}{E_j}\sum_{l=1}^{n}\mathbb{E}\left[z_{lj}\middle|\mathbf{X},\theta^t\right](\mathbf{x}_l-\boldsymbol{\mu}_j^{t+1})(\mathbf{x}_l-\boldsymbol{\mu}_j^{t+1})^{\mathrm{T}}\\ &= \frac{\sum_{l=1}^{n}\mathbb{E}\left[z_{lj}\middle|\mathbf{X},\theta^t\right](\mathbf{x}_l-\boldsymbol{\mu}_j^{t+1})(\mathbf{x}_l-\boldsymbol{\mu}_j^{t+1})^{\mathrm{T}}}{\sum_{l=1}^{n}\mathbb{E}\left[z_{lj}\middle|\mathbf{X},\theta^t\right]}\end{aligned} \tag{9.15}$$

Equations 9.12–9.15, then, constitute the EM solution to learning a Gaussian mixture model from an unlabelled sample. I have presented it here in its most general form, explicitly modelling unequal cluster sizes and covariance matrices. The latter is important as it allows for clusters of different shapes, unlike the *K*-means algorithm which assumes that all clusters have the same spherical shape. Consequently, the boundaries between clusters will not be linear, as they are in the clusterings learned by *K*-means. Figure 9.8 demonstrates the convergence of EM on a simple univariate data set, as well as the existence of multiple stationary configurations.

Figure 9.8. (left) The blue line shows the true Gaussian mixture model from which the 10 points on the *x*-axis were sampled; the colour of the points indicates whether they came from the left or the right Gaussian. The other lines show convergence of Expectation-Maximisation to a stationary configuration from a random initialisation. (**right**) This plot shows four stationary configurations for the same set of points. The EM algorithm was run for 20 iterations; the thickness of one of the lines demonstrates that this configuration takes longer to converge.

In conclusion, Expectation-Maximisation is a versatile and powerful method to deal with missing variables in a principled way. As we have seen in detail for the Gaussian mixture model, the main ingredient is an expression for the parametric likelihood function $P(X \cup Z|\theta)$, from which the update equations can be derived by means of the Q function. A word of caution is also in order, since – except in the simplest cases – there will be more than one stationary configuration. Like

with K-means, the optimisation should therefore be run multiple times with different starting configurations.

9.5 Compression-based models

We end this chapter with a brief discussion of an approach to machine learning that is both closely related to and quite distinct from the probabilistic approach. Consider the maximum a posteriori decision rule again:

$$y_{\text{MAP}} = \arg\max_y P(X = x|Y = y)P(Y = y)$$

Y	$P(\text{Viagra} = 1\|Y)$	$IC(\text{Viagra} = 1\|Y)$	$P(\text{Viagra} = 0\|Y)$	$IC(\text{Viagra} = 0\|Y)$
spam	0.40	**1.32 bits**	0.60	**0.74 bits**
ham	0.12	**3.06 bits**	0.88	**0.18 bits**

Table 9.2. Example marginal likelihoods.

Taking negative logarithms, we can turn this into an equivalent minimisation:

$$y_{\text{MAP}} = \arg\min_y -\log P(X = x|Y = y) - \log P(Y = y) \tag{9.16}$$

This follows because for any two probabilities $0 < p < p' < 1$ we have $\infty > -\log p > -\log p' > 0$. If an event has probability p of happening, the negative logarithm of p quantifies the *information content* of the message that the event has indeed happened. This makes intuitive sense, as the less expected an event is, the more information an announcement of the event contains. The unit of information depends on the base of the logarithm: it is customary to take logarithms to the base 2, in which case information is measured in bits. For example, if you toss a fair coin once and tell me it came up heads, this contains $-\log_2 1/2 = 1$ bit of information; if you roll a fair die once and let me know it came up six, the information content of your message is $-\log_2 1/6 = 2.6$ bits. Equation 9.16 tells us that the MAP decision rule chooses the least surprising or the most expected class for an instance x given particular prior distributions and likelihoods. We write $IC(X|Y) = -\log_2 P(X|Y)$ and $IC(Y) = -\log_2 P(Y)$.

Example 9.7 (Information-based classification). Table 9.2 reproduces the

left table in Table 1.3 on p.29 together with the relevant information content quantities. If Y is uniformly distributed then $IC(Y = \text{spam}) = 1$ bit and $IC(Y = \text{ham}) = 1$ bit. It follows that

$$\arg\min_y \left(IC(\text{Viagra} = 1 | Y = y) + IC(Y = y)\right) = \text{spam}$$

$$\arg\min_y \left(IC(\text{Viagra} = 0 | Y = y) + IC(Y = y)\right) = \text{ham}$$

If ham is four times as likely as spam then $IC(Y = \text{spam}) = 2.32$ bit and $IC(Y = \text{ham}) = 0.32$ bit, and $\text{argmin}_y\ (IC(\text{Viagra} = 1|Y = y) + IC(Y = y)) = \text{ham}$.

Clearly, for a uniform distribution over k outcomes, each outcome has the same information content $-\log_2 1/k = \log_2 k$. For a non-uniform distribution these information contents differ, and hence it makes sense to compute the average information content or *entropy* $\sum_{i=1}^{k} -p_i \log_2 p_i$. We have encountered entropy before as an *impurity measure* in Section 5.1.

So far I have not really told you anything new, other than that there is a one-to-one relationship between probability and information content. What really kicks things off in compression-based learning is a fundamental result from information theory proved by Claude Shannon in 1948. Shannon's result says – loosely speaking – that we cannot transmit information at a rate that surpasses entropy, but we can get arbitrarily close to the optimal rate by designing clever binary codes. Some well-known codes include the Shannon–Fanon code and the Huffman code, which are worth looking up as they employ a simple tree structure to build the code from empirical probabilities. Even more efficient codes, such as arithmetic coding, combine multiple messages into a single code word.

Assuming the availability of a near-optimal code, we can now turn the tables and use information content – or 'description length' as it is more commonly called – as a proxy for probability. One simplified version of the minimum description length (MDL) principle runs as follows.

Definition 9.1 (Minimum description length principle). *Let $L(m)$ denote the length in bits of a description of model m, and let $L(D|m)$ denote the length in bits of a description of data D given model m. According to the minimum description length principle, the preferred model is the one minimising the description length of model and data given model:*

$$m_{\mathrm{MDL}} = \arg\min_{m \in M} \left(L(m) + L(D|m) \right) \tag{9.17}$$

In a predictive learning context, 'description of data given model' refers to whatever information we need, in addition to the model and the feature values of the data, to infer the target labels. If the model is 100% accurate no further information is needed, so this term essentially quantifies the extent to which the model is incorrect. For example, in a uniform two-class setting we need one bit for every data point incorrectly classified by the model. The term $L(m)$ quantifies the complexity of the model. For instance, if we are fitting a polynomial to the data we need to encode the degree of the polynomial as well as its roots, up to a certain resolution. MDL learning thus trades off accuracy and complexity of a model: the complexity term serves to avoid overfitting in a similar way to the *regularisation* term in ridge regression in Section 7.1 and the *slack variable* term in soft-margin SVMs in Section 7.3.

What encoding to use in order to determine the model complexity $L(m)$ is often not straightforward and to some extent subjective. This is similar to the Bayesian perspective, where we need to define a prior distribution on models. The MDL viewpoint offers a concrete way of defining model priors by means of codes.

9.6 Probabilistic models: Summary and further reading

In this chapter we covered a range of machine learning models that are all based on the idea that features and target variables can be modelled as random variables, giving the opportunity to explicitly represent and manipulate the level of certainty we have about those variables. Such models are usually predictive in that they result in a conditional distribution $P(Y|X)$ with which Y can be predicted from X. Generative models estimate the joint distribution $P(Y, X)$ – often through the likelihood $P(X|Y)$ and the prior $P(Y)$ – from which the posterior $P(Y|X)$ can be obtained, while conditional models learn the posterior $P(Y|X)$ directly without spending resources on learning $P(X)$. The 'Bayesian' approach to machine learning is characterised by concentrating on the full posterior distribution wherever this is feasible, rather than just deriving a maximising value.

In Section 9.1 we saw that the normal or Gaussian distribution supports

many useful geometric intuitions, essentially because the negative logarithm of the Gaussian likelihood can be interpreted as a squared distance. Straight decision boundaries result from having the same per-cla covariance matrices, which means that models resulting in such linear boundaries, including linear classifiers, linear regression and *K*-means clustering, can be interpreted from a probabilistic viewpoint that makes their inherent assumptions explicit. Two examples of this are that the basic linear classifier is Bayes-optimal for uncorrelated, unit-variance Gaussian features; and least-squares regression is optimal for linear functions contaminated by Gaussian noise on the target variable.

- Section 9.2 was devoted to different versions of the naive Bayes classifier, which makes the simplifying assumption that features are independent within each class. Lewis (1998) gives an overview and history. This mode is widely used in information retrieval and text classification as it is often good ranker if not a good probability estimator. While the model that is usually understood as naive Bayes treats features as categorical or Bernou random variables, variants employing a multinomial model tend to better model the number of occurrences of words in a document (McCallum and Nigam, 1998). Real-valued features can be taken into account by either modelling them as normally distributed within each class, or by non-parametric density estimation – John and Langley (1995) suggest that the latter gives better empirical results. Webb, Boughton and Wang (2005) discuss ways of relaxing the strong independence assumptions made by naive Bayes. Probability smoothing by means of the *m*-estimate was introduced by Cestnik (1990).

- Perhaps paradoxically, I don't think there is anything particularly 'Bayesian' about the naive Bayes classifier. While it is a generative probabilistic model estimating the posterior $P(Y|X)$ through the joint $P(Y, X)$, in practice the posterior is very poorly calibrated owing to the unrealistic independence assumptions. The reason naive Bayes is often successful is because of the quality of $\text{argmax}_Y\ P(Y|X)$ rather than the quality of the posterior as such, as analysed by Domingos and Pazzani (1997). Furthermore, even the use of Bayes' rule in determining the maximising Y can be avoided, as it only serves to transform uncalibrated likelihoods into uncalibrated posteriors. So my recommendation is to use naive Bayes likelihoods as scores on an unknown scale whose decision

threshold needs to be calibrated by means of ROC analysis, as has been discussed several times before.

- In Section 9.3 we looked at the widely used logistic regression model. The basic idea is to combine a linear decision boundary with logistic calibratio but to train this in a discriminative fashion by optimising conditional likelihood. So, rather than modelling the classes as clouds of points and deriving a decision boundary from those clouds, logistic regression concentrates on areas of class overlap. It is an instance of the larger class c generalised linear models (Nelder and Wedderburn, 1972). Jebara (2004) discusses the advantages of discriminative learning in comparison with generative models. Discriminative learning can also be applied to sequential data in the form of conditional random fields (Lafferty *et al.*, 2001)

- Section 9.4 presented the Expectation-Maximisation algorithm as a genera way of learning models involving unobserved variables. This general form of EM was proposed by Dempster, Laird and Rubin (1977) based on a variety of earlier work. We have seen how it can be applied to Gaussian mixture models to obtain a more general version of K-means predictive clustering, which is also able to estimate cluster shapes and sizes. Howeve this increases the number of parameters of the model and thus the risk of getting stuck in a non-optimal stationary configuration. (Little and Rubin, 1987) is a standard reference for dealing with missing data.

- Finally, in Section 9.5 we briefly discussed some ideas related to learning as compression. The link with probabilistic modelling is that both seek to model and exploit the non-random aspects of the data. In a simplified setting, the minimum description length principle can be derived from Bayes' rule by taking the negative logarithm, and states that models minimising the description length of the model and of the data given the model should be preferred. The first term quantifies the complexity of the model, and the second term quantifies its accuracy (as only the model's errors need to be encoded explicitly). The advantage of the MDL principle is that encoding schemes are often more tangible and easier to define than prior distributions. However, not just any encoding will do: as with their probabilistic counterparts, these schemes need to be justified in the domai being modelled. Pioneering work in this area has been done by Solomono (1964A, B); Wallace and Boulton (1968); Rissanen (1978), among others.

An excellent introduction and overview is provided by Grünwald (2007).

---

[1]Statisticians call a prior that has the same mathematical form as a posterior distribution a *conjugate*

[2]Note that we do not require the two distributions to be equal, but rather that they reach the same maximum for Y. It is not hard to demonstrate that this, too, is not generally the case.

[3]A common mistake is to think that the angle of rotation of the ellipse depends on the correlation coefficient; in fact, it is solely determined by the relative magnitudes of the marginal standard deviations.

[4]Specifically, $x^T Ax$ defines a hyperellipse if A is symmetric and positive definite. Both properties are satisfied if A is the inverse of a non-singular covariance matrix.

[5]Notice that the term 'regression' is a bit of a misnomer here, since, even though a probability estimator approximates an unknown function, the training labels are classes rather than true function values.

[6]Of course, if you're not sure whether the sample is large enough it is better to smooth these relative frequency estimates by, e.g., the *Laplace correction* (Section 2.3).

[7]Model parameters are also 'hidden' in a sense, but they are different from hidden variables in that you would never expect to observe the value of a parameter (e.g., a class mean), whereas a hidden variable could be observed in principle but happens to be unobserved in the case at hand.

CHAPTER *10*

Features

PREVIOUSLY I REFERRED to features as 'the workhorses of machine learning' – it is therefore high time to consider them in more detail. Features, also called attributes, are defined as mappings $f_i : \mathscr{X} \to \mathscr{F}_i$ from the instance space $\mathscr{X}$ to the feature domain $\mathscr{F}_i$. We can distinguish features by their domain: common feature domains include real and integer numbers, but also discrete sets such as colours, the Booleans, and so on. We can also distinguish features by the range of permissible operations. For example, we can calculate a group of people's average age but not their average blood type, so taking the average value is an operation that is permissible on some features but not on others. We will take a closer look at different kinds of feature in Section 10.1.

Although many data sets come with pre-defined features, they can be manipulated in many ways. For example, we can change the domain of a feature by rescaling or discretisation; we can select the best features from a larger set and only work with the selected ones; or we can combine two or more features into a new feature. In fact, a model itself is a way of constructing a new feature that solves the task at hand. Feature transformations will be investigated in Section 10.2, while feature construction and selection is the topic of Section 10.3

10.1 Kinds of feature

Consider two features, one describing a person's age and the other their house number. Both features map into the integers, but the way we use those features can be quite different. Calculating the average age of a group of people is meaningful, but an average house number is probably not very useful! In other words, what matters is not just the domain of a feature, but also the range of permissible operations. These, in turn, depend on whether the feature values are expressed on a meaningful *scale*. Despite appearances, house numbers are not really integers but *ordinals*: we can use them to determine that number 10's neighbours are number 8 and number 12, but we cannot assume that the distance between 8 and 10 is the same as the distance between 10 and 12. Because of the

absence of a linear scale it is not meaningful to add or subtract house numbers, which precludes operations such as averaging.

Calculations on features

Let's take a closer look at the range of possible calculations on features, often referred to as aggregates or statistics. Three main categories are *statistics of central tendency*, *statistics of dispersion* and *shape statistics*. Each of these can be interpreted either as a theoretical property of an unknown population or a concrete property of a given sample – here we will concentrate on sample statistics.

Starting with statistics of central tendency, the most important ones are

- the *mean* or average value;
- the *median*, which is the middle value if we order the instances from lowe to highest feature value; and
- the *mode*, which is the majority value or values.

Of these statistics, the mode is the one we can calculate whatever the domain of the feature: so, for example, we can say that the most frequent blood type in a group of people is O+. In order to calculate the median, we need to have an ordering on the feature values: so we can calculate both the mode and the median house number in a set of addresses.[1] In order to calculate the mean, we need a feature expressed on some scale: most often this will be a linear scale for which we calculate the familiar arithmetic mean, but Background 10.1 discusses means for some other scales. It is often suggested that the median tends to lie between the mode and the mean, but there are plenty of exceptions to this 'rule'. The famous statistician Karl Pearson suggested a more specific rule of thumb (with therefore even more exceptions): the median tends to fall one-third of the way from mean to mode.

Imagine a swimmer who swims the same distance d on two different days, taking a seconds one day and b seconds the next. On average, it took her therefore $c = (a + b)/2$ seconds, with an average speed of $d/c = 2d/(a + b)$. Notice how this average speed is *not* calculated as the normal or *arithmetic*

mean of the speeds, which would yield $(d/a + d/b)/2$: to calculate average speed over a fixed distance we use a different mean called the *harmonic mean*. Given two numbers x and y (in our swimming example these are the speeds on either day, d/a and d/b), the harmonic mean h is defined as

$$h(x,y) = \frac{2}{1/x + 1/y} = \frac{2xy}{x+y}$$

Since $1/h(x, y) = (1/x + 1/y)/2$, we observe that calculating the harmonic mean on a scale with unit u corresponds to calculating the arithmetic mean on the *reciprocal scale* with unit $1/u$. In the example, speed with fixed distance is expressed on a scale reciprocal to the time scale, and since we use the arithmetic mean to average time, we use the harmonic mean to average speed. (If we average speed over a fixed *time* interval this is expressed on the same scale as distance and thus we would use the arithmetic mean.)

A good example of where the harmonic mean is used in machine learning arises when we average precision and recall of a classifier. Remember that precision is the proportion of positive predictions that is correct ($prec = TP/(TP + FP)$), and recall is the proportion of positives that is correctly predicted ($rec = TP/(TP + FN)$). Suppose we first calculate the number of mistakes averaged over the classes: this is the arithmetic mean $Fm = (FP + FN)/2$. We can then derive

$$\frac{TP}{TP+Fm} = \frac{TP}{TP+(FP+FN)/2} = \frac{2TP}{(TP+FP)+(TP+FN)} = \frac{2}{1/prec+1/rec}$$

We recognise the last term as the harmonic mean of precision and recall. Since the enumerator of both precision and recall is fixed, taking the arithmetic mean of the denominators corresponds to taking the harmonic mean of the ratios. In information retrieval this harmonic mean of precision and recall is very often used and called the *F-measure*.

Yet other means exist for other scales. In music, going from one note to a note one octave higher corresponds to doubling the frequency. So frequencies f and $4f$ are two octaves apart, and it makes sense to take the octave in between with frequency $2f$ as their mean. This is achieved by the *geometric mean*, which is defined as $g(x,y) = \sqrt{xy}$. Since $\log\sqrt{xy} = (\log xy)/2 = (\log x + \log y)/2$ it follows that the geometric mean corresponds to the arithmetic mean on a logarithmic scale. All these means have in common that the mean of two values is an intermediate value, and that they can easily be extended to more than two values.

Background 10.1. On scales and means.

The second kind of calculation on features are statistics of dispersion or 'spread'. Two well-known statistics of dispersion are the *variance* or average squared deviation from the (arithmetic) mean, and its square root, the *standard deviation*. Variance and standard deviation essentially measure the same thing, but the latter has the advantage that it is expressed on the same scale as the feature itself. For example, the variance of the body weight in kilograms of a group of people is measured in kg^2 (kilograms-squared), whereas the standard deviation is measured in kilograms. The absolute difference between the mean and the median is never larger than the standard deviation – this is a consequence of *Chebyshev's inequality*, which states that at most $1/k^2$ of the values are more than k standard deviations away from the mean.

A simpler dispersion statistic is the difference between maximum and minimum value, which is called the *range*. A natural statistic of central tendency to be used with the range is the *midrange point*, which is the mean of the two extreme values. These definitions assume a linear scale but can be adapted to other scales using suitable transformations. For example, for a feature expressed on a logarithmic scale, such as frequency, we would take the ratio of the highest and lowest frequency as the range, and the harmonic mean of these two extremes as the midrange point.

Other statistics of dispersion include *percentiles*. The p-th percentile is the value such that p per cent of the instances fall below it. If we have 100 instances, the 80th percentile is the value of the 81st instance in a list of increasing values.[2] If p is a multiple of 25 the percentiles are also called *quartiles*, and if it is a multiple of 10 the percentiles are also called *deciles*. Note that the 50th percentile, the 5th decile and the second quartile are all the same as the median. Percentiles, deciles and quartiles are special cases of *quantiles*. Once we have quantiles we can measure dispersion as the distance between different quantiles. For instance, the *interquartile range* is the difference between the third and first quartile (i.e., the 75th and 25th percentile).

Example 10.1 (Percentile plot). Suppose you are learning a model over an instance space of countries, and one of the features you are considering is

the gross domestic product (GDP) per capita. Figure 10.1 shows a so-called *percentile plot* of this feature. In order to obtain the p-th percentile, you intersect the line $y = p$ with the dotted curve and read off the corresponding percentile on the x-axis. Indicated in the figure are the 25th, 50th and 75th percentile. Also indicated is the mean (which has to be calculated from the raw data). As you can see, the mean is considerably higher than the median; this is mainly because of a few countries with very high GDP per capita. In other words, the mean is more sensitive to *outliers* than the median, which is why the median is often preferred to the mean for skewed distributions like this one.

Figure 10.1. Percentile plot of GDP per capita for 231 countries (data obtained from www.wolframalpha.com by means of the query 'GDP per capita'). The vertical dotted lines indicate, from left to right: the first quartile ($900); the median ($3600); the mean ($11284); andthe third quartile ($14 750). The interquartile range is $13 850, while the standard deviation is $16 189.

You might think that the way I drew the percentile plot is the wrong way around: surely it would make more sense to have p on the x-axis and the percentiles on the y-axis? One advantage of drawing the plot this way is that, by interpreting the y-axis as probabilities, the plot can be read as a *cumulative probability distribution*: a plot of $P(X \leq x)$ against x for a random variable X. For example, the plot shows that $P(X \leq \mu)$ is approximately 0.70, where μ = $11284 is the mean GDP per capita. In other words, if you choose a random country the probability that its GDP per capita is less than the average is about 0.70.

Since GDP per capita is a real-valued feature, it doesn't necessarily make sense

to talk about its mode, since if you measure the feature precisely enough every country will have a different value. We can get around this by means of a *histogram*, which counts the number of feature values in a particular interval or bin.

Figure 10.2. Histogram of the data from Figure 10.1, with bins of $2000 wide.

Example 10.2 (Histogram). A histogram of the data from Example 10.1 is shown in Figure 10.2. The left-most bin is the mode, with well over a third of the countries having a GDP per capita of not more than $2000. This demonstrates that the distribution is extremely *right-skewed* (i.e., has a long right tail), resulting in a mean that is considerably higher than the median.

The skew and 'peakedness' of a distribution can be measured by shape statistics such as skewness and kurtosis. The main idea is to calculate the third and fourth *central moment* of the sample. In general, the k-th central moment of a sample $\{x_i, \ldots, x_n\}$ is defined as $m_k = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^k$, where μ is the sample mean. Clearly, the first central moment is the average deviation from the mean – this is always zero, as the positive and negative deviations cancel each other out – and the second central moment is the average squared deviation from the mean, otherwise known as the variance. The third central moment m_3 can again be positive or negative. *Skewness* is then defined as m_3/σ^3, where σ is the sample's standard deviation. A positive value of skewness means that the

distribution is right-skewed, which means that the right tail is longer than the left tail. Negative skewness indicates the opposite, left-skewed case. *Kurtosis* is defined as m_4/σ^4. As it can be shown that a normal distribution has kurtosis 3, people often use *excess kurtosis* $m_4/\sigma^4 - 3$ as the statistic of interest. Briefly, positive excess kurtosis means that the distribution is more sharply peaked than the normal distribution.

Kind	*Order*	*Scale*	*Tendency*	*Dispersion*	*Shape*
Categorical	×	×	mode	n/a	n/a
Ordinal	√	×	median	quantiles	n/a
Quantitative	√	√	mean	range, interquartile range, variance, standard deviation	skewness, kurtosis

Table 10.1. Kinds of feature, their properties and allowable statistics. Each kind inherits the statistics from the kinds above it in the table. For instance, the mode is a statistic of central tendency that can be computed for any kind of feature.

Example 10.3 (Skewness and kurtosis). In the GDP per capita example we can calculate skewness as 2.12 and excess kurtosis as 2.53. This confirms that the distribution is heavily right-skewed, and also more sharply peaked than the normal distribution.

Categorical, ordinal and quantitative features

Given these various statistics we can distinguish three main kinds of feature: those with a meaningful numerical scale, those without a scale but with an ordering, and those without either. We will call features of the first type *quantitative*; they most often involve a mapping into the reals (another term in common use is 'continuous'). Even if a feature maps into a subset of the reals, such as age expressed in years, the various statistics such as mean or standard deviation still require the full scale of the reals.

Features with an ordering but without scale are called *ordinal features*. The domain of an ordinal feature is some totally ordered set, such as the set of characters or strings. Even if the domain of a feature is the set of integers, denoting the feature as ordinal means that we have to dispense with the scale, as

we did with house numbers. Another common example are features that express a rank order: first, second, third, and so on. Ordinal features allow the mode and median as central tendency statistics, and quantiles as dispersion statistics.

Features without ordering or scale are called *categorical features* (or sometimes 'nominal' features). They do not allow any statistical summary except the mode. One subspecies of the categorical features is the *Boolean feature*, which maps into the truth values true and false. The situation is summarised in Table 10.1.

Models treat these different kinds of feature in distinct ways. First, consider tree models such as decision trees. A split on a categorical feature will have as many children as there are feature values. Ordinal and quantitative features, on the other hand, give rise to a binary split, by selecting a value v_0 such that all instances with a feature value less than or equal to v_0 go to one child, and the remaining instances to the other child. It follows that tree models are insensitive to the scale of quantitative features. For example, whether a temperature feature is measured on the Celsius scale or on the Fahrenheit scale will not affect the learned tree. Neither will switching from a linear scale to a logarithmic scale have any effect: the split threshold will simply be $\log v_0$ instead of v_0. In general, tree models are insensitive to *monotonic* transformations on the scale of a feature, which are those transformations that do not affect the relative order of the feature values. In effect, *tree models ignore the scale of quantitative features, treating them as ordinal*. The same holds for rule models.

Now let's consider the naive Bayes classifier. We have seen that this model works by estimating a likelihood function $P(X|Y)$ for each feature X given the class Y. For categorical and ordinal features with k values this involves estimating $P(X = v_1|Y), \ldots, P(X = v_k|Y)$. In effect, ordinal features are treated as categorical ones, ignoring the order. Quantitative features cannot be handled at all, unless they are discretised into a finite number of bins and thus converted to categoricals. Alternatively, we could assume a parametric form for $P(X|Y)$, for instance a normal distribution. We will return to this later in this chapter when we discuss feature calibration.

While naive Bayes only really handles categorical features, many geometric models go in the other direction: they can only handle quantitative features. Linear models are a case in point: the very notion of linearity assumes a Euclidean instance space in which features act as Cartesian coordinates, and thus need to be quantitative. Distance-based models such as k-nearest neighbour and K-means require quantitative features if their distance metric is Euclidean distance, but we can adapt the distance metric to incorporate categorical features

by setting the distance to 0 for equal values and 1 for unequal values (the ☞ *Hamming distance* as defined in Section 8.1). In a similar vein, for ordinal features we can count the number of values between two feature values (if we encode the ordinal feature by means of integers, this would simply be their difference). This means that distance-based methods can accommodate all feature types by using an appropriate distance metric. Similar techniques can be used to extend support vector machines and other kernel-based methods to categorical and ordinal features.

Structured features

It is usually tacitly assumed that an instance is a vector of feature values. In other words, the instance space is a Cartesian product of d feature domains: $\mathscr{X} = \mathscr{F}_1 \times \ldots \times \mathscr{F}_d$. This means that there is no other information available about an instance apart from the information conveyed by its feature values. Identifying an instance with its vector of feature values is what computer scientists call an *abstraction*, which is the result of filtering out unnecessary information. Representing an e-mail as a vector of word frequencies is an example of an abstraction.

However, sometimes it is necessary to avoid such abstractions, and to keep more information about an instance than can be captured by a finite vector of feature values. For example, we could represent an e-mail as a long string; or as a sequence of words and punctuation marks; or as a tree that captures the HTML mark-up; and so on. Features that operate on such structured instance spaces are called *structured features*.

Example 10.4 (Structured features). Suppose an e-mail is represented as a sequence of words. This allows us to define, apart from the usual word frequency features, a host of other features, including:

- ☞ whether the phrase 'machine learning' – or any other set of consecutive words – occurs in the e-mail;
- ☞ whether the e-mail contains at least eight consecutive words in a language other than English;
- ☞ whether the e-mail is palindromic, as in 'Degas, are we not drawn onward, we freer few, drawn onward to new eras aged?'

Furthermore, we could go beyond properties of single e-mails and express relations such as whether one e-mail is quoted in another e-mail, or whether two e-mails have one or more passages in common.

Structured features are not unlike *queries* in a database query language such as SQL or a declarative programming language such as Prolog. In fact, we have already seen examples of structured features in Section 6.4 when we looked at learning Prolog clauses such as the following:

```
fish(X):-bodyPart(X,Y).
fish(X):-bodyPart(X,pairOf(Z)).
```

The first clause has a single structured feature in the body which tests for the existence of some unspecified body part, while the second clause has another structured feature testing for the existence of a pair of unspecified body parts. The defining characteristic of structured features is that they involve *local variables* that refer to objects other than the instance itself. In a logical language such as Prolog it is natural to interpret local variables as existentially quantified, as we just did. However, it is equally possible to use other forms of *aggregation* over local variables: e.g., we can count the number of body parts (or pairs of body parts) an instance has.

↓ to, from →	*Quantitative*	*Ordinal*	*Categorical*	*Boolean*
Quantitative	**normalisation**	**calibration**	**calibration**	**calibration**
Ordinal	**discretisation**	**ordering**	**ordering**	**ordering**
Categorical	**discretisation**	**unordering**	**grouping**	
Boolean	**thresholding**	**thresholding**	**binarisation**	

Table 10.2. An overview of possible feature transformations. **Normalisation and calibration** adapt the scale of quantitative features, or add a scale to features that don't have one. **Ordering** adds or adapts the order of feature values without reference to a scale. The other operations abstract away from unnecessary detail, either in a deductive way (**unordering**, **binarisation**) or by introducing new information (**thresholding**, **discretisation**).

Structured features can be constructed either prior to learning a model, or simultaneously with it. The first scenario is often called *propositionalisation* because the features can be seen as a translation from first-order logic to propositional logic without local variables. The main challenge with propositionalisation approaches is how to deal with combinatorial explosion of the number of potential features. Notice that features can be logically related:

e.g., the second clause above covers a subset of the instances covered by the first one. It is possible to exploit this if structured feature construction is integrated with model building, as in inductive logic programming.

10.2 Feature transformations

Feature transformations aim at improving the utility of a feature by removing, changing or adding information. We could order feature types by the amount of detail they convey: quantitative features are more detailed than ordinal ones, followed by categorical features, and finally Boolean features. The best-known feature transformations are those that turn a feature of one type into another of the next type down this list. But there are also transformations that change the scale of quantitative features, or add a scale (or order) to ordinal, categorical and Boolean features. Table 10.2 introduces the terminology we will be using.

The simplest feature transformations are entirely deductive, in the sense that they achieve a well-defined result that doesn't require making any choices. *Binarisation* transforms a categorical feature into a set of Boolean features, one for each value of the categorical feature. This loses information since the values of a single categorical feature are mutually exclusive, but is sometimes needed if a model cannot handle more than two feature values. *Unordering* trivially turns an ordinal feature into a categorical one by discarding the ordering of the feature values. This is often required since most learning models cannot handle ordinal features directly. An interesting alternative that we will explore below is to add a scale to the feature by means of calibration.

Figure 10.3. (left) Coverage curve obtained by ranking countries on decreasing GDP per capita, using 23 Euro countries as the positive class. The orange split sets the threshold equal to the mean, selecting 19 Euro countries and 49 non-Euro countries. The green split sets the threshold equal to the median, selecting 21 Euro countries and 94 non-Euro countries. The red points are on the convex hull of the coverage curve and

indicate potentially optimal splits when the class label is taken into account. (**right**) Coverage curve of the same feature, using 50 countries in the Americas as the positive class. The red splits indicate potentially optimal thresholds with relatively many positives above the threshold, while the green splits indicate potentially optimal thresholds with relatively many positives below the threshold.

In the remainder of this section we consider feature transformations that add information, the most important of which are discretisation and calibration.

Thresholding and discretisation

Thresholding transforms a quantitative or an ordinal feature into a Boolean feature by finding a feature value to split on. I briefly alluded to this in Chapter 5 as a way to split on quantitative features in decision trees. Concretely, let $f : \mathcal{X} \to \mathbb{R}$ be a quantitative feature and let $t \in \mathbb{R}$ be a threshold, then $f_t : \mathcal{X} \to \{\text{true}, \text{false}\}$ is a Boolean feature defined by $f_t(x) = \text{true}$ if $f(x) \geq t$ and $f_t(x) = \text{false}$ if $f(x) < t$. We can choose such thresholds in an unsupervised or a supervised way.

Example 10.5 (Unsupervised and supervised thresholding). Consider the GDP per capita feature plotted in Figure 10.1 again. Without knowing how this feature is to be used in a model, the most sensible thresholds are the statistics of central tendency such as the mean and the median. This is referred to as *unsupervised thresholding*.

In a *supervised* learning setting we can do more. For example, suppose we want to use the GDP per capita as a feature in a decision tree to predict whether a country is one of the 23 countries that use the Euro as their official currency (or as one of their currencies). Using the feature as a ranker, we can construct a coverage curve (Figure 10.3 (left)). We see that for this feature the mean is not the most obvious threshold, as it splits right in the middle of a run of negatives. A better split is obtained at the start of that run of negatives, or at the end of the following run of positives, indicated by the red points at either end of the mean split. More generally, any point on the convex hull of the coverage curve represents a candidate threshold; which one to choose is informed by whether we put more value on picking out positives or negatives. As it happens in this example, the median threshold is on the convex hull, but this cannot be guaranteed in general as, by definition, unsupervised thresholding methods select the

threshold independently from the target.

Figure 10.3 (right) shows the same feature with a different target: whether a country is in the Americas. We see that part of the curve is below the ascending diagonal, indicating that, in comparison with the whole data set, the initial segment of the ranking contains a smaller proportion of American countries. This means that potentially useful thresholds can also be found on the *lower convex hull*.

In summary, unsupervised thresholding typically involves calculating some statistic over the data, whereas supervised thresholding requires sorting the data on the feature value and traversing down this ordering to optimise a particular objective function such as information gain. Non-optimal split points could be filtered out by means of constructing the upper and lower convex hull, but in practice this is unlikely to be more efficient computationally than a straightforward sweep over the sorted instances.

If we generalise thresholding to multiple thresholds we arrive at one of the most commonly used non-deductive feature transformations. *Discretisation* transforms a quantitative feature into an ordinal feature. Each ordinal value is referred to as a *bin* and corresponds to an interval of the original quantitative feature. Again, we can distinguish between supervised and unsupervised approaches. *Unsupervised discretisation* methods typically require one to decide the number of bins beforehand. A simple method that often works reasonably well is to choose the bins so that each bin has approximately the same number of instances: this is referred to as *equal-frequency discretisation*. If we choose two bins then this method coincides with thresholding on the median. More generally, the bin boundaries are quantiles: for instance, with 10 bins the bin boundaries of equal-width discretisation are deciles. Another unsupervised discretisation method is *equal-width discretisation*, which chooses the bin boundaries so that each interval has the same width. The interval width can be established by dividing the feature range by the number of bins if the feature has upper and lower limits; alternatively, we can take the bin boundaries at an integer number of standard deviations above and below the mean. An interesting alternative is to treat feature discretisation as a univariate clustering problem. For example, in order to generate K bins we can uniformly sample K initial bin centres and run K-means until convergence. We can alternatively use any of the other clustering methods discussed in Chapter 8: K-medoids, partitioning around medoids and hierarchical agglomerative clustering.

Switching now to *supervised discretisation* methods, we can distinguish between *top–down* or *divisive* discretisation methods on the one hand, and *bottom–up* or *agglomerative* discretisation methods on the other. Divisive methods work by progressively splitting bins, whereas agglomerative methods proceed by initially assigning each instance to its own bin and successively merging bins. In either case an important role is played by the *stopping criterion*, which decides whether a further split or merge is worthwhile. We give an example of each strategy. A natural generalisation of thresholding leads to a top–down *recursive partitioning* algorithm (Algorithm 10.1). This discretisation algorithm finds the best threshold according to some scoring function Q, and proceeds to recursively split the left and right bins. One scoring function that is often used is information gain.

Example 10.6 (Recursive partitioning using information gain). Consider the following feature values, which are ordered on increasing value for convenience.

Instance	*Value*	*Class*
e_1	–5.0	⊖
e_2	–3.1	⊕
e_3	–2.7	⊖
e_4	0.0	⊖
e_5	7.0	⊖
e_6	7.1	⊕
e_7	8.5	⊕
e_8	9.0	⊖
e_9	9.0	⊕
e_{10}	13.7	⊖
e_{11}	15.1	⊖
e_{12}	20.1	⊖

This feature gives rise to the following ranking: ⊖⊕⊖⊖⊖⊕⊕[⊖⊕]⊖⊖⊖,

where the square brackets indicate a tie between instances e_8 and e_9. The corresponding coverage curve is depicted in Figure 10.4. Tracing information gain isometrics through each possible split, we see that the best split is ⊖⊕⊖⊖⊖|⊕⊕[⊖⊕]|⊖⊖⊖. Repeating the process once more gives the discretisation ⊖⊕⊖⊖⊖|⊕⊕[⊖⊕]|⊖⊖⊖.

Figure 10.4. (left) A coverage curve visualising the ranking of four positive and eight negative examples by a feature to be discretised. The curved lines are information gain isometrics through possible split points; the solid isometric indicates the best split [4+,5–][0+, 3–] according to information gain. (**middle**) Recursive partitioning proceeds to split the segment [4+,5–] into [1+,4–][3+,1–]. (**right**) Ifwe stop here, the blue curve visualises the discretised (but still ordinal) feature.

Clearly, we can stop the recursive partitioning algorithm when the empirical probabilities are the same across the ranking; this has pure bins and bins with a constant feature value as special cases. With this stopping criterion, the algorithm will successfully identify all straight line segments in the ranking. In fact, it is not hard to see that this holds true even if we change the scoring function – the split points may be found in a different order, but the end result will be the same. In practice more aggressive stopping criteria are used, which does mean that the end result depends on the scoring function. For example, in Figure 10.4 we see that the split ⊖|[⊕⊖]⊕⊕⊖⊖⊖⊕⊕ has the second-highest information gain but ends up not being chosen at all, while with a different scoring function it might have been chosen in the first round. One of the most popular stopping criteria applies a minimum description length argument to decide whether a given bin should be split further.

Algorithm 10.1: RecPart(S, f, Q) – supervised discretisation by means of recursive partitioning.

Input : set of labelled instances S ranked on feature values $f(x)$; scoring function Q.

Output : sequence of thresholds $t_1, \ldots, t_{k-1}$.

```
if stopping criterion applies then return ∅;
Split S into S_l and S_r using threshold t that optimises Q ;
T_l = RecPart(S_l, f, Q);
T_r = RecPart(S_r, f, Q);
return T_l ∪ {t} ∪ T_r;
```

Algorithm 10.2: AggloMerge(S, f, Q) – supervised discretisation by means of agglomerative merging.

Input : set of labelled instances S ranked on feature values $f(x)$; scoring function Q.

Output : sequence of thresholds.

```
initialise bins to data points with the same scores;
merge consecutive pure bins ;                    // optional optimisation
repeat
    evaluate Q on consecutive bin pairs;
    merge the pairs with best Q (unless they invoke the stopping criterion);
until no further merges are possible;
return thresholds between bins;
```

It should be noted that the data set in Example 10.6 is probably so small that the stopping criterion will kick in straight away and recursive partitioning will be unable to go beyond a single bin. More generally, this kind of discretisation tends to be fairly conservative. For example, on the Euro data in Figure 10.3 (left) recursive partitioning produces two bins, selecting 20 Euro countries and 53 non-Euro countries (the red point in between the mean and median splits). On the American countries data in Figure 10.3 (right) we again obtain two bins, corresponding to the third red point from the right.

An algorithm for bottom–up *agglomerative merging* is given in Algorithm 10.2. Again the algorithm can take various choices for the scoring function and the stopping criterion: a popular choice is to use the χ^2 *statistic* for both.

Example 10.7 (Agglomerative merging using χ^2). We continue Example 10.6. Algorithm 10.2 initialises the bins to ⊖|⊕|⊖⊖⊖|⊕⊕|[⊖⊕]|⊖⊖⊖. We illustrate the calculation of the χ^2 statistic for the last two bins. We construct the following contingency table:

	Left bin	*Right bin*	
⊕	**1**	**0**	**1**
⊖	**1**	**3**	**4**
	2	**3**	**5**

At the basis of the χ^2 statistic lies a comparison of these observed frequencies with expected frequencies obtained from the row and column marginals. For example, the marginals say that the top row contains 20% of the total mass and the left column 40%; so if rows and columns were statistically independent we would expect 8% of the mass – or 0.4 of the five instances – in the top-left cell. Following a clockwise direction, the expected frequencies for the other cells are 0.6, 2.4 and 1.6. If the observed frequencies are close to the expected ones, this suggests that these two bins are candidates for merging since the split appears to have no bearing on the class distribution.

The χ^2 statistic sums the squared differences between the observed and expected frequencies, each term normalised by the expected frequency:

$$\chi^2 = \frac{(1-0.4)^2}{0.4} + \frac{(0-0.6)^2}{0.6} + \frac{(3-2.4)^2}{2.4} + \frac{(1-1.6)^2}{1.6} = 1.88$$

Going left-to-right through the other pairs of consecutive bins, the χ^2 values are 2, 4, 5 and 1.33 (there's an easy way to calculate the χ^2 value for two pure bins, which I'll leave you to discover). This tells us that the fourth and fifth bin are first to be merged, leading to ⊖|⊕|⊖⊖⊖|⊕⊕[⊖⊕]|⊖⊖⊖. We then recompute the χ^2 values (in fact, only those involving the newly merged bin need to be recomputed), yielding 2, 4, 3.94 and 3.94. We now merge the first two bins, giving the partition ⊖⊕|⊖⊖⊖|⊕⊕[⊖⊕]|⊖⊖⊖. This changes the first χ^2 value to 1.88, so we again merge the first two bins, arriving at ⊖⊕⊖⊖⊖|⊕⊕[⊖⊕]|⊖⊖⊖ (the same three bins as in Example 10.6).

In agglomerative discretisation the stopping criterion usually takes the form of a simple threshold on the scoring function. In the case of the χ^2 statistic, the threshold can be derived from the p-value associated with the χ^2 distribution, which is the probability of observing a χ^2 value above the threshold if the two variables are actually independent. For two classes (i.e., one degree of freedom) and a p-value of 0.10 the χ^2 threshold is 2.71, which in our example means that we stop at the above three bins. For a lower p-value of 0.05 the χ^2 threshold is 3.84, which means that we eventually merge all the bins.

Notice that both top–down and bottom–up supervised discretisation bear some resemblance to algorithms we have seen previously: recursive partitioning shares the divide-and-conquer nature of the ☞*decision tree* training algorithm (Algorithm 5.1 on p.5), and agglomerative discretisation by merging consecutive bins is related to ☞*hierarchical agglomerative clustering* (Algorithm 8.4 on p.8). It is also worth mentioning that, although our examples were predominantly drawn from binary classification, most methods can handle more than two classes without complication.

Normalisation and calibration

Thresholding and discretisation are feature transformations that remove the scale of a quantitative feature. We now turn our attention to adapting the scale of a quantitative feature, or adding a scale to an ordinal or categorical feature. If this is done in an unsupervised fashion it is usually called normalisation, whereas calibration refers to supervised approaches taking in the (usually binary) class labels. *Feature normalisation* is often required to neutralise the effect of different quantitative features being measured on different scales. If the features are approximately normally distributed, we can convert them into ☞*z-scores* (Background 9.1 on p.267) by centring on the mean and dividing by the standard deviation. In certain cases it is mathematically more convenient to divide by the variance instead, as we have seen in Section 7.1. If we don't want to assume normality we can centre on the median and divide by the interquartile range.

Sometimes feature normalisation is understood in the stricter sense of expressing the feature on a [0,1] scale. This can be achieved in various ways. If we know the feature's highest and lowest values h and l, then we can simply apply the linear scaling $f \rightarrow (f - l)/(h - l)$. We sometimes have to guess the value of h or l, and truncate any value outside $[l, h]$. For example, if the feature measures age in years, we may take $l = 0$ and $h = 100$, and truncate any $f > h$ to

1. If we can assume a particular distribution for the feature, then we can work out a transformation such that almost all feature values fall in a certain range. For instance, we know that more than 99% of the probability mass of a normal distribution falls within $\pm 3\sigma$ of the mean, where σ is the standard deviation, so the linear scaling $f \rightarrow (f - \mu)/6\sigma + 1/2$ virtually removes the need for truncation.

Feature calibration is understood as a supervised feature transformation adding a meaningful scale carrying class information to arbitrary features. This has a number of important advantages. For instance, it allows models that require scale, such as linear classifiers, to handle categorical and ordinal features. It also allows the learning algorithm to choose whether to treat a feature as categorical, ordinal or quantitative. We will assume a binary classification context, and so a natural choice for the calibrated feature's scale is the posterior probability of the positive class, conditioned on the feature's value. This has the additional advantage that models that are based on such probabilities, such as naive Bayes, do not require any additional training once the features are calibrated, as we shall see. The problem of feature calibration can thus be stated as follows: given a feature $F : \mathscr{X} \rightarrow \mathscr{F}$, construct a calibrated feature $F^c : \mathscr{X} \rightarrow [0,1]$ such that $F^c(x)$ estimates the probability $F^c(x) = P(\oplus|v)$, where $v = F(x)$ is the value of the original feature for x.

For categorical features this is as straightforward as collecting relative frequencies from a training set.

Example 10.8 (Calibration of categorical features). Suppose we want to predict whether or not someone has diabetes from categorical features including whether the person is obese or not, whether he or she smokes, and so on. We collect some statistics which tell us that 1 in every 18 obese persons has diabetes while among non-obese people this is 1 in 55 (data obtained from `www.wolframalpha.com` with the query 'diabetes'). If $F(x) = 1$ for person x who is obese and $F(y) = 0$ for person y who isn't, then the calibrated feature values are $F^c(x) = 1/18 = 0.055$ and $F^c(y) = 1/55 = 0.018$.

In fact, it would be better to compensate for the non-uniform class distribution, in order to avoid over-emphasising the class prior, which is better taken into account in the decision rule. This can be achieved as follows. If m of n obese people have diabetes, then this corresponds to a posterior odds of $m/(n - m)$ or a

likelihood ratio of $m/c(n-m)$, where c is the prior odds of having diabetes (since posterior odds is likelihood ratio times prior odds). Working with the likelihood ratio is equivalent to assuming a uniform class distribution. Converting the likelihood ratio into a probability gives

$$F^c(x) = \frac{\frac{m}{c(n-m)}}{\frac{m}{c(n-m)}+1} = \frac{m}{m+c(n-m)}$$

In our example, if the prior odds of having diabetes is $c = 1/48$, then $F^c(x) = 1/(1 + 17/48) = 48/(48 + 17) = 0.74$. The extent to which this probability is more than 1/2 quantifies the extent to which obese people are more likely than average to have diabetes. For non-obese people the probability is $1/(1 + 54/48) = 48/(48 + 54) = 0.47$, so they are slightly less likely than average to have diabetes. Keep in mind also that it is usually a good idea to smooth these probability estimates by means of the Laplace correction, which adds 1 to m and 2 to n. This leads to the final expression for calibrating a categorical feature:

$$F^c(x) = \frac{m+1}{m+1+c(n-m+1)}$$

Ordinal and quantitative features can be discretised and then calibrated as categorical features. In the remainder of this section we look at calibration methods that maintain the ordering of the feature. For example, suppose we want to use body weight as an indicator for diabetes. A calibrated weight feature attaches a probability to every weight, such that these probabilities are non-decreasing with weight. This is related to our discussion of ☞*calibrating classifier scores* in Section 7.4, as those calibrated probabilities should likewise takes the ranking of the classifier's predictions into account. In fact, the two approaches to classifier calibration – by employing the logistic function and by constructing the ROC convex hull – are directly applicable to feature calibration, since a quantitative feature can simply be treated as a univariate scoring classifier.

We briefly reiterate the main points of *logistic calibration*, but with a slight change in notation. Let $F : \mathcal{X} \to \mathbb{R}$ be a quantitative feature with class means $\mu^{\oplus}$ and $\mu^{\ominus}$ and variance σ^2. Assuming the feature is normally distributed within each class with the same variance, we can express the likelihood ratio of a feature value v as

$$LR(v) = \frac{P(v|\oplus)}{P(v|\ominus)} = \exp\left(\frac{-(v-\mu^{\oplus})^2 + (v-\mu^{\ominus})^2}{2\sigma^2}\right)$$
$$= \exp\left(\frac{\mu^{\oplus}-\mu^{\ominus}}{\sigma}\frac{v-(\mu^{\oplus}+\mu^{\ominus})/2}{\sigma}\right) = \exp\left(d'z\right)$$

where $d' = (\mu^{\oplus} - \mu^{\ominus})/\sigma$ is the difference between the means in proportion to the standard deviation, which is known as *d-prime* in signal detection theory; and $z = (v - \mu)/\sigma$ is the z-score associated with v (notice we take the mean as $\mu = (\mu^{\oplus} + \mu^{\ominus})/2$ to simulate an equal class distribution). Again we work directly with the likelihood ratio to neutralise the effect of a non-uniform class distribution, and we obtain the calibrated feature value as

$$F^c(x) = \frac{LR(F(x))}{1+LR(F(x))} = \frac{\exp\left(d'z(x)\right)}{1+\exp\left(d'z(x)\right)}$$

You may recognise the logistic function we discussed in Chapter 7 (see Figure 7.11 on p.222).

In essence, logistic feature calibration performs the following steps.

1. Estimate the class means $\mu^{\oplus}$ and $\mu^{\ominus}$ and the standard deviation σ.

2. Transform $F(x)$ into z-scores $z(x)$, making sure to use $\mu = (\mu^{\oplus} + \mu^{\ominus})/2$ as the feature mean.

3. Rescale the z-scores to $F^d(x) = d'\, z(x)$ with $d' = (\mu^{\oplus} - \mu^{\ominus})/\sigma$.

4. Apply a sigmoidal transformation to $F^d(x)$ to give calibrated probabilities $F^c(x) = \frac{\exp(F^d(x))}{1+\exp(F^d(x))}$.

Sometimes it is preferred to work directly with $F^d(x)$, as it is expressed on a scale linearly related to the original feature's scale, and the Gaussian assumption implies that we expect that scale to be additive. For example, distance-based models expect additive features in order to calculate Euclidean distance. In contrast, the scale of F^c is multiplicative. Notice that the two are interdefinable as $F^d(x) = \ln\frac{F^c(x)}{1-F^c(x)} = \ln F^c(x) - \ln(1-F^c(x))$. I will call the feature space spanned by F^d *log-odds space*, since $\exp(F^d(x)) = LR(x)$ and the likelihood ratio is equal to the odds if we're assuming a uniform class prior. Calibrated features F^c live in *probability space*.

Figure 10.5. (left) Two-class Gaussian data. The middle line is the decision boundary learned by the basic linear classifier; the other two are parallel lines through the class means. (**middle**) Logistic calibration to log-odds space is a linear transformation; assuming unit standard deviations, the basic linear classifier is now the fixed line $F_1^{\mathrm{d}}(x) + F_2^{\mathrm{d}}(x) = 0$. (**right**) Logistic calibration to probability space is a non-linear transformation that pushes data away from the decision boundary.

Example 10.9 (Logistic calibration of two features). Logistic feature calibration is illustrated in Figure 10.5. I generated two sets of 50 points by sampling bivariate Gaussians with identity covariance matrix, centred at (2,2) and (4,4). I then constructed the basic linear classifier as well as two parallel decision boundaries through the class means. Tracing these three lines in calibrated space will help us understand feature calibration.

In the middle figure we see the transformed data in log-odds space, which is clearly a linear rescaling of the axes. The basic linear classifier is now the line $F_1^{\mathrm{d}}(x) + F_2^{\mathrm{d}}(x) = 0$ through the origin. In other words, for this simple classifier feature calibration has removed the need for further training: instead of fitting a decision boundary to the data, we have fitted the data to a fixed decision boundary! (I should add that I cheated very slightly here, as I fixed $\sigma = 1$ in the calibration process – had I estimated each feature's standard deviation from the data, the decision boundary would most likely have had a slightly different slope.)

On the right we see the transformed data in probability space, which clearly has a non-linear relationship with the other two feature spaces. The basic linear classifier is still linear in this space, but actually this is no longer true for more than two features. To see this, note that $F_1^{\mathrm{c}}(x) + F_2^{\mathrm{c}}(x) = 1$ can be rewritten as

$$\frac{\exp\left(F_1^{\mathrm{d}}(x)\right)}{1+\exp\left(F_1^{\mathrm{d}}(x)\right)} + \frac{\exp\left(F_2^{\mathrm{d}}(x)\right)}{1+\exp\left(F_2^{\mathrm{d}}(x)\right)} = 1$$

which can be simplified to $\exp\left(F_1^{\mathrm{d}}(x)\right)\exp\left(F_2^{\mathrm{d}}(x)\right) = 1$ and hence to $F_1^{\mathrm{d}}(x) + F_2^{\mathrm{d}}(x) = 0$. However, if we add a third feature not all cross-terms cancel and we obtain a non-linear boundary.

The log-odds representation does hold an interest in another respect. An arbitrary linear decision boundary in log-odds space is represented by $\sum_i w_i F_i^{\mathrm{d}}(x) = t$. Taking natural logarithms this can be rewritten as

$$\exp\left(\sum_i w_i F_i^{\mathrm{d}}(x)\right) = \prod_i \exp\left(w_i F_i^{\mathrm{d}}(x)\right) = \prod_i \left(\exp\left(F_i^{\mathrm{d}}(x)\right)\right)^{w_i} = \prod_i LR_i(x)^{w_i} = \exp(t) = t'$$

This exposes a connection with the naive Bayes models discussed in Section 9.2, whose decision boundaries are also defined as products of likelihood ratios for individual features. The basic naive Bayes model has $w_i = 1$ for all i and $t' = 1$, which means that *fitting data to a fixed linear decision boundary in log-odds space by means of feature calibration can be understood as training a naive Bayes model*. Changing the slope of the decision boundary corresponds to introducing non-unit feature weights, which is similar to the way feature weights arose in the multinomial naive Bayes model.

It is instructive to investigate the distribution of the calibrated feature a bit more (I will omit the technical details). Assuming the uncalibrated distributions were two Gaussian bumps, what do the calibrated distributions look like? We have already seen that calibrated data points are pulled away from the decision boundary, so we would expect the peaks of the calibrated distributions to be closer to their extreme values. How much closer depends solely on d'; Figure 10.6 depicts the calibrated distributions for various values of d'.

We move on to *isotonic calibration*, a method that requires order but ignores scale and can be applied to both ordinal and quantitative features. We essentially use the feature as a univariate ranker, and construct its ROC curve and convex hull to obtain a piecewise-constant calibration map. Suppose we have an ROC curve, and suppose the i-th segment of the curve involves n_i training examples, out of which m_i are positives. The corresponding ROC segment has slope $l_i = m_i/(c(n_i - m_i))$, where c is the prior odds. Suppose first the ROC curve is convex: i.e., $i < j$ implies $l_i \geq l_j$. In that case, we can use the same formula as for categorical features to obtain a calibrated feature value:

$$v_l^c = \frac{m_l + 1}{m_l + 1 + c(n_l - m_l + 1)} \tag{10.1}$$

Figure 10.6. Per-class distributions of a logistically calibrated feature for different values of d', the distance between the uncalibrated class means in proportion to the feature's standard deviation. The red and blue curves depict the distributions for the positive and negative class for a feature whose means are one standard deviation apart ($d' = 1$). The other curves are for $d' \in \{0.5, 1.4, 1.8\}$.

As before, this achieves both probability smoothing through Laplace correction and compensation for non-uniform class distributions. If the ROC curve is not convex, there exist $i < j$ such that $l_i < l_j$. Assuming we want to maintain the original feature ordering, we first construct the convex hull of the ROC curve. The effect of this is that we join adjacent segments in the ROC curve that are part of a concavity, until no concavities remain. We recalculate the segments and assign calibrated feature values as in Equation 10.1.

Example 10.10 (Isotonic feature calibration). The following table gives sample values of a weight feature in relation to a diabetes classification problem. Figure 10.7 shows the ROC curve and convex hull of the feature and the calibration map obtained by isotonic calibration.

Weight	*Diabetes?*	*Calibrated weight*	*Weight*	*Diabetes?*	*Calibrated weight*
130	⊕	0.83	81	⊖	0.43
127	⊕	0.83	80	⊕	0.43
111	⊕	0.83	79	⊖	0.43
106	⊕	0.83	77	⊕	0.43
103	⊖	0.60	73	⊖	0.40
96	⊕	0.60	68	⊖	0.40
90	⊕	0.60	67	⊕	0.40
86	⊖	0.50	64	⊖	0.20
85	⊕	0.50	61	⊖	0.20
82	⊖	0.43	56	⊖	0.20

For example, a weight of 80 kilograms is calibrated to 0.43, meaning that three out of seven people in that weight interval have diabetes (after Laplace correction).

Figure 10.7. (left) ROC curve and convex hull of an uncalibrated feature. Calibrated feature values are obtained from the proportion of positives in each segment of the ROC convex hull. (right) The corresponding piecewise-constant calibration map, which maps the uncalibrated feature values on the *x*-axis to the calibrated feature values on the *y*-axis.

Example 10.11 gives a bivariate illustration. As is clearly visible, for quantitative features the process amounts to supervised discretisation of the feature values, which means that many points are mapped to the same point in calibrated space. This is different from logistic calibration, which is invertible.

Example 10.11 (Isotonic calibration of two features). Figure 10.8 shows the result of isotonic calibration on the same data as in Example 10.9, both in log-odds space and in probability space. Because of the discrete nature of

isotonic calibration, even the transformation to log-odds space is no longer linear: the basic linear classifier becomes a series of axis-parallel line segments. This is also true in the opposite direction: if we imagine a linear decision boundary in log-odds space or in probability space, this maps to a decision boundary following the dotted lines in the original feature space. Effectively, isotonic feature calibration has changed the linear grading model into a grouping model.

Figure 10.8. (left) The data from Figure 10.5, with grid lines indicating the discretisation obtained by isotonic feature calibration. (**middle**) Isotonically calibrated data in log-odds space. (**right**) Isotonically calibrated data in probability space.

In summary, isotonic feature calibration performs the following steps.

1. Sort the training instances on feature value and construct the ROC curve. The sort order is chosen such that the ROC curve has AUC $\geq 1/2$.

2. Construct the convex hull of this curve, and count the number of positives m_i and the total number of instances n_i in each segment of the convex hull.

3. Discretise the feature according to the convex hull segments, and associate a calibrated feature value $v_i^{\mathrm{c}} = \frac{m_i+1}{m_i+1+c(n_i-m_i+1)}$ with each segment.

4. If an additive scale is required, use $v_i^{\mathrm{d}} = \ln \frac{v_i^{\mathrm{c}}}{1-v_i^{\mathrm{c}}} = \ln v_i^{\mathrm{c}} - \ln(1 - v_i^{\mathrm{c}})$.

Incomplete features

At the end of this section on feature transformations we briefly consider what to do if we don't know a feature's value for some of the instances. We encountered

this situation in Example 1.2 on p.26, where we discussed how to classify an e-mail if we didn't know whether it contained one of the vocabulary words or not. Probabilistic models handle this rather gracefully by taking a weighted average over all possible values of the feature:

$$P(Y|X) = \sum_z P(Y, Z = z|X) = \sum_z P(Y|X, Z = z)P(Z = z)$$

Here, Y is the target variable as usual, X stands for the features that are observed for the instance to be classified, while Z are the features that are unobserved at classification time. The distribution $P(Z)$ can be obtained from the trained model, at least for a generative model – if our model is discriminative we need to estimate $P(Z)$ separately.

Missing feature values at training time are trickier to handle. First of all, the very fact that a feature value is missing may be correlated with the target variable. For example, the range of medical tests carried out on a patient is likely to depend on their medical history. For such features it may be best to have a designated 'missing' value so that, for instance, a tree model can split on it. However, this would not work for, say, a linear model. In such cases we can complete the feature by 'filling in' the missing values, a process known as *imputation*. For instance, in a classification problem we can calculate the per-class means, medians or modes over the observed values of the feature and use this to impute the missing values. A somewhat more sophisticated method takes feature correlation into account by building a predictive model for each incomplete feature and uses that model to 'predict' the missing value. It is also possible to invoke the ☞*Expectation-Maximisation* algorithm (Section 9.4), which goes roughly as follows: assuming a multivariate model over all features, use the observed values for maximum-likelihood estimation of the model parameters, then derive expectations for the unobserved feature values and iterate.

10.3 Feature construction and selection

The previous section on feature transformation makes it clear that there is a lot of scope in machine learning to play around with the original features given in the data. We can take this one step further by constructing new features from several original features. A simple example of this can be used to improve the ☞ *naive Bayes* classifier discussed in Section 9.2. Remember that in text

classification applications we have a feature for every word in the vocabulary, which disregards not only the order of words but also their adjacency. This means that sentences such as 'they write about machine learning' and 'they are learning to write about a machine' will be virtually indistinguishable, even though the former is about machine learning and the latter is not. It may therefore sometimes be necessary to include phrases consisting of multiple words in the dictionary and treat them as single features. In the information retrieval literature, a multi-word phrase is referred to as an *n-gram* (unigram, bigram, trigram and so on).

Taking this idea one step further, we can construct a new feature from two Boolean or categorical features by forming their Cartesian product. For example, if we have one feature Shape with values Circle, Triangle and Square, and another feature Colour with values Red, Green and Blue, then their Cartesian product would be the feature (Shape, Colour) with values (Circle, Red), (Circle, Green), (Circle, Blue), (Triangle, Red), and so on. The effect that this would have depends on the model being trained. Constructing Cartesian product features for a naive Bayes classifier means that the two original features are no longer treated as independent, and so this reduces the strong bias that naive Bayes models have. This is not the case for tree models, which can already distinguish between all possible pairs of feature values. On the other hand, a newly introduced Cartesian product feature may incur a high information gain, so it can possibly affect the model learned.

There are many other ways of combining features. For instance, we can take arithmetic or polynomial combinations of quantitative features (we saw examples of this in the use of a ☞*kernel* in Example 1.9 on p.43 and Section 7.5). One attractive possibility is to first apply concept learning or subgroup discovery, and then use these concepts or subgroups as new Boolean features. For instance, in the dolphin domain we could first learn subgroups such as Length = [3,5] ∧ Gills = no and use these as Boolean features in a subsequent tree model. Notice that this expands the expressive power of tree models through the use of negation: e.g., (Length = [3,5] ∧ Gills = no) = false is equivalent to the disjunction Length ≠ [3,5] ∨ Gills = yes, which is not directly expressible by a feature tree.

Once we have constructed new features it is often a good idea to select a suitable subset of them prior to learning. Not only will this speed up learning as fewer candidate features need to be considered, it also helps to guard against overfitting. There are two main approaches to feature selection. The *filter* approach scores features on a particular metric and the top-scoring features are selected. Many of the metrics we have seen so far can be used for feature

scoring, including information gain, the χ^2 statistic, the correlation coefficient, to name just a few. An interesting variation is provided by the *Relief* feature selection method, which repeatedly samples a random instance x and finds its nearest hit h (instance of the same class) as well as its nearest miss m (instance of opposite class). The i-th feature's score is then decreased by $\mathrm{Dis}(x_i, h_i)^2$ and increased by $\mathrm{Dis}(x_i, m_i)^2$, where Dis is some distance measure (e.g., Euclidean distance for quantitative features, Hamming distance for categorical features). The intuition is that we want to move closer to the nearest hit while differentiating from the nearest miss.

One drawback of a simple filter approach is that no account is taken of redundancy between features. Imagine, for the sake of the argument, duplicating a promising feature in the data set: both copies score equally high and will be selected, whereas the second one provides no added value in the context of the first one. Secondly, feature filters do not detect dependencies between features as they are solely based on marginal distributions. For example, consider two Boolean features such that half the positives have the value true for both features and the other half have the value false for both, whereas all negatives have opposite values (again distributed half-half over the two possibilities). It follows that each feature in isolation has zero information gain and hence is unlikely to be selected by a feature filter, despite their combination being a perfect classifier. One could say that feature filters are good at picking out possible root features for a decision tree, but not necessarily good at selecting features that are useful further down the tree.

To detect features that are useful in the context of other features we need to evaluate sets of features; this usually goes under the name of *Wrapper* approaches. The idea is that feature selection is 'wrapped' in a search procedure that usually involves training and evaluating a model with a candidate set of features. *Forward selection* methods start with an empty set of features and add features to the set one at a time, as long as they improve the performance of the model. *Backward elimination* starts with the full set of features and aims at improving performance by removing features one at a time. Since there are an exponential number of subsets of features it is usually not feasible to search all possible subsets, and most approaches apply a 'greedy' search algorithm that never reconsiders the choices it makes.

Matrix transformations and decompositions

We can also view feature construction and selection from a geometric perspective, assuming quantitative features. To this end we represent our data set as a matrix **X** with n data points in rows and d features in columns, which we want to transform into a new matrix **W** with n rows and r columns by means of matrix operations. To simplify matters a bit, we assume that **X** is zero-centred and that **W** = **XT** for some d-by-r transformation matrix **T**. For example, feature scaling corresponds to **T** being a d-by-d diagonal matrix; this can be combined with feature selection by removing some of **T**'s columns. A rotation is achieved by **T** being orthogonal, i.e., $\mathbf{TT}^{\mathrm{T}} = \mathbf{I}$. Clearly, several such transformations can be combined (see also Background 1.2 on p.24).

One of the best-known algebraic feature construction methods is *principal component analysis* (*PCA*). Principal components are new features constructed as linear combinations of the given features. The first principal component is given by the direction of maximum variance in the data; the second principal component is the direction of maximum variance orthogonal to the first component, and so on. PCA can be explained in a number of different ways: here, we will derive it by means of the *singular value decomposition (SVD)*. Any n-by-d matrix can be uniquely written as a product of three matrices with special properties:

$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}} \tag{10.2}$$

Here, **U** is an n-by-r matrix, **Σ** is an r-by-r matrix and **V** is an d-by-r matrix (for the moment we will assume $r = d < n$). Furthermore, **U** and **V** are orthogonal (hence rotations) and **Σ** is diagonal (hence a scaling). The columns of **U** and **V** are known as the left and right singular vectors, respectively; and the values on the diagonal of **Σ** are the corresponding singular values. It is customary to order the columns of **V** and **U** so that the singular values are decreasing from top-left to bottom-right.

Now, consider the n-by-r matrix $\mathbf{W} = \mathbf{U}\boldsymbol{\Sigma}$, and notice that $\mathbf{XV} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{U}\boldsymbol{\Sigma} = \mathbf{W}$ by the orthogonality of **V**. In other words, we can construct **W** from **X** by means of the transformation **V**: this is the reformulation of **X** in terms of its principal components. The newly constructed features are found in **UΣ**: the first row is the first principal component, the second row is the second principal component, and so on. These principal components have a geometric interpretation as the directions in which **X** has largest, second-largest, … variance. Assuming the data is zero-centred, these directions can be brought out by a combination of rotation and scaling, which is exactly what PCA does.

We can also use SVD to rewrite the scatter matrix in a standard form:

$$\mathbf{S} = \mathbf{X}^T\mathbf{X} = (\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T)^T(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T) = (\mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^T)(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T) = \mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^T$$

This is known as the *eigendecomposition* of the matrix **S**: the columns of **V** are the eigenvectors of **S**, and the elements on the diagonal of $\boldsymbol{\Sigma}^2$ – which is itself a diagonal matrix – are the eigenvalues. The right singular vectors of the data matrix **X** are the eigenvectors of the scatter matrix $\mathbf{S} = \mathbf{X}^T\mathbf{X}$, and the singular values of **X** are the square root of the eigenvalues of **S**. We can derive a similar expression for the Gram matrix $\mathbf{G} = \mathbf{X}\mathbf{X}^T = \mathbf{U}\boldsymbol{\Sigma}^2\mathbf{U}^T$, from which we see that the eigenvectors of the Gram matrix are the left singular vectors of **X**. This demonstrates that in order to perform principal components analysis it is sufficient to perform an eigendecomposition of the scatter or Gram matrices, rather than a full singular value decomposition.

We have seen something resembling SVD in Section 1.1, where we considered the following matrix product:

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 2 & 2 \\ 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 2 \\ 1 & 0 & 1 & 1 \\ 0 & 2 & 2 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The matrix on the left expresses people's preferences for films (in columns). The right-hand side decomposes or factorises this into film genres: the first matrix quantifies people's appreciation of genres;the last matrix associates films with genres; and the middle matrix tells us the weight of each genre in determining preferences. This is not actually the decomposition computed by SVD, because the left and right matrices in the product are not orthogonal. However, one could argue that this factorisation better captures the data, because the person-by-genre and the film-by-genre matrices are Boolean and sparse, which they won't be in the SVD. The downside is that adding integer or Boolean constraints makes the decomposition problem non-convex (there are local optima) and computationally harder. Matrix decomposition with additional constraints is a very active research area.

These matrix decomposition techniques are often used for dimensionality reduction. The *rank* of an n-by-d matrix is d (assuming $d < n$ and no columns are linear combinations of other columns). The above decompositions are full-rank because $r = d$, and hence the data matrix is reconstructed exactly. A low-rank approximation of a matrix is a factorisation where r is chosen as small as

possible while still sufficiently approximating the original matrix. The reconstruction error is usually measured as the sum of the squared differences of the entries in $\mathbf{X}$ and the corresponding entries in $\mathbf{U\Sigma V}^{\mathrm{T}}$. It can be shown that a truncated singular value decomposition with $r < d$ results in the lowest reconstruction error in this sense among all decompositions of rank up to r . Truncated SVD and PCA are popular ways to combine feature construction and feature selection for quantitative features.

One interesting aspect of matrix decompositions such as SVD is that they expose a previously hidden variable in the data. This can be seen as follows. Consider a decomposition or approximation $\mathbf{U\Sigma V}^{\mathrm{T}}$ with diagonal $\mathbf{\Sigma}$ but not necessarily orthogonal $\mathbf{U}$ and $\mathbf{V}$, and denote the i-the column of $\mathbf{U}$ and $\mathbf{V}$ as $\mathbf{U}_{\cdot i}$ (an n-vector) and $\mathbf{V}_{\cdot i}$ (a d-vector). Then $\mathbf{U}_{\cdot i}\sigma_i(\mathbf{V}_{\cdot i})^{\mathrm{T}}$ is an outer product that produces an n-by-d matrix with rank 1 (σ_i denotes the i-th diagonal value of $\mathbf{\Sigma}$). A rank-1 matrix is such that every column is obtained from a single basis vector multiplied by a scalar (and the same for rows). Assuming $\mathbf{U}$ and $\mathbf{V}$ have rank r these basis vectors are independent and so summing up these rank-1 matrices for all i produces the original matrix:

$$\mathbf{U\Sigma V}^{\mathrm{T}} = \sum_{i=1}^{r} \mathbf{U}_{\cdot i}\sigma_i(\mathbf{V}_{\cdot i})^{\mathrm{T}}$$

For example, the film rating matrix can be written as follows:

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 2 & 2 \\ 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 2 \\ 1 & 0 & 1 & 1 \\ 0 & 2 & 2 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 2 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The matrices on the right can be interpreted as rating models conditioned on genre.

Exposing hidden variables in the data is one of the main applications of matrix decomposition methods. For example, in information retrieval PCA is known under the name *latent semantic indexing* (*LSA*) ('latent' is synonymous with 'hidden'). Instead of film genres, LSA uncovers document topics by decomposing matrices containing word counts per document, under the assumption that the word counts per topic are independent and can thus simply be added up.[3] The other main application of matrix factorisation is *completion* of

missing entries in a matrix, the idea being that if we approximate the observed entries in the matrix as closely as possible using a low-rank decomposition, this allows us to infer the missing entries.

10.4 Features: Summary and further reading

In this chapter we have given features some long-overdue attention. Features are the telescopes through which we observe the data universe and therefore an important unifying force in machine learning. Features are related to measurements in science, but there is no widespread consensus on how to formalise and categorise different measurements – I have taken inspiration from Stevens' scales of measurements (Stevens, 1946), but otherwise aimed to stay close to current practice in machine learning.

- The main kinds of feature distinguished in Section 10.1 are categorical, ordinal and quantitative features. The latter are expressed on a quantitative scale and admit the calculation of the widest range of statistics of tendency (mean, median, mode; see (von Hippel, 2005) for a discussion of rules of thumb regarding these), dispersion (variance and standard deviation, range interquartile range) and shape (skewness and kurtosis). In machine learnin quantitative features are often referred to as continuous features, but I thin this term is inappropriate as it wrongly suggests that their defining feature is somehow an unlimited precision. It is important to realise that quantitative features do not necessarily have an additive scale: e.g., quantitative features expressing a probability are expressed on a multiplicative scale, and the use of Euclidean distance, say, would be inappropriate for non-additive features. Ordinal features have order but no scale; and categorical features (also called nominal or discrete) have neith order nor scale.

- Structured features are first-order logical statements that refer to parts of objects by means of local variables and use some kind of aggregation, suc as existential quantification or counting, to extract a property of the main object. Constructing first-order features prior to learning is often referred t as propositionalisation; Kramer *et al*. (2000) and Lachiche (2010) give surveys, and an experimental comparison of different approaches is carrie out by Krogel *et al*. (2003).

-

In Section 10.2 we looked at a number of feature transformations. Discretisation and thresholding are the best-known of these, turning a quantitative feature into a categorical or a Boolean one. One of the most effective discretisation methods is the recursive partitioning algorithm using information gain to find the thresholds and a stopping criterion derived from the minimum description length principle proposed by Fayy and Irani (1993). Other overviews and proposals are given by Boullé (200 2006). The agglomerative merging approach using χ^2 was proposed by Kerber (1992).

- We have seen that in a two-class setting, supervised discretisation can be visualised by means of coverage curves. This then naturally leads to the idea of using these coverage curves and their convex hull to calibrate rath than just discretise the features. After all, ordinal and quantitative features are univariate rankers and scoring classifiers and thus the same classifier calibration methods can be applied, in particular logistic and isotonic calibration as discussed in Section 7.4. The calibrated features live in probability space, but we might prefer to work with log-odds space instea as this is additive rather than multiplicative. Fitting data to a fixed linear decision boundary in calibrated log-odds space is closely related to trainin a naive Bayes model. Isotonic calibration leads to piecewise axis-parallel decision boundaries;owing to the discretising nature of isotonic calibratio this can be understood as the constructing of a grouping model, even if th original model in the uncalibrated space was a grading model.

- Section 10.3 was devoted to feature construction and selection. Early approaches to feature construction and constructive induction were proposed by Ragavan and Rendell (1993); Donoho and Rendell (1995). T instance-based Relief feature selection method is due to Kira and Rendell (1992) and extended by Robnik-Sikonja and Kononenko (2003). The distinction between filter approaches to feature selection – which evaluate features on their individual merits – and wrapper approaches, which evaluate sets of features, is originally due to Kohavi and John (1997). Hal (1999) proposes a filter approach called correlation-based feature selectio that aims at combining the best of both worlds. Guyon and Elisseeff (200 give an excellent introduction to feature selection.

- Finally, we looked at feature construction and selection from alinear algeb perspective. Matrix decomposition and factorisation is an actively researched technique that was instrumental in winning a recent film

recommendation challenge worth $1 million (Koren *et al.*, 2009). Decomposition techniques employing additional constraints include non-negative matrix decomposition (Lee *et al.*, 1999).

Boolean matrix decomposition is studied by Miettinen (2009). Mahoney and Drineas (2009) describe a matrix decomposition technique that uses actual columns and rows of the data matrix to preserve sparsity (unlike SVD which produces dense matrices even if the original matrix is sparse). Latent semantic indexing and a probabilistic extension is described by Hofmann (1999). Ding and He (2004) discuss the relationship between *K*-means clustering and principal component analysis.

---

[1] If our sample contains an even number of instances, there are two middle values. If the feature has a scale it is customary to take the mean of those two values as the median; if the feature doesn't have a scale, or if it is important that we select a value actually occurring in the sample, we can either select both as the lower and upper median, or we can make a random choice.

[2] Similar to the median there are issues with non-integer ranks, and they can be dealt with in different ways; however, significant differences do not arise unless the sample size is very small.

[3] Other models are possible: e.g., in Boolean matrix decomposition the matrix product is changed to a Boolean product in which integer addition is replaced by Boolean disjunction (so that $1 + 1 = 1$), with the effect that additional topics do not provide additional explanatory power for the occurrence of a word in a document.

CHAPTER *11*

Model ensembles

TWO HEADS ARE BETTER THAN ONE – a well-known proverb suggesting that two minds working together can often achieve better results. If we read 'features' for 'heads' then this is certainly true in machine learning, as we have seen in the preceding chapters. But we can often further improve things by combining not just features but whole models, as will be demonstrated in this chapter. Combinations of models are generally known as *model ensembles*. They are among the most powerful techniques in machine learning, often outperforming other methods. This comes at the cost of increased algorithmic and model complexity.

The topic of model combination has a rich and diverse history, to which we can only partly do justice in this short chapter. The main motivations came from computational learning theory on the one hand, and statistics on the other. It is a well-known statistical intuition that averaging measurements can lead to a more stable and reliable estimate because we reduce the influence of random fluctuations in single measurements. So if we were to build an ensemble of slightly different models from the same training data, we might be able to similarly reduce the influence of random fluctuations in single models. The key question here is how to achieve diversity between these different models. As we shall see, this can often be achieved by training models on random subsets of the data, and even by constructing them from random subsets of the available features.

The motivation from computational learning theory went along the following lines. As we have seen in Section 4.4, learnability of hypothesis languages is studied in the context of a learning model, which determines what we mean by learnability. PAC-learnability requires that a hypothesis be approximately correct most of the time. An alternative learning model called *weak learnability* requires only that a hypothesis is learned that is slightly better than chance. While it appears obvious that PAC-learnability is stricter than weak learnability, it turns out that the two learning models are in fact equivalent: a hypothesis language is PAC-learnable if and only if it is weakly learnable. This was proved constructively by means of an iterative algorithm that repeatedly constructs a

hypothesis aimed at correcting the mistakes of the previous hypothesis, thus 'boosting' it. The final model combined the hypotheses learned in each iteration, and therefore establishes an ensemble.

In essence, ensemble methods in machine learning have the following two things in common:

- they construct multiple, diverse predictive models from adapted versions o the training data (most often reweighted or resampled);
- they combine the predictions of these models in some way, often by simpl averaging or voting (possibly weighted).

It should, however, also be stressed that these commonalities span a very large and diverse space, and that we should correspondingly expect some methods to be practically very different even though superficially similar. For example, it makes a big difference whether the way in which training data is adapted for the next iteration takes the predictions of the previous models into account or not. We will explore this space by means of the two best-known ensemble methods: bagging in Section 11.1 and boosting in Section 11.2. A short discussion of these and related ensemble methods then follows in Section 11.3, before we conclude the chapter in the usual way with a summary and pointers for further reading.

11.1 Bagging and random forests

Bagging, short for 'bootstrap aggregating', is a simple but highly effective ensemble method that creates diverse models on different random samples of the original data set. These samples are taken uniformly with replacement and are known as *bootstrap samples*. Because samples are taken with replacement the bootstrap sample will in general contain duplicates, and hence some of the original data points will be missing even if the bootstrap sample is of the same size as the original data set. This is exactly what we want, as differences between the bootstrap samples will create diversity among the models in the ensemble. To get an idea of how different bootstrap samples might be, we can calculate the probability that a particular data point is not selected for a bootstrap sample of size n as $(1 - 1/n)^n$, which for $n = 5$ is about one-third and has limit $1/e = 0.368$ for $n \to \infty$. This means that each bootstrap sample is likely to leave out about a third of the data points.

Figure 11.1. (left) An ensemble of five basic linear classifiers built from bootstrap samples with bagging. The decision rule is majority vote, leading to a piecewise linear decision boundary. **(right)** If we turn the votes into probabilities, we see the ensemble is effectively a grouping model: each instance space segment obtains a slightly different probability.

Algorithm 11.1 gives the basic bagging algorithm, which returns the ensemble as a set of models. We can choose to combine the predictions from the different models by voting – the class predicted by the majority of models wins – or by averaging, which is more appropriate if the base classifiers output scores or probabilities. An illustration is given in Figure 11.1. I trained five basic linear classifiers on bootstrap samples from 20 positive and 20 negative examples. We can clearly see the diversity of the five linear classifiers, which is helped by the fact that the data set is quite small. The figure demonstrates the difference between combining predictions through voting (Figure 11.1 (left)) and creating a probabilistic classifier by averaging (Figure 11.1 (right)). With voting we see that bagging creates a piecewise linear decision boundary, something that is impossible with a single linear classifier. If we transform the votes from each model into probability estimates, we see that the different decision boundaries partition the instance space into segments that can potentially each receive a different score.

Algorithm 11.1: Bagging(D, T, $\mathscr{A}$) – train an ensemble of models from bootstrap samples.

Input : data set D; ensemble size T; learning algorithm $\mathscr{A}$.
Output : ensemble of models whose predictions are to be combined by voting or averaging.

```
for t = 1 to T do
    build a bootstrap sample D_t from D by sampling |D| data points with
    replacement;
    run 𝒜 on D_t to produce a model M_t;
end
return {M_t | 1 ≤ t ≤ T}
```

Bagging is particularly useful in combination with tree models, which are quite sensitive to variations in the training data. When applied to tree models, bagging is often combined with another idea: to build each tree from a different random subset of the features, a process also referred to as *subspace sampling*. This encourages the diversity in the ensemble even more, and has the additional advantage that the training time of each tree is reduced. The resulting ensemble method is called *random forests*, and the algorithm is given in Algorithm 11.2.

Since a decision tree is a grouping model whose leaves partition the instance space, so is a random forest: its corresponding instance space partition is essentially the intersection of the partitions of the individual trees in the ensemble. While the random forest partition is therefore finer than most tree partitions, it can in principle be mapped back to a single tree model (because intersection corresponds to combining the branches of two different trees). This is different from bagging linear classifiers, where the ensemble has a decision boundary that can't be learned by a single base classifier. One could say, therefore, that the random forest algorithm implements an alternative training algorithm for tree models.

Algorithm 11.2: RandomForest(D, T, d) – train an ensemble of tree models from bootstrap samples and random subspaces.

```
Input  : data set D; ensemble size T; subspace dimension d.
Output : ensemble of tree models whose predictions are to be combined by
         voting or averaging.
for t = 1 to T do
    build a bootstrap sample D_t from D by sampling |D| data points with
    replacement;
    select d features at random and reduce dimensionality of D_t accordingly;
    train a tree model M_t on D_t without pruning;
end
return {M_t | 1 ≤ t ≤ T}
```

11.2 Boosting

Boosting is an ensemble technique that is superficially similar to bagging, but uses a more sophisticated technique than bootstrap sampling to create diverse training sets. The basic idea is simple and appealing. Suppose we train a linear classifier on a data set and find that its training error rate is $\in$. We want to add another classifier to the ensemble that does better on the misclassifications of the first classifier. One way to do that is to duplicate the misclassified instances: if our base model is the basic linear classifier, this will shift the class means towards the duplicated instances. A better way to achieve the same thing is to give the misclassified instances a higher weight, and to modify the classifier to take these weights into account (e.g., the basic linear classifier can calculate the class means as a weighted average).

But how much should the weights change? The idea is that half of the total weight is assigned to the misclassified examples, and the other half to the rest. Since we started with uniform weights that sum to 1, the current weight assigned to the misclassified examples is exactly the error rate $\in$, so we multiply their weights by $1/2\in$. Assuming $\in < 0.5$ this is an increase in weight as desired. The weights of the correctly classified examples get multiplied by $1/2(1 - \in)$, so the adjusted weights again sum to 1. In the next round we do exactly the same, except we take the non-uniform weights into account when evaluating the error rate.

Example 11.1 (Weight updates in boosting). Suppose a linear classifier achieves performance as in the contingency table on the left. The error rate is $\in = (9 + 16)/100 = 0.25$. The weight update for the misclassified examples is a factor $1/2\in = 2$ and for the correctly classified examples $1/2(1 - \in) = 2/3$.

	Predicted ⊕	*Predicted* ⊖	
Actual ⊕	24	16	40
Actual ⊖	9	51	60
	33	67	100

	⊕	⊖	
⊕	16	32	48
⊖	18	34	52
	34	66	100

Taking these updated weights into account leads to the contingency table on the right, which has a (weighted) error rate of 0.5.

We need one more ingredient in our boosting algorithm and that is a confidence factor α for each model in the ensemble, which we will use to form an ensemble prediction that is a weighted average of each individual model. Clearly we want α to increase with decreasing $\in$: a common choice is

$$\alpha_t = \frac{1}{2}\ln\frac{1-\epsilon_t}{\epsilon_t} = \ln\sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \tag{11.1}$$

which we will justify in a moment. The basic boosting algorithm is given in Algorithm 11.3. Figure 11.2 (left) illustrates how a boosted ensemble of five basic linear classifiers can achieve zero training error. It is clear that the resulting decision boundary is much more complex than could be achieved by a single basic linear classifier. In contrast, a bagged ensemble of basic linear classifiers has learned five very similar decision boundaries, the reason being that on this data set the bootstrap samples are all very similar.

I will now justify the particular choice for α_t in Equation 11.1. First, I will show that the two weight updates for the misclassified instances and the correctly classified instances can be written as reciprocal terms δ_t and $1/\delta_t$ normalised by some term Z_t:

$$\frac{1}{2\epsilon_t} = \frac{\delta_t}{Z_t} \qquad\qquad \frac{1}{2(1-\epsilon_t)} = \frac{1/\delta_t}{Z_t}$$

The second expression gives $\delta_t = 2(1 - \in_t)/Z_t$; substituting this back into the first

expression yields

$$Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)} \qquad \delta_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} = \exp(\alpha_t) \tag{11.2}$$

Algorithm 11.3: Boosting(D, T, $\mathscr{A}$) – train an ensemble of binary classifiers from reweighted training sets.

Input : data set D; ensemble size T; learning algorithm $\mathscr{A}$.
Output : weighted ensemble of models.

$w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$; // start with uniform weights
for $t = 1$ to T **do**
  run $\mathscr{A}$ on D with weights w_{ti} to produce a model M_t;
  calculate weighted error ϵ_t;
  if $\epsilon_t \geq 1/2$ **then**
    set $T \leftarrow t-1$ and break
  end
  $\alpha_t \leftarrow \frac{1}{2}\ln\frac{1-\epsilon_t}{\epsilon_t}$; // confidence for this model
  $w_{(t+1)i} \leftarrow \frac{w_{ti}}{2\epsilon_t}$ for misclassified instances $x_i \in D$; // increase weight
  $w_{(t+1)j} \leftarrow \frac{w_{tj}}{2(1-\epsilon_t)}$ for correctly classified instances $x_j \in D$; // decrease weight
end
return $M(x) = \sum_{t=1}^{T} \alpha_t M_t(x)$

Figure 11.2. (left) An ensemble of five boosted basic linear classifiers with majority vote. The linear classifiers were learned from blue to red; none of them achieves zero training error, but the ensemble does. **(right)** Applying bagging results in a much more homogeneous ensemble, indicating that there is little diversity in the bootstrap samples.

So the weight update for misclassified instances is exp (α_t) /Z_t and for correctly

classified instances exp $(-\alpha_t)/Z_t$. Using the fact that $y_i\, M_t(x_i) = +1$ for instances correctly classified by model M_t and -1 otherwise, we can write the weight update as

$$w_{(t+1)i} = w_{ti}\frac{\exp\left(-\alpha_t y_i M_t(x_i)\right)}{Z_t}$$

which is the expression commonly found in the literature.

Let us now step back and pretend that we haven't yet decided what α_t should be in each round. Since the weight updates are multiplicative, we have

$$w_{(T+1)i} = w_{1i}\prod_{t=1}^{T}\frac{\exp\left(-\alpha_t y_i M_t(x_i)\right)}{Z_t} = \frac{1}{|D|}\frac{\exp\left(-y_i M(x_i)\right)}{\prod_{t=1}^{T} Z_t}$$

where $M(x_i) = \sum_{t=1}^{T} \alpha_t\, M_t(x_i)$ is the model represented by the boosted ensemble. The weights always add up to 1 over the instance space, and so

$$\prod_{t=1}^{T} Z_t = \frac{1}{|D|}\sum_{i=1}^{|D|}\exp\left(-y_i M(x_i)\right)$$

Notice that $\exp(-y_i M(x_i))$ is always positive and at least 1 if $-y_i M(x_i)$ is positive, which happens if x_i is misclassified by the ensemble (i.e., $\text{sign}(M(x_i) \neq y_i)$. So the right-hand side of this expression is at least equal to the training error of the boosted ensemble, and $\prod_{t=1}^{T} Z_t$ is an upper bound on that training error. A simple heuristic would therefore be to greedily minimise

$$Z_t = \sum_{i=1}^{|D|} w_{ti}\exp\left(-\alpha_t y_i M_t(x_i)\right) \tag{11.3}$$

in each boosting round. Now, the sum of the weights of instances incorrectly classified by M_t is ϵ_t, and so

$$Z_t = \epsilon_t \exp(\alpha_t) + (1-\epsilon_t)\exp(-\alpha_t)$$

Taking the derivative with respect to α_t, setting it to zero and solving for α_t yields α_t as given in Equation 11.1 and Z_t as given in Equation 11.2.

Notice that Equation 11.3 demonstrates that the loss function minimised by boosting is ☞*exponential loss* exp $(-y\hat{s}(x))$ which we already encountered in Figure 2.6 on p.63. Notice, furthermore, that minimising Z_t means minimising $2\sqrt{\epsilon_t(1-\epsilon_t)}$ according to Equation 11.2. You may recognise this as the ☞ $\sqrt{\text{Gini}}$

impurity measure we investigated in Chapter 5. There, we saw that this splitting criterion is insensitive to changes in the class distribution (see Figure 5.7 on p.146). Here, it arises essentially because of the way weight updates are implemented in the boosting algorithm.

Boosted rule learning

An interesting variant of boosting arises when our base models are partial classifiers that sometimes abstain from making a prediction. For example, suppose that our base classifiers are conjunctive rules whose head is fixed to predicting the positive class. An individual rule therefore either predicts the positive class for those instances it covers, or otherwise abstains from making a prediction. We can use boosting to learn an ensemble of such rules that takes a weighted vote among its members.

We need to make some small adjustments to the boosting equations, as follows. Notice that ϵ_t is the weighted error of the t-th base classifier. Since our rules always predict positive for covered instances, these errors only concern covered negatives, which we will indicate by $\epsilon_t^{\ominus}$. Similarly, we indicate the weighted sum of covered positives as $\epsilon_t^{\oplus}$, which will play the role of $1 - \epsilon_t$. However, with abstaining rules there is a third component, indicated as ϵ_t^0, which is the weighted sum of instances which the rule doesn't cover $(\epsilon_t^0 + \epsilon_t^{\oplus} + \epsilon_t^{\ominus} = 1)$. We then have

$$Z_t = \epsilon_t^0 + \epsilon_t^{\ominus} \exp(\alpha_t) + \epsilon_t^{\oplus} \exp(-\alpha_t)$$

The value of α_t which maximises this is

$$\alpha_t = \frac{1}{2} \ln \frac{\epsilon_t^{\oplus}}{\epsilon_t^{\ominus}} = \ln \sqrt{\frac{\epsilon_t^{\oplus}}{\epsilon_t^{\ominus}}} \tag{11.4}$$

which gives

$$Z_t = \epsilon_t^0 + 2\sqrt{\epsilon_t^{\oplus}\epsilon_t^{\ominus}} = 1 - \epsilon_t^{\oplus} - \epsilon_t^{\ominus} + 2\sqrt{\epsilon_t^{\oplus}\epsilon_t^{\ominus}} = 1 - \left(\sqrt{\epsilon_t^{\oplus}} - \sqrt{\epsilon_t^{\ominus}}\right)^2$$

This means that in each boosting round we construct a rule that maximises $\left|\sqrt{\epsilon_t^{\oplus}} - \sqrt{\epsilon_t^{\ominus}}\right|$ and set its confidence factor to α_t as in Equation 11.4. In order to obtain a prediction from the ensemble, we add up the confidence factors of all rules covering it. Note that these confidence factors are negative if $\epsilon_t^{\oplus} < \epsilon_t^{\ominus}$, which indicates that the rule correlates with the negative class; this is not a problem as such, but can be avoided by changing the objective function for individual rules to $\sqrt{\epsilon_t^{\oplus}} - \sqrt{\epsilon_t^{\ominus}}$.

The weight updates after each iteration of boosting are the same as previously, except that the weights of examples not covered by the rule do not change. Boosted rule learning is therefore similar to the ☞ *weighted covering* (Algorithm 6.5 on p.182) algorithm for subgroup discovery. The difference is that there we wanted to promote rule overlap without reference to the class and hence decrease the weight of all covered examples, whereas here we decrease the weight of covered positives and increase the weight of covered negatives.

11.3 Mapping the ensemble landscape

Now that we have looked at two often-used ensemble methods in somewhat more detail, we consider howtheir differences in performance might be explained, before turning attention to some of themany other ensemble methods in the literature.

Bias, variance and margins

Ensemble methods are a good vehicle to further understand the ☞ *bias–variance dilemma* we discussed in the context of regression in Section 3.2. Broadly speaking, there are three reasons why a model may misclassify a test instance. First, it may simply be unavoidable in the given feature space if instances from different classes are described by the same feature vectors. In a probabilistic context this happens when the per-class distributions P(X|Y) overlap, so that the same instance has non-zero likelihoods for several classes. In such a situation,

the best we can hope to do is to approximate the target concept.

The second reason for classification errors is that the model lacks expressivity to exactly represent the target concept. For example, if the data is not linearly separable then even the best linear classifier will make mistakes. This is the bias of a classifier, and it is inversely related to its expressivity. Although there is no generally agreed way to measure expressivity or bias of a classifier[1] it is intuitively clear that, say, a hyperbolic decision boundary has lower bias than a linear one. It is also clear that tree models have the lowest possible bias, as their leaves can be made arbitrarily small to cover singleton instances.

It may seem that low-bias models are generally preferable. However, a practical rule of thumb in machine learning is that *low-bias models tend to have high variance, and vice versa*. Variance is the third source of classification errors. A model has high variance if its decision boundary is highly dependent on the training data. For example, the nearest-neighbour classifier's instance space segments are determined by a single training point, so if I move a training point in a segment bordering on the decision boundary, that boundary will change. Tree models have high variance for a different reason: if I change the training data sufficiently for another feature to be selected at the root of the tree, then the rest of the tree is likely to be different as well. An example of a low-variance model is the basic linear classifier, because it averages over all the points in a class.

Now look back at Figure 11.1 on p.332. The bagged ensemble of basic linear classifiers has learned a piecewise linear decision boundary that exceeds the expressivity of a single linear classifier. This illustrates that bagging, like any ensemble method, is capable of reducing the bias of a high-bias base model such as a linear classifier. However, if we compare this with boosting in Figure 11.2 on p.336, we see that the reduction in bias resulting from bagging is much smaller than that of boosting. In fact, *bagging is predominantly a variance-reduction technique, while boosting is primarily a bias-reduction technique*. This explains why bagging is often used in combination with high-variance models such as tree models (☞ *random forests* in Algorithm 11.2), whereas boosting is typically used with high-bias models such as linear classifiers or univariate decision trees (also called *decision stumps*).

Another way to understand boosting is in terms of margins. Intuitively, the margin is the signed distance from the decision boundary, with the sign indicating whether we are on the correct or the wrong side. It has been observed in experiments that boosting is effective in increasing the margins of examples, even if they are already on the correct side of the decision boundary. The effect is that boosting may continue to improve performance on the test set even after

the training error has been reduced to zero. Given that boosting was originally conceived in a PAC-learning framework, which is not specifically aimed at increasing margins, this was a surprising result.

Other ensemble methods

There are many other ensemble methods beyond bagging and boosting. The main variation lies in the way predictions of the base models are combined. Notice that this could itself be defined as a learning problem: given the predictions of some base classifiers as features, learn a *metamodel* that best combines their predictions. For example, in boosting we could learn the weights α_t rather than deriving them from each base model's error rate. Learning a linear metamodel is known as *stacking*. Several variations on this theme exist: e.g., decision trees have been used as the metamodel.

It is also possible to combine different base models into a heterogeneous ensemble: in this way the base model diversity derives from the fact that base models are trained by different learning algorithms, and so they can all use the same training set. Some of these base models might employ different settings of a parameter: for example, the ensemble might include several support vector machines with different values of the complexity parameter which regulates the extent to which margin errors are tolerated.

Generally speaking, then, model ensembles consist of a set of base models and a metamodel that is trained to decide how base model predictions should be combined. Implicitly, training a metamodel involves an assessment of the quality of each base model: for instanceif the metamodel is linear as in stacking, a weight close to zero means that the corresponding base classifier does not contribute much to the ensemble. It is even conceivable that a base classifier obtains a negative weight, meaning that in the context of the other base models its predictions are best inverted. We could go one step further and try to *predict* how well a base model is expected to perform, even before we train it! By formulating this as a learning problem at the meta-level, we arrive at the field of meta-learning.

Meta-learning

Meta-learning first involves training a variety of models on a large collection of data sets. The aim is then to construct a model that can help us answer questions

such as the following:

- In which situations is a decision tree likely to outperform a support vector machine?
- When can a linear classifier be expected to perform poorly?
- Can the data be used to give suggestions for setting particular parameters?

The key question in meta-learning is how to design the features on which the metamodel is built. These features should combine data set characteristics and relevant aspects of the trained model. Data set characteristics should go much further than simply listing the number and kind of features and the number of instances, as it is unlikely that anything can be predicted about a model's performance from just that information. For example, we can try to assess the noise level of a data set by measuring the size of a trained decision tree before and after pruning. Training simple models such as decision stumps on a data set and measuring their performance also gives useful information.

In Background 1.1 on p.20 we referred to the no free lunch theorem, which states that no learning algorithm can outperform any other learning algorithm over the set of all possible learning problems. As a corollary, we have that meta-learning over all possible learning problems is futile: if it wasn't, we could build a single hybrid model that uses a metamodel to tell us which base model would achieve better than random performance on a particular data set. It follows that we can only hope to achieve useful meta-learning over non-uniform distributions of learning problems.

11.4 Model ensembles: Summary and further reading

In this short chapter we have discussed some of the fundamental ideas underlying ensemble methods. What all ensemble methods have in common is that they construct several base models from adapted versions of the training data, on top of which some technique is employed to combine the predictions or scores from the base models into a single prediction of the ensemble. We focused on bagging and boosting as two of the most commonly used ensemble methods. A good introduction to model ensembles is given by Brown (2010). The standard reference on classifier combination is Kuncheva (2004) and a more

recent overview is given by Zhou (2012).

- In Section 11.1 we discussed bagging and random forests. Bagging trains diverse models from samples of the training data, and was introduced by Breiman (1996*a*). Random forests, usually attributed to Breiman (2001), combine bagged decision trees with random subspaces; similar ideas were developed by Ho (1995) and Amit and Geman (1997). These techniques a particularly useful to reduce the variance of low-bias models such as tree models.

- Boosting was discussed in Section 11.2. The key idea is to train diverse models by increasing the weight of previously misclassified examples. Th helps to reduce the bias of otherwise stable learners such as linear classifiers or decision stumps. An accessible overview is given by Schapir (2003). Kearns and Valiant (1989, 1994) posed the question whether a wea learning algorithm that performs just slightly better than random guessing can be boosted into an arbitrarily accurate strong learning algorithm. Schapire (1990) introduced a theoretical form of boosting to show the equivalence of weak and strong learnability. The AdaBoost algorithm on which Algorithm 11.3 is based was introduced by Freund and Schapire (1997). Schapire and Singer (1999) give multi-class and multi-label extensions of AdaBoost. A ranking version of AdaBoost was proposed by Freund *et al.* (2003). The boosted rule learning approach that can handle classifiers that may abstain was inspired by Slipper (Cohen and Singer, 1999), a boosted version of Ripper (Cohen, 1995).

- In Section 11.3 we discussed bagging and boosting in terms of bias and variance. Schapire, Freund, Bartlett and Lee (1998) provide a detailed theoretical and experimental analysis of boosting in terms of improving th margin distribution. I also mentioned some other ensemble methods that train a metamodel for combining the base models. Stacking employs a linear metamodel and was introduced by Wolpert (1992) for classification and extended by Breiman (1996*b*) for regression. Meta-decision trees wer introduced by Todorovski and Dzeroski (2003).

- We also briefly discussed meta-learning as a technique for learning about the performance of learning algorithms. The field originated from an early empirical study documented by Michie *et al.* (1994). Recent references ar Brazdil *et al.* (2009, 2010). Unpruned and unpruned decision trees were used to obtain data set characteristics by Peng *et al.* (2002). The idea of

training simple models to obtain further data characteristics is known as landmarking (Pfahringer *et al.*, 2000).

---

[1] While squared loss nicely decomposes into squared bias and variance as shown in Equation 3.2 on p.93, loss functions used in classification such as 0-1 loss can be decomposed in several ways.

CHAPTER *12*

Machine learning experiments

MACHING LEARNING IS a practical subject as much as a computational one. While we may be able to prove that a particular learning algorithm converges to the theoretically optimal model under certain assumptions, we need actual data to investigate, e.g., the extent to which those assumptions are actually satisfied in the domain under consideration, or whether convergence happens quickly enough to be of practical use. We thus evaluate or run particular models or learning algorithms on one or more data sets, obtain a number of measurements and use these to answer particular questions we might be interested in. This broadly characterises what is known as machine learning *experiments*.

In the natural sciences, an experiment can be seen as a question to nature about a scientific theory. For example, Arthur Eddington's famous 1919 experiment to verify Einstein's theory of general relativity asked the question: Are rays of light bent by gravitational fields produced by large celestial objects such as the Sun? To answer this question, the perceived position of stars was recorded under several conditions including a total solar eclipse. Eddington was able to show that these measurements indeed differed to an extent unexplained by Newtonian physics but consistent with general relativity.

While you don't have to travel to the island of Príncipe to perform machine learning experiments, they bear some similarity to experiments in physics in that *machine learning experiments pose questions about models that we try to answer by means of measurements on data.* The following are common examples of the types of question we are interested in:

- How does model m perform on data from domain $\mathscr{D}$?
- Which of these models has the best performance on data from domain $\mathscr{D}$?
- How do models produced by learning algorithm $\mathscr{A}$ perform on data from domain $\mathscr{D}$?
- Which of these learning algorithms gives the best model on data from

domain $\mathscr{D}$?

Assuming we have access to data from domain $\mathscr{D}$, we perform measurements on our models using this data in order to answer these questions.[1] There is a large statistical literature about the technicalities of data experiments, and it is all too easy to mistake the forest for the trees. What I mean is that there is a certain tendency in the machine learning literature to approach experimentation in a formulaic manner, recording a fixed set of measurements and significance tests (the trees) with scant consideration for the question we are aiming to answer (the forest). My aim in this short chapter is not to get buried in technicalities, but rather to give you some appreciation of the importance of choosing measurements that are appropriate for your particular experimental objective (Section 12.1). In Sections 12.2 and 12.3 we take a closer look at how to measure and interpret them.

12.1 What to measure

A good starting point for our measurements are the ☞*evaluation measures* in Table 2.3 on p.57. However, measurements don't have to be scalars: a ROC or coverage curve also counts as a measurement in this context. The appropriateness of any of these for our purposes depends on how we define performance in relation to the question the experiment is designed to answer: let's call it our *experimental objective*. It is important not to confuse performance measures and experimental objectives: the former is something we can measure, while the latter is what we are really interested in. There is often a discrepancy between the two. For example, in psychology our experimental objective may be to quantify a person's intelligence level, and our chosen measurement may be the IQ score achieved on a standardised test – while the IQ score may correlate with intelligence level, it is clear they are not the same thing.

In machine learning the situation is usually more concrete, and our experimental objective – accuracy, say – is something we can measure in principle, or at least estimate (since we're generally interested in accuracy on unseen data). However, there may be unknown factors we have to account for. For example, the model may need to operate in different *operating contexts* with different class distributions. In such a case we can treat accuracy on future data as a random variable and take its expectation, assuming some probability distribution over the proportion of positives *pos*. Since $acc = pos \cdot tpr + (1-pos) \cdot$

tnr, and assuming we can measure true positive and negative rates independently of the class distribution, we have (assuming a uniform distribution over *pos*)

$$\begin{aligned}\mathbb{E}[acc] &= \mathbb{E}[pos \cdot tpr + (1 - pos) \cdot tnr] = \mathbb{E}[pos]\, tpr + \mathbb{E}[1 - pos]\, tnr \\ &= tpr/2 + tnr/2 = avg\text{-}rec\end{aligned}$$

In other words, even though estimating accuracy in future contexts is our experimental objective, the fact that we are anticipating the widest possible range of class distributions suggests that the evaluation measure we should use on our test data is not accuracy, but average recall.

Example 12.1 (Expected accuracy for unknown class distributions). Imagine your classifier achieves the following result on a test data set:

	Predicted ⊕	*Predicted* ⊖	
Actual ⊕	60	20	80
Actual ⊖	0	20	20
	60	40	100

This gives *tpr* = 0.75, *tnr* = 1.00 and *acc* = 0.80. However, this is conditioned on having four times as many positives as negatives. If we take the expectation over *pos* uniformly sampled from the unit interval, expected accuracy increases to (*tpr* + *tnr*)/2 = 0.88 = *avg-rec*. This is higher because the test data underemphasises the classifier's good performance on the negatives.

Suppose you have a second classifier achieving the following result on the test set:

	Predicted ⊕	*Predicted* ⊖	
Actual ⊕	75	5	80
Actual ⊖	10	10	20
	85	15	100

This gives *tpr* = 0.94, *tnr* = 0.50, *acc* = 0.85 and *avg-rec* = 0.72. These experimental results tell you that the second model is better if the class

distribution in the test set is representative, but the first model should be chosen if we have no prior information about the class distribution in the operating context.

As the example demonstrates, if you choose accuracy as your evaluation measure, you are making an implicit assumption that the class distribution in the test set is representative for the operating context in which the model is going to be deployed. Furthermore, if all you recorded in your experiments is accuracy, you will not be able to switch to average recall later if you realise that you need to incorporate varying class distributions. It is therefore good practice to record sufficient information to be able to reproduce the contingency table if needed. A sufficient set of measurements would be true positive rate, true negative rate (or false positive rate), the class distribution and the size of the test set.

As a second example of how your choice of evaluation measures can carry implicit assumptions we consider the case of precision and recall as often reported in the information retrieval literature.

Example 12.2 (Precision and recall as evaluation measures). In the second contingency table in Example 12.1 we have precision *prec* = 75/85 = 0.88 and recall *rec* = 75/80 = 0.94 (remember that recall and true positive rate are different names for the same measure). The F-measure is the harmonic mean of precision and recall (see Background 10.1), which is 0.91.

Now consider the following contingency table:

	Predicted ⊕	*Predicted* ⊖	
Actual ⊕	75	5	80
Actual ⊖	10	910	920
	85	915	1000

We have a much higher number of true negatives and therefore a much higher true negative rate and accuracy (both rounded to 0.99). On the other hand, true positive rate/recall, precision and F-measure stay exactly the same.

This example demonstrates that *the combination of precision and recall, and therefore the F-measure, is insensitive to the number of true negatives*. This is not a deficiency of the F-measure: quite the contrary, it is very useful in domains where negatives abound, and it would therefore be very easy to achieve high accuracy by always predicting negative. Examples of such domains include search and query engines (most search items are not answers to most queries) and link prediction in networks (most pairs of nodes are not linked). The point is rather to emphasise that if you choose F-measure as your evaluation measure, you are making an implicit assertion that true negatives are not relevant for your operating context.

Finally, I would like to draw attention to a much-neglected evaluation measure that has, nevertheless, practical significance. This is the *predicted positive rate* which is the number of positive predictions (the sum of the left column in the contingency table) in proportion to the number of instances:

$$ppr = \frac{TP + FP}{Pos + Neg} = pos \cdot tpr + (1 - pos) \cdot fpr$$

While the predicted positive rate doesn't tell us much about the classification performance of the classifier, it does tell us what the classifier estimates the class distribution to be. It is also something that is normally under control of a ranker or scoring classifier if the entire test set is given: e.g., a predicted positive rate of 1/2 is simply achieved by setting the threshold such that the ranking splits into equal parts. This suggests a connection between classification accuracy and ranking accuracy: for example, it can be shown that if we split a ranking of n instances at one of the n+1 possible split points chosen uniformly at random, the expected accuracy is equal to

$$\mathbb{E}[acc] = \frac{n}{n+1}\frac{2\mathrm{AUC} - 1}{4} + 1/2$$

Example 12.3 (Expected accuracy and AUC). Suppose a ranker obtains the following ranking on a small test set: ⊕⊕⊖⊖⊕⊖. This corresponds to two ranking errors out of a possible nine, so has AUC = 7/9. There are seven potential split points, corresponding to predicted positive rates of (from left

to right) 0,1/6,…,5/6,1 and corresponding accuracies 3/6,4/6,5/6,4/6,3/6,4/6,3/6. The expected accuracy over all possible split points is (3+4+5+4+3+4+3)/(6 · 7) = 26/42. On the other hand, (2AUC–1)/4 = 5/36 and so $\frac{n}{n+1}$(2AUC–1)/4+1/2 = 5/42+1/2 = 26/42.

This discussion is intended to highlight two things. First, if a model is a good ranker but its probability estimates are not well-calibrated, it might be a good idea to set the decision threshold such that it achieves a particular predicted positive rate (e.g., *ppr* = *pos*) rather than invoking the MAP decision rule. Secondly, in such situations AUC is a good evaluation measure because it is linearly related to expected accuracy in that scenario.

In summary, your choice of evaluation measures should reflect the assumptions you are making about your experimental objective as well as possible contexts in which your models operate. We have looked at the following cases:

- Accuracy is a good evaluation measure if the class distribution in your tes set is representative for the operating context.
- Average recall is the evaluation measure of choice if all class distributions are equally likely.
- Precision and recall shift the focus from classification accuracy to a performance analysis ignoring the true negatives.
- Predicted positive rate and AUC are relevant measures in a ranking contex

In the next section we consider how to estimate these evaluation measures from data.

12.2 How to measure it

The evaluation measures we discussed in the previous section are all calculated from a contingency table. The question of 'how to measure it' thus seems to have a very straightforward answer: construct the contingency table from a test set and perform the relevant calculations. However, two issues demand our

attention: (*i*) which data to base our measurements on, and (*ii*) how to assess the inevitable uncertainty associated with every measurement. In this section we are concerned with the first issue; the second issue will be discussed in the next section.

When we measure something – say, a person's height – several times, we expect some variation to occur from one measurement to the next. This is inherent to the measurement process: e.g., you may be stretching the tape measure a bit less for the second measurement, or you may be reading it from a slightly different angle.[2] This variation can be modelled by treating our measurement as a random variable characterised by its mean – the value we are trying to measure – and variance σ^2, both of which are unknown but can be estimated. It follows from this model that if you measure a person's height many times, the sample variance in the measured values converges to σ^2. A standard trick is to average k measurements, as this gets the variance in your estimate down to σ^2/k. What this means is that, if you repeated this averaging for many sets of k measurements, the averages have a sample variance σ^2/k. Crucially, this assumes that your measurements are independent: if you are introducing a systematic error by using a faulty tape measure, averaging won't help!

Now suppose you are measuring a classifier's accuracy (or its true positive rate, or the predicted positive rate, or any other evaluation measure discussed earlier), rather than a person's height. The natural model here is that each test instance represents a Bernoulli trial with success probability a, the true but unknown accuracy of the classifier. We estimate a by counting the number of correctly classified test instances A and setting $\hat{a} = A/n$; notice that A has a binomial distribution. The variance of a single Bernoulli trial is $a(1 - a)$; averaged over n test instances it is $a(1 - a)/n$, assuming the test instances are chosen independently. We can estimate the variance by plugging in our estimate for a: this will help us to assess the uncertainty in $\hat{a}$, as we shall see in the next section. Under certain conditions we can improve our estimate by averaging k independent estimates $\hat{a}_i$ and take their sample variance $\frac{1}{k-1}\sum_{i=1}^{k}(\hat{a}_i - \overline{a})^2$ instead, with $\overline{a} = \frac{1}{k}\sum_{i=1}^{k}\hat{a}_i$ the sample mean.[3]

How do we obtain k independent estimates of a? If we have plenty of data, we can sample k independent test sets of size n and estimate a on each of them. Notice that if we are evaluating a learning algorithm rather than a given model we need to set aside training data which needs to be separate from the test data. If we don't have a lot of data, the following *cross-validation* procedure is often applied: randomly partition the data in k parts or 'folds', set one fold aside for

testing, train a model on the remaining $k - 1$ folds and evaluate it on the test fold. This process is repeated k times until each fold has been used for testing once. This may seem curious at first since we are evaluating k models rather than a single one, but this makes sense if we are evaluating a learning algorithm (whose output is a model, so we want to average over models) rather than a single model (whose outputs are instance labels, so we want to average over those). By averaging over training sets we get a sense of the variance of the learning algorithm (i.e., its dependence on variations in the training data), although it should be noted that the training sets in cross-validation have considerable overlap and are clearly not independent. Once we are satisfied with the performance of our learning algorithm, we can run it over the entire data set to obtain a single model.

Cross-validation is conventionally applied with $k = 10$, although this is somewhat arbitrary. A rule of thumb is that individual folds should contain at least 30 instances, as this allows us to approximate the binomial distribution of the number of correctly classified instances in a fold by a normal distribution. So if we have fewer than 300 instances we need to adjust k accordingly. Alternatively, we can set $k = n$ and train on all but one test instance, repeated n times: this is known as *leave-one-out* cross-validation (or the jackknife in statistics). This means that in each single-instance 'fold' our accuracy estimate is 0 or 1, but by averaging n of those we get an approximately normal distribution by the central limit theorem. If we expect the learning algorithm to be sensitive to the class distribution we should apply *stratified cross-validation*: this aims at achieving roughly the same class distribution in each fold. Cross-validation runs can be repeated for different random partitions into folds and the results averaged again to further reduce variance in our estimates: this is referred to as, e.g., 10 times 10-fold cross-validation. It should be kept in mind that this leads increasingly to independence assumptions being violated – if we take this too far our accuracy estimate will overfit the given data and not be representative for new data.

Example 12.4 (Cross-validation). The following table gives a possible result of evaluating three learning algorithms on a data set with 10-fold cross-validation:

Fold	*Naive Bayes*	*Decision tree*	*Nearest neighbour*
1	0.6809	0.7524	0.7164
2	0.7017	0.8964	0.8883
3	0.7012	0.6803	0.8410
4	0.6913	0.9102	0.6825
5	0.6333	0.7758	0.7599
6	0.6415	0.8154	0.8479
7	0.7216	0.6224	0.7012
8	0.7214	0.7585	0.4959
9	0.6578	0.9380	0.9279
10	0.7865	0.7524	0.7455
avg	0.6937	0.7902	0.7606
stdev	0.0448	0.1014	0.1248

The last two lines give the average and standard deviation over all ten folds. We can see that nearest neighbour has the highest standard deviation. Clearly the decision tree achieves the best result, but should we completely discard nearest neighbour?

Cross-validation can also be applied to ROC curves obtained from a scoring classifier. This is because every instance participates in exactly one test fold and receives a score from the corresponding model. We can therefore simply merge all test folds which produces a single ranking.

12.3 How to interpret it

Once we have estimates of a relevant evaluation measure for our models or learning algorithms we can use them to select the best one. The fundamental issue here is how to deal with the inherent uncertainty in these estimates. We will discuss two key concepts: confidence intervals and significance tests. An understanding of these concepts is necessary if you want to appreciate current practice in interpreting results from machine learning experiments – however, it is good to realise that current practice is coming under increasing scrutiny. It should also be noted that the methods described here represent only a tiny fraction of the vast spectrum of possibilities.

Suppose our estimate $\hat{a}$ follows a normal distribution around the true mean a with standard deviation σ. Assuming for the moment that we know these parameters, we can calculate for any interval the likelihood of the estimate falling in the interval, by calculating the area under the normal density function in that interval. For example, the likelihood of obtaining an estimate within ±1 standard deviation around the mean is 68%. Thus, if we take 100 estimates from independent test sets, we expect 68 of them to be within one standard deviation on either side of the mean – or equivalently, we expect the true mean to fall within one standard deviation on either side of the estimate in 68 cases. This is called the 68% *confidence interval* of the estimate. For two standard deviations the confidence level is 95% – these values can be looked up in probability tables or calculated using statistical packages such as Matlab or R. Notice that confidence intervals for normally distributed estimates are symmetric because the normal distribution is symmetric, but this is not generally the case: e.g., the binomial distribution is asymmetric (except for $p = 1/2$). Notice also that, in case of symmetry, we can easily change the interval into a one-sided interval: for example, we expect the mean to be more than one standard deviation *above* the estimate in 16 cases out of 100, which gives a one-sided 84% confidence interval from minus infinity to the mean plus one standard deviation.

More generally, in order to construct confidence intervals we need to know (*i*) the sampling distribution of the estimates, and (*ii*) the parameters of that distribution. We saw previously that accuracy estimated from a single test set with n instances follows a scaled binomial distribution with variance $\hat{a}(1 - \hat{a})/n$. This would lead to asymmetric confidence intervals, but the skew in the binomial distribution is only really noticeable if $na(1 - a) < 5$: if that is not the case the normal distribution is a good approximation for the binomial one. So we use the binomial expression for the variance and use the normal distribution to construct the confidence intervals.

Example 12.5 (Confidence interval). Suppose 80 out of 100 test instances are correctly classified. We have $\hat{a} = 0.80$ with an estimated variance of $\hat{a}(1 - \hat{a})/n = 0.0016$ or a standard deviation of $\sqrt{\hat{a}(1-\hat{a})/n} = 0.04$. Notice $n\hat{a}(1 - \hat{a}) = 16 \geq 5$ so the 68% confidence interval is estimated as [0.76, 0.84] in accordance with the normal distribution, and the 95% interval is [0.72, 0.88].

If we reduce the size of our test sample to 50 and find that 40 test

instances are correctly classified, then the standard deviation increases to 0.06 and the 95% confidence interval widens to [0.68, 0.92]. If the test sample drops to less than 30 instances we would need to construct an asymmetric confidence interval using tables for the binomial distribution.

Notice that *confidence intervals are statements about estimates rather than statements about the true value of the evaluation measure*. The statement 'assuming the true accuracy a is 0.80, the probability that a measurement m falls in the interval [0.72, 0.88] is 0.95' is correct, but we cannot reverse this to say 'assuming a measurement $m = 0.80$, the probability that the true accuracy falls in the interval [0.72, 0.88] is 0.95'. To infer $P(a \in [0.72, 0.88] \mid m = 0.80)$ from $P(m \in [0.72, 0.88] \mid a = 0.80)$ we must somehow invoke Bayes' rule, but this requires meaningful prior distributions over both true accuracies and measurements, which we don't generally have.

We can, however, use similar reasoning to test a particular *null hypothesis* we have about a. For example, suppose our null hypothesis is that the true accuracy is 0.5 and that the standard deviation derived from the binomial distribution is therefore $\sqrt{0.5(1-0.5)/100} = 0.05$. Given our estimate of 0.80, we then calculate the *p-value*, which is the probability of obtaining a measurement of 0.80 or higher given the null hypothesis. The p-value is then compared with a pre-defined significance level, say $\alpha = 0.05$: this corresponds to a confidence of 95%. The null hypothesis is rejected if the p-value is smaller than α; in our case this applies since $p = 1.9732 \cdot 10^{-9}$.

This idea of *significance testing* can be extended to learning algorithms evaluated in cross-validation. For a pair of algorithms we calculate the difference in accuracy on each fold. The difference between two normally distributed variables is also normally distributed. Our null hypothesis is that the true difference is 0, so that any differences in performance are attributed to chance. We calculate a p-value using the normal distribution, and reject the null hypothesis if the p-value is below our significance level α. The one complication is that we don't have access to the true standard deviation in the differences, which therefore needs to be estimated. This introduces additional uncertainty into the process, which means that the sampling distribution is bell-shaped like the normal distribution but slightly more heavy-tailed. This distribution is referred to as Student's t -distribution or simply the *t-distribution*.[4] The extent to

which the t -distribution is more heavy-tailed than the normal distribution is regulated by the number of *degrees of freedom*: in our case this is equal to 1 less than the number of folds (since the final fold is completely determined by the other ones). The whole procedure is known as the *paired t -test*.

Example 12.6 (Paired t -test). The following table demonstrates the calculation of a paired t -test on the results in Example 12.4. The numbers show pairwise differences in each fold. The null hypothesis in each case is that the differences come from a normal distribution with mean 0 and unknown standard deviation.

Fold	*NB–DT*	*NB–NN*	*DT–NN*
1	-0.0715	-0.0355	0.0361
2	-0.1947	-0.1866	0.0081
3	0.0209	-0.1398	-0.1607
4	-0.2189	0.0088	0.2277
5	-0.1424	-0.1265	0.0159
6	-0.1739	-0.2065	-0.0325
7	0.0992	0.0204	-0.0788
8	-0.0371	0.2255	0.2626
9	-0.2802	-0.2700	0.0102
10	0.0341	0.0410	0.0069
avg	-0.0965	-0.0669	0.0295
stdev	0.1246	0.1473	0.1278
p-value	**0.0369**	**0.1848**	**0.4833**

The p-value in the last line of the table is calculated by means of the t -distribution with $k - 1 = 9$ degrees of freedom, and only the difference between the naive Bayes and decision tree algorithms is found significant at the $\alpha = 0.05$ level.

Interpretation of results over multiple data sets

The *t*-test can be applied for comparing two learning algorithms over a single data set, typically using results obtained in cross-validation. It is not appropriate for multiple data sets because performance measures cannot be compared across data sets (they are not 'commensurate'). In order to compare two learning algorithms over multiple data sets we need to use a test specifically designed for that purpose such as *Wilcoxon's signed-rank test*. The idea is to rank the performance differences in absolute value, from smallest (rank 1) to largest (rank n). We then calculate the sum of ranks for positive and negative differences separately, and take the smaller of these sums as our test statistic. For a large number of data sets (at least 25) this statistic can be converted to one which is approximately normally distributed, but for smaller numbers the *critical value* (the value of the statistic at which the p-value equals α) can be found in a statistical table.

Example 12.7 (Wilcoxon's signed-rank test). We use the performance differences between naive Bayes and decision tree as in the previous example, but now assume for the sake of argument that they come from 10 different data sets.

Data set	*NB-DT*	*Rank*
1	-0.0715	**4**
2	-0.1947	**8**
3	0.0209	**1**
4	-0.2189	**9**
5	-0.1424	**6**
6	-0.1739	**7**
7	0.0992	**5**
8	-0.0371	**3**
9	-0.2802	**10**
10	0.0341	**2**

The sum of ranks for positive differences is 1+5+2 = 8 and for negative differences 4+8+9+6+7+3+10 = 47. The critical value for 10 data sets at the α = 0.05 level is 8, which means that if the smallest of the two sums of ranks is less than or equal to 8 the null hypothesis that the ranks are

distributed the same for positive and negative differences can be rejected. This applies in this case, so we conclude that the performance difference between naive Bayes and decision trees is significant according to Wilcoxon's signed-rank test (as it was for the paired t -test in Example 12.6).

The Wilcoxon signed-rank test assumes that larger performance differences are better than smaller ones, but otherwise makes no assumption about their commensurability – in other words, performance differences are treated as ordinals rather than real-valued. Furthermore, there is no normality assumption regarding the distribution of these differences[5] which means, among other things, that the test is less sensitive to outliers.

If we want to compare k algorithms over n data sets we need to use specialised significance tests to avoid that our confidence level drops with each additional pairwise comparison between algorithms. The *Friedman test* is designed for exactly this situation. Like the Wilcoxon signed-rank test it is based on ranked rather than absolute performance, and hence makes no assumption regarding the distribution of the performance measurements.[6] The idea is to rank the performance of all k algorithms per data set, from best performance (rank 1) to worst performance (rank k). Let R_{ij} denote therankofthe j-th algorithm on the i-th data set, and let $R_j = \left(\sum_i R_{ij}\right)/n$ be the average rank of the j-th algorithm. Under the null hypothesis that all algorithms perform equally these average ranks R_j should be the same. In order to test this we calculate the following quantities:

1. the average rank $\overline{R} = \frac{1}{nk}\sum_{ij} R_{ij} = \frac{k+1}{2}$;
2. the sum of squared differences $n\sum_j (R_j - \overline{R})^2$; and
3. the sum of squared differences $\frac{1}{n(k-1)}\sum_{ij}(R_{ij} - \overline{R})^2$.

There is an analogy with clustering here, in that the second quantity measures the spread between the rank 'centroids' – which we want to be large – and the third quantity measures the spread over all ranks. The Friedman statistic is the ratio of the former and latter quantities.

Example 12.8 (Friedman test). We use the data from Example 12.4, assuming it comes from different data sets rather than cross-validation folds. The following table shows the ranks in brackets:

Data set	*Naive Bayes*	*Decision tree*	*Nearest neighbour*
1	0.6809 (3)	0.7524 (1)	0.7164 (2)
2	0.7017 (3)	0.8964 (1)	0.8883 (2)
3	0.7012 (2)	0.6803 (3)	0.8410 (1)
4	0.6913 (2)	0.9102 (1)	0.6825 (3)
5	0.6333 (3)	0.7758 (1)	0.7599 (2)
6	0.6415 (3)	0.8154 (2)	0.8479 (1)
7	0.7216 (1)	0.6224 (3)	0.7012 (2)
8	0.7214 (2)	0.7585 (1)	0.4959 (3)
9	0.6578 (3)	0.9380 (1)	0.9279 (2)
10	0.7865 (1)	0.7524 (2)	0.7455 (3)
avg rank	2.3	1.6	2.1

We have $\overline{R} = 2$, $n\Sigma_j(R_j - \overline{R})^2 = 2.6$ and $\frac{1}{n(k-1)}\Sigma_{ij}(R_{ij} - \overline{R})^2 = 1$, so the Friedman statistic is 2.6. The critical value for $k = 3$ and $n = 10$ at the $\alpha = 0.05$ level is 7.8, so we cannot reject the null hypothesis that all algorithms perform equally. In comparison, if the average ranks were 2.7, 1.3 and 2.0, then the null hypothesis would be rejected at that significance level.

The Friedman test tells us whether the average ranks as a whole display significant differences, but further analysis is needed on a pairwise level. This is achieved by applying a *post-hoc test* once the Friedman test gives significance. The idea is to calculate the *critical difference* (*CD*) against which the difference in average rank between two algorithms is compared. The *Nemenyi test* calculates the critical difference as follows:

$$\mathrm{CD} = q_\alpha \sqrt{\frac{k(k+1)}{6n}} \tag{12.1}$$

where q_α depends on the significance level α as well as k: for $\alpha = 0.05$ and $k = 3$ it is 2.343, leading to a critical difference of 1.047 in our example. If the average ranks are 2.7, 1.3 and 2.0, then only the difference between the first and second algorithm exceeds the critical difference. Figure 12.1 (top) demonstrates a useful visual representation of the results of the Nemenyi post-hoc test.

Figure 12.1. (top) Critical difference diagram for the pairwise Nemenyi test. Average ranks for each algorithm are plotted on the real axis. The critical difference is shown as a bar above the figure, and any group of consecutively ranked algorithms such that the outermost ones are less than the critical difference apart are connected by a horizontal thick line. The diagram shows, e.g., that the performance of the top ranked algorithm is significantly better than the bottom three. **(bottom)** Critical difference diagram for the Bonferroni–Dunn test with **CN2** as control. The critical differences are now drawn symmetrically around the average rank of the control. The top ranked algorithm is significantly better than the control, and the bottom ranked one is significantly worse. (Figures courtesy of Tarek Abudawood (2011)).

A variant of the Nemenyi test called the *Bonferroni–Dunn test* can be applied when we perform pairwise tests only against a control algorithm. The calculation of the critical difference is the same, except q_α is adjusted to reflect the fact that we make $k - 1$ pairwise comparisons rather than $k(k - 1)/2$. For example, for $\alpha = 0.05$ and $k = 3$ we have $q_\alpha = 2.241$, which is slightly lower than the value used for the Nemenyi test, leading to a tighter critical difference. Figure 12.1 (bottom) shows a graphical representation of the Bonferroni–Dunn post-hoc test.

12.4 Machine learning experiments: Summary and further reading

In this chapter we have taken a look at how we can use data to answer questions about the performance of models and learning algorithms. A 'machine learning experimenter' needs to address three questions: (*i*) what to measure, (*ii*) how to measure it, and (*iii*) how to interpret it. An excellent source – particularly for the last two questions – is Japkowicz and Shah (2011).

- In order to decide what to measure, we first need to explicate our experimental objective. We also need to consider the operating context: performance aspects that might change when using the model. For exampl the operating context might be given by the class distribution, but we may have no prior knowledge telling us that certain distributions are more likel than others. Example 12.1 on p.345 demonstrated that in such a case average recall would be more appropriate as a performance measure, even if the experimental objective is accuracy. We also looked at the difference between a precision–recall analysis which ignores the true negatives, and a true/false positive rate analysis which takes them into account; a fuller analysis is provided by Davis and Goadrich (2006). The relation between accuracy as experimental objective and AUC as performance measure is studied by Hernández-Orallo *et al.* (2011).

- Once we decided what to measure, we need to establish a measuring protocol. The most common approach is *k*-fold cross-validation, which divides the data into *k* folds, repeatedly trains on *k*–1 of those and tests on the remaining one. It is of paramount importance that there be no information leak between the training data used to learn the model and the test data used to evaluate it. A common mistake is to use cross-validation t find the best setting of one or more parameters of a learning algorithm, say the complexity parameter of a support vector machine. This is methodologically wrong as parameter tuning should be carried out as part of the training process, without any access to the test data. A methodologically sound option is to use *internal cross-validation* by settin aside a validation fold in each cross-validation run for parameter tuning. Experimental studies regarding cross-validation are carried out by Dietterich (1998) and Bouckaert and Frank (2004): the former recommenc five times two-fold cross-validation and the latter ten times ten-fold. ROC curves can be drawn in cross-validation as each instance appears in a test fold exactly once, and so we can collect the scores on all test folds. Fawce (2006) considers alternatives including horizontal and vertical averaging.

- In the context of interpreting experimental results we looked at confidence

intervals and significance tests. Confidence intervals have a clear statistica interpretation: they quantify the likelihood of a measurement falling in a particular interval, assuming a particular true value. Significance tests extend this to reasoning about a particular null hypothesis, such as 'these learning algorithms do not perform differently on these data sets'. Significance tests are designed for particular protocols: the *t* -test can be used for evaluating two learning algorithms on two data sets, Wilcoxon's signed-rank test is applicable for comparing two algorithms over multiple data sets, and Friedman's test (or analysis of variance) compares multiple algorithms over multiple data sets. An excellent discussion of these and related tests is provided by Demšar (2006).

☞ It should be mentioned that there is much discussion on the use of significance tests in machine learning, and on the wider issue regarding machine learning as an experimental science. The importance of experiments in machine learning was stressed early on by Pat Langley in two influential papers (Langley, 1988; Kibler and Langley, 1988); howeve more recently he expressed criticism at the way experimental methodolog in machine learning has become rather inflexible (Langley, 2011). Other authors critical of current practice include Drummond (2006) and Demšar (2008).

---

[1]It is much harder to answer questions about a new domain given measurements on different domains, although that is what we are ultimately interested in.

[2]Some variation is also due to the fact that somebody tends to be taller fresh out of bed in the morning than at the end of a day of sitting and standing up, but I ignore that here and assume an unambiguous true value that we are trying to measure.

[3]Notice that we divide by $k - 1$ rather than k in the expression for the sample variance, to account for the uncertainty in our estimate of the sample mean.

[4]It was published by William Sealy Gosset in 1908 under the pseudonym 'Student' because his employer, the Guinness brewery in Dublin, did not want the competition to know that they were using statistics.

[5]In statistical terminology the test is 'non-parametric' as opposed to a parametric test such as the *t*-test which assumes a particular distribution. Parametric tests are generally more powerful when that assumed distribution is appropriate but can be misleading when it is not.

[6]A well-known parametric alternative to the Friedman test is *analysis of variance* (ANOVA).

A Epilogue: Where to go from here

ND SO WE HAVE come to the end of our journey through the 'making sense of data' landscape. We have seen how machine learning can build models from features for solving tasks involving data. We have seen how models can be predictive or descriptive; learning can be supervised or unsupervised; and models can be logical, geometric, probabilistic or ensembles of such models. Now that I have equipped you with the basic concepts to understand the literature, there is a whole world out there for you to explore. So it is only natural for me to leave you with a few pointers to areas you may want to learn about next.

One thing that we have often assumed in the book is that the data comes in a form suitable for the task at hand. For example, if the task is to label e-mails we conveniently learn a classifier from data in the form of labelled e-mails. For tasks such as class probability estimation I introduced the output space (for the model) as separate from the label space (for the data) because the model outputs (class probability estimates) are not directly observable in the data and have to be reconstructed. An area where the distinction between data and model output is much more pronounced is *reinforcement learning*. Imagine you want to learn how to be a good chess player. This could be viewed as a classification task, but then you require a teacher to score every move. What happens in practical situations is that every now and then you receive a reward or a punishment – e.g., winning the game, or losing one of your pieces. The challenge is then to assign credit or blame to individual moves that led to such rewards or punishments being incurred. Reinforcement learning is a principled way to learn policies for deciding which action to take in which situation or state. This is currently one of the most active subfields of machine learning. The standard reference is Sutton and Barto (1998), and you should have no trouble finding more recent workshop proceedings or journal special issues.

There are many other tasks that require us to relax some of our assumptions. For example, in multi-class classification we assume that classes are mutually exclusive. In *multi-label classification* we drop that assumption, so that an instance can be labelled with an arbitrary subset of labels. This is natural, e.g., when tagging online material such as blog posts. The dependence between labels is an additional source of information: for example, knowing that the tag 'machine learning' applies makes the tag 'reinforcement learning' more likely. Multi-label learning aims to exploit this information by learning the dependence between the labels as well as the mapping between the features and each individual label. For relevant work in the area see, e.g., Tsoumakas *et al.* (2012). A related area is *preference learning*, where the goal is to learn instance-dependent preferences between class labels (Fürnkranz and Hüllermeier, 2010). Increasing the complexity of the model outputs even further, we arrive at the general area of *structured output prediction* (Bakir *et al.*, 2007).

Going back to multi-label learning, although each label establishes a separate binary classification task, the goal is to avoid learning completely separate models for each task. This is, in fact, a special case of what is called *multi-task learning*. For example, each task could be to predict a separate real-valued target variable on the same instance space, and the learner is aiming to exploit, say, correlations between the target variables. Closely related to this is the area of *transfer learning*, which studies the transfer of models between tasks. A relevant

reference for both areas is Silver and Bennett (2008).

Another assumption that deserves closer scrutiny is the availability of data in a single batch. In *online learning*, also called incremental learning, the model needs to be updated each time a new data point arrives. One application of this is in the area of *sequence prediction* (Cesa-Bianchi and Lugosi, 2006). With the increase in sensor data this setting is rapidly gaining importance, as can be witnessed from the growing area of *learning from data streams* (Gama and Gaber, 2007). Sometimes it is convenient to give the learner a more active role in data acquisition, for example by issuing queries for examples to be labelled by the teacher. *Active learning* studies exactly this setting (Settles, 2011).

Ultimately, machine learning is – and, in all likelihood, will remain – a research area at the nexus of two distinct developments. On the one hand, it is widely recognised that the ability for learning and self-training is necessary for achieving machine intelligence in any form. An area in machine learning that has this quest at heart is *deep learning*, which aims at employing hierarchies of autonomously constructed features (Bengio, 2009). On the other hand, machine learning is an indispensable tool for dealing with the data deluge. Building machine learning models is an essential step in the *data mining* process, which poses specific challenges such as being able to deal with 'big data' and cloud computing platforms. I hope that this book has kindled your interest in one of these exciting developments.

Important points to remember

Machine learning is the systematic study of algorithms and systems that improve their knowledge or performance with experience.
3

Tasks are addressed by models, whereas learning problems are solved by learning algorithms that produce models.
12

Machine learning is concerned with using the right features to build the right models that achieve the right tasks.
12

Models lend the machine learning field diversity, but tasks and features give it unity.
13

Use likelihoods if you want to ignore the prior distribution or assume it uniform, and posterior probabilities otherwise.
28

Everything should be made as simple as possible, but not simpler.
30

In a coverage plot, classifiers with the same accuracy are connected by line segments with slope 1.
59

In a normalised coverage plot, line segments with slope 1 connect

classifiers with the same average recall.
60

The area under the ROC curve is the ranking accuracy.
67

Grouping model ROC curves have as many line segments as there are instance space segments in the model;grading models have one line segment for each example in the data set.
69

By decreasing a model's refinement we sometimes achieve better ranking performance.
69

Concavities in ROC curves can be remedied by combining segments through tied scores.
77

To avoid overfitting, the number of parameters estimated from the data must be considerably less than the number of data points.
93

In descriptive learning the task and learning problem coincide.
95

The LGG is the most conservative generalisation that we can learn from the data.
108

Every concept between the least general one and one of the most general ones is also a possible hypothesis.
112

An upward path through the hypothesis space corresponds to a coverage curve.
114

Decision trees are strictly more expressive than conjunctive concepts.
131

One way to avoid overfitting and encourage learning is to deliberately choose a restrictive hypothesis language.
131

The ranking obtained from the empirical probabilities in the leaves of a decision tree yields a convex ROC curve on the training data.

138

Entropy and Gini index are sensitive to fluctuations in the class distribution, $\sqrt{\text{Gini}}$ isn't.

147

Rule lists are similar to decision trees in that the empirical probabilities associated with each rule yield convex ROC and coverage curves on the training data.

166

$(X^TX)^{-1}$ acts as a transformation that decorrelates, centres and normalises the features.

202

Assuming uncorrelated features effectively decomposes a multivariate regression problem into *d* univariate problems.

203

A general way of constructing a linear classifier with decision boundary $\mathrm{w} \cdot \mathrm{x} = t$ is by constructing **w** as $M^{-1}(n^{\oplus}\mu^{\oplus} - n^{\ominus}\mu^{\ominus})$.

207

In the dual, instance-based view of linear classification we are learning instance weights α_i rather than feature weights w_j.

210

A minimal-complexity soft margin classifier summarises the classes by their class means in a way very similar to the basic linear classifier.

219

The basic linear classifier can be interpreted from a distance-based perspective as constructing exemplars that minimise squared Euclidean distance within each class, and then applying a nearest-exemplar decision rule.

239

Probabilities do not have to be interpreted as estimates of relative frequencies, but can carry the more general meaning of (possibly subjective) degrees of belief.

265

For uncorrelated, unit-variance Gaussian features, the basic linear classifier is Bayes-optimal. 271

The negative logarithm of the Gaussian likelihood can be interpreted as a squared distance. 271

A good probabilistic treatment of a machine learning problem achieves a balance between solid theoretical foundations and the pragmatism required to obtain a workable solution. 273

An often overlooked consequence of having uncalibrated probability estimates such as those produced by naive Bayes is that both the ML and MAP decision rules become inadequate. 277

Tree models ignore the scale of quantitative features, treating them as ordinal. 305

Fitting data to a fixed linear decision boundary in log-odds space by means of feature calibration can be understood as training a naive Bayes model. 318

Low-bias models tend to have high variance, and vice versa. 338

Bagging is predominantly a variance-reduction technique, while boosting is primarily a bias-reduction technique. 339

Machine learning experiments pose questions about models that we try to answer by means of measurements on data. 343

The combination of precision and recall, and therefore the F-measure, is insensitive to the number of true negatives. 346

Confidence intervals are statements about estimates rather than

statements about the true value of the evaluation measure.

References

Abudawood, T. (**2011**). Multiclass subgroup discovery: Heuristics, algorithms and predictiveness. Ph.D. thesis, University of Bristol, Department of Computer Science, Faculty of Engineering. 357

Abudawood, T. and Flach, P.A. (**2009**). Evaluation measures for multiclass subgroup discovery. In W.L. Buntine, M. Grobelnik, D. Mladenić and J. Shawe-Taylor (eds.), *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD 2009), Part I, LNCS*, volume 5781, pp. 35–50. Springer. 193

Agrawal, R., Imielinski, T. and Swami, A.N. (**1993**). Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia (eds.), *Proceedings of the ACM International Conference on Management of Data (SIGMOD 1993)*, pp. 207–216. ACM Press. 103

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H. and Verkamo, A.I. (**1996**). Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pp. 307–328. AAAI/MIT Press. 193

Allwein, E.L., Schapire, R.E. and Singer, Y. (**2000**). Reducing multiclass to binary: A unifying approach for margin classifiers. In P. Langley (ed.), *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pp. 9–16. Morgan Kaufmann. 102

Amit, Y. and Geman, D. (**1997**). Shape quantization and recognition with randomized trees. *Neural Computation* 9(7):1545–1588. 341

Angluin, D., Frazier, M. and Pitt, L. (**1992**). Learning conjunctions of Horn clauses. *Machine Learning* 9:147–164. 128

Bakir, G., Hofmann, T., Schölkopf, B., Smola, A.J., Taskar, B. and

Vishwanathan, S.V.N. (**2007**). *Predicting Structured Data*. MIT Press. 361

Banerji, R.B. (**1980**). *Artificial Intelligence: A Theoretical Approach*. Elsevier Science. 127

Bengio, Y. (**2009**). Learning deep architectures for AI. *Foundations and Trends in Machine Learning* 2(1):1–127. 361

Best, M.J. and Chakravarti, N. (**1990**). Active set algorithms for isotonic regression; a unifying framework. *Mathematical Programming* 47(1):425–439. 80, 229

Blockeel, H. (**2010***a*). Hypothesis language. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 507–511. Springer. 127

Blockeel, H. (**2010***b*). Hypothesis space. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 511–513. Springer. 127

Blockeel, H., De Raedt, L. and Ramon, J. (**1998**). Top-down induction of clustering trees. In J.W. Shavlik (ed.), *Proceedings of the Fifteenth International Conference on Machine Learning (ICML1998)*, pp. 55–63. Morgan Kaufmann. 103, 156

Blumer, A., Ehrenfeucht, A., Haussler, D. and Warmuth, M.K. (**1989**). Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM* 36(4):929–965. 128

Boser, B.E., Guyon, I. and Vapnik, V. (**1992**). A training algorithm for optimal margin classifiers. In *Proceedings of the International Conference on Computational Learning Theory (COLT 1992)*, pp. 144–152. 229

Bouckaert, R. and Frank, E. (**2004**). Evaluating the replicability of significance tests for comparing learning algorithms. In H. Dai, R. Srikant and C. Zhang (eds.), *Advances in Knowledge Discovery and Data Mining, LNCS*, volume 3056, pp. 3–12. Springer. 358

Boullé, M. (**2004**). Khiops: A statistical discretization method of continuous attributes. *Machine Learning* 55(1):53–69. 328

Boullé, M. (**2006**). MODL: A Bayes optimal discretization method for

continuous attributes. *Machine Learning* 65(1):131–165. 328

Bourke, C., Deng, K., Scott, S.D., Schapire, R.E. and Vinodchandran, N.V. (**2008**). On reoptimizing multiclass classifiers. *Machine Learning* 71(2–3):219–242. 102

Brazdil, P., Giraud-Carrier, C.G., Soares, C. and Vilalta, R. (**2009**). *Metalearning – Applications to Data Mining*. Springer. 342

Brazdil, P, Vilalta, R., Giraud-Carrier, C.G. and Soares, C. (**2010**). Metalearning. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 662–666. Springer. 342

Breiman, L. (**1996***a*). Bagging predictors. *Machine Learning* 24(2):123–140. 341

Breiman, L. (**1996***b*). Stacked regressions. *Machine Learning* 24(1):49–64. 342

Breiman, L. (**2001**). Random forests. *Machine Learning* 45(1):5–32. 341

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (**1984**). *Classification and Regression Trees*. Wadsworth. 156

Brier, G.W. (**1950**). Verification of forecasts expressed in terms of probability. *Monthly Weather Review* 78(1):1–3. 80

Brown, G. (**2010**). Ensemble learning. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 312–320. Springer. 341

Bruner, J.S., Goodnow, J.J. and Austin, G.A. (**1956**). *A Study of Thinking*. Science Editions. 2nd edn 1986. 127

Cesa-Bianchi, N. and Lugosi, G. (**2006**). *Prediction, Learning, and Games*. Cambridge University Press. 361

Cestnik, B. (**1990**). Estimating probabilities: A crucial task in machine learning. In *Proceedings of the European Conference on Artificial Intelligence (ECAI 1990)*, pp. 147–149. 296

Clark, P. and Boswell, R. (**1991**). Rule induction with CN2: Some recent

improvements. In Y. Kodratoff (ed.), *Proceedings of the European Working Session on Learning (EWSL 1991), LNCS*, volume 482, pp. 151–163. Springer. 192

Clark, P. and Niblett, T. (**1989**). The CN2 induction algorithm. *Machine Learning* 3:261–283. 192

Cohen, W.W. (**1995**). Fast effective rule induction. In A. Prieditis and S.J. Russell (eds.), *Proceedings of the Twelfth International Conference on Machine Learning (ICML 1995)*, pp. 115–123. Morgan Kaufmann. 192, 341

Cohen, W.W. and Singer, Y. (**1999**). A simple, fast, and effictive rule learner. In J. Hendler and D. Subramanian (eds.), *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI 1999)*, pp. 335–342. AAAI Press / MIT Press. 341

Cohn, D. (**2010**). Active learning. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 10–14. Springer. 128

Cortes, C. and Vapnik, V. (**1995**). Support-vector networks. *Machine Learning,* 20(3):273–297. 229

Cover, T. and Hart, E (**1967**). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* 13(1):21–27. 260

Cristianini, N. and Shawe-Taylor, J. (**2000**). *An Introduction to Support Vector Machines*. Cambridge University Press. 229

Dasgupta, S. (**2010**). Active learning theory. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 14–19. Springer. 128

Davis, J. and Goadrich, M. (**2006**). The relationship between precision-recall and ROC curves. In W.W. Cohen and A. Moore (eds.), *Proceedings of the Twenty-Third International Conference on Machine Learning (ICML 2006)*, pp. 233–240. ACM Press. 358

De Raedt, L. (**1997**). Logical settings for concept-learning. *Artificial Intelligence* 95(1):187–201. 128

De Raedt, L. (**2008**). *Logical and Relational Learning*. Springer. 193

De Raedt, L. (**2010**). Logic of generality. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 624–631. Springer. 128

De Raedt, L. and Kersting, K. (**2010**). Statistical relational learning. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 916–924. Springer. 193

Dempster, A.P., Laird, N.M. and Rubin, D.B. (**1977**). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)* pp. 1–38. 296

Demšar, J. (**2008**). On the appropriateness of statistical tests in machine learning. In *Proceedings of the ICML'08 Workshop on Evaluation Methods for Machine Learning*. 359

Demšar, J. (**2006**). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* 7: 1–30. 359

Dietterich, T.G. (**1998**). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation* 10(7):1895–1923. 358

Dietterich, T.G. and Bakiri, G. (**1995**). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research* 2: 263–286. 102

Dietterich, T.G., Kearns, M.J. and Mansour, Y. (**1996**). Applying the weak learning framework to understand and improve c4.5. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 96–104. 156

Ding, C.H.Q. and He, X. (**2004**). *K*-means clustering via principal component analysis. In C.E. Brodley (ed.), *Proceedings of the Twenty-First International Conference on Machine Learning (ICML 2004)*. ACM Press. 329

Domingos, P. and Pazzani, M. (**1997**). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning* 29(2):103–130. 296

Donoho, S.K. and Rendell, L.A. (**1995**). Rerepresenting and restructuring domain theories: A constructive induction approach. *Journal of Artificial Intelligence Research* 2: 411–446. 328

Drummond, C. (**2006**). Machine learning as an experimental science (revisited). In *Proceedings of the AAAI'06 Workshop on Evaluation Methods for Machine Learning*. 359

Drummond, C. and Holte, R.C. (**2000**). Exploiting the cost (in)sensitivity of decision tree splitting criteria. In P. Langley (ed.), *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pp. 239–246. Morgan Kaufmann. 156

Egan, J.P. (**1975**). *Signal Detection Theory and ROC Analysis*. Academic Press. 80

Fawcett, T. (**2006**). An introduction to ROC analysis. *Pattern Recognition Letters* 27(8):861–874. 80, 358

Fawcett, T. and Niculescu-Mizil, A. (**2007**). PAV and the ROC convex hull. *Machine Learning* 68(1):97–106. 80, 229

Fayyad, U.M. and Irani, K.B. (**1993**). Multi-interval discretization of continuousvalued attributes for classification learning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 1993)*, pp. 1022–1029. 328

Ferri, C., Flach, P.A. and Hernández-Orallo, J. (**2002**). Learning decision trees using the area under the ROC curve. In C. Sammut and A.G. Hoffmann (eds.), *Proceedings of the Nineteenth International Conference on Machine Learning (ICML 2002*), pp. 139–146. Morgan Kaufmann. 156

Ferri, C., Flach, P.A. and Hernández-Orallo, J. (**2003**). Improving the AUC of probabilistic estimation trees. In N. Lavrač, D. Gamberger, L. Todorovski and H. Blockeel (eds.), *Proceedings of the European Conference on Machine Learning (ECML 2003), LNCS*, volume 2837, pp. 121–132. Springer. 156

Fix, E. and Hodges, J.L. (**1951**). Discriminatory analysis. Nonparametric discrimination: Consistency properties. Technical report, USAF School of Aviation Medicine, Texas: Randolph Field. Report Number 4, Project Number 21–49-004. 260

Flach, P.A. (**1994**). *Simply Logical – Intelligent Reasoning by Example*. Wiley. 193

Flach, P.A. (**2003**). The geometry of ROC space: Understanding machine learning metrics through ROC isometrics. In T. Fawcett and N. Mishra (eds.), *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*, pp. 194–201. AAAI Press. 156

Flach, P.A. (**2010***a*). First-order logic. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 410–415. Springer. 128

Flach, P.A. (**2010***b*). ROC analysis. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 869–875. Springer. 80

Flach, P.A. and Lachiche, N. (**2001**). Confirmation-guided discovery of first-order rules with Tertius. *Machine Learning* 42(1/2):61–95. 193

Flach, P.A. and Matsubara, E.T. (**2007**). A simple lexicographic ranker and probability estimator. In J.N. Kok, J. Koronacki, R.L. de Mántaras, S. Matwin, D. Mladenic and A. Skowron (eds.), *Proceedings of the Eighteenth European Conference on Machine Learning (ECML 2007), LNCS*, volume 4701, pp. 575–582. Springer. 80, 229

Freund, Y., **Iyer, R.D., Schapire, R.E. and Singer, Y.** (**2003**). An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research* 4: 933–969. 341

Freund, Y. and Schapire, R.E. (**1997**). A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci*. 55(1):119–139. 341

Fürnkranz, J. (**1999**). Separate-and-conquer rule learning. *Artificial Intelligence Review* 13(1):3–54. 192

Fürnkranz, J. (**2010**). Rule learning. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 875–879. Springer. 192

Fürnkranz, J. and Flach, P.A. (**2003**). An analysis of rule evaluation metrics. In T. Fawcett and N. Mishra (eds.), *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*, pp. 202–209. AAAI Press. 79

Fürnkranz, J. and Flach, P.A. (**2005**). ROC 'n' Rule learning – towards a better understanding of covering algorithms. *Machine Learning* 58(1):39–77.

192

Fürnkranz, J., Gamberger, D. and Lavrač, N. (**2012**). *Foundations of Rule Learning*. Springer. 192

Fürnkranz, J. and Hüllermeier, E. (eds.) **(2010)**. *Preference Learning*. Springer. 361

Fürnkranz, J. and Widmer, G. (**1994**). Incremental reduced error pruning. In *Proceedings of the Eleventh International Conference on Machine Learning (ICML 1994)*, pp. 70–77. 192

Gama, J. and Gaber, M.M. (eds.) **(2007)**. *Learning from Data Streams: Processing Techniques in Sensor Networks*. Springer. 361

Ganter, B. and Wille, R. (**1999**). *Formal Concept Analysis: Mathematical Foundations*. Springer. 127

Garriga, G.C., Kralj, P. and Lavrač, N. (**2008**). Closed sets for labeled data. *Journal of Machine Learning Research* 9: 559–580. 127

Gärtner, T. (**2009**). *Kernels for Structured Data*. World Scientific. 230

Grünwald, P.D. (2007). *The Minimum Description Length Principle*. MIT Press. 297

Guyon, I. and Elisseeff, A. (**2003**). An introduction to variable and feature selection. *Journal of Machine Learning Research* 3: 1157–1182. 328

Hall, M.A. (**1999**). Correlation-based feature selection for machine learning. Ph.D. thesis, University of Waikato. 328

Han, J., Cheng, H., Xin, D. and Yan, X. (**2007**). Frequent pattern mining: Current status and future directions. *Data Mining and Knowledge Discovery* 15(1):55–86. 193

Hand, D.J. and Till, R.J. (**2001**). A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning* 45(2):171–186. 102

Haussler, D. (**1988**). Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence* 36(2):177–221. 128

Hernández-Orallo, J., Flach, P.A. and Ferri, C. (**2011**). Threshold choice methods: The missing link. Available online at http://arxiv.org/abs/1112.2640. 358

Ho, T.K. (**1995**). Random decision forests. In *Proceedings of the International Conference on Document Analysis and Recognition*, p. 278. IEEE Computer Society, Los Alamitos, CA, USA. 341

Hoerl, A.E. and Kennard, R.W. (**1970**). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics* pp. 55–67. 228

Hofmann, T. (**1999**). Probabilistic latent semantic indexing. In *Proceedings of the Twenty-Second Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR 1999)*, pp. 50–57. ACM Press. 329

Hunt, E.B., Marin, J. and Stone, P.J. (**1966**). *Experiments in Induction*. Academic Press. 127, 156

Jain, A.K., Murty, M.N. and Flynn, P.J. (**1999**). Data clustering: A review. *ACM Computing Surveys* 31(3):264–323. 261

Japkowicz, N. and Shah, M. (**2011**). *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press. 357

Jebara, T. (**2004**). *Machine Learning: Discriminative and Generative*. Springer. 296

John, G.H. and Langley, P. (**1995**). Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI 1995)*, pp. 338–345. Morgan Kaufmann. 295

Kaufman, L. and Rousseeuw, P.J. (**1990**). *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley. 261

Kearns, M.J. and Valiant, L.G. (**1989**). Cryptographic limitations on learning

Boolean formulae and finite automata. In D.S. Johnson (ed.), *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing (STOC 1989)*, pp. 433–444. ACM Press. 341

Kearns, M.J. and Valiant, L.G. (**1994**). Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the ACM* 41(1):67–95. 341

Kerber, R. (**1992**). Chimerge: Discretization of numeric attributes. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI 1992)*, pp. 123–128. AAAI Press. 328

Kibler, D.F. and Langley, P. (**1988**). Machine learning as an experimental science. In *Proceedings of the European Working Session on Learning (EWSL 1988)*, pp. 81–92. 359

King, R.D., Srinivasan, A. and Dehaspe, L. (**2001**). Warmr: A data mining tool for chemical data. *Journal of Computer-Aided Molecular Design* 15(2):173–181. 193

Kira, K. and Rendell, L.A. (**1992**). The feature selection problem: Traditional methods and a new algorithm. In W.R. Swartout (ed.), *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI 1992)*, pp. 129–134. AAAI Press / MIT Press. 328

Klösgen, W. (**1996**). Explora: A multipattern and multistrategy discovery assistant. In *Advances in Knowledge Discovery and Data Mining*, pp. 249–271. MIT Press. 103

Kohavi, R. and John, G.H. (**1997**). Wrappers for feature subset selection. *Artificial Intelligence* 97(1–2):273–324. 328

Koren, Y., **Bell, R. and Volinsky, C.** (**2009**). Matrix factorization techniques for recommender systems. *IEEE Computer* 42(8):30–37. 328

Kramer, S. (**1996**). Structural regression trees. In *Proceedings of the National Conference on Artificial Intelligence (AAAI 1996)*, pp. 812–819. 156

Kramer, S., Lavrač, N. and Flach, P.A. (**2000**). Propositionalization approaches to relational data mining. In S. Džeroski and N. Lavrač (eds.), *Relational Data Mining*, pp. 262–286. Springer. 328

Krogel, M.A., Rawles, S., Zelezný, F., Flach, P.A., Lavrač, N. and Wrobel, S. (**2003**). Comparative evaluation of approaches to propositionalization. In T. Horváth (ed.), *Proceedings of the Thirteenth International Conference on Inductive Logic Programming (ILP 2003), LNCS*, volume 2835, pp. 197–214. Springer. 328

Kuncheva, L.I. (**2004**). *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley and Sons. 341

Lachiche, N. (**2010**). Propositionalization. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 812–817. Springer. 328

Lachiche, N. and Flach, P.A. (**2003**). Improving accuracy and cost of two-class and multiclass probabilistic classifiers using ROC curves. In T. Fawcett and N. Mishra (eds.), *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*, pp. 416–423. AAAI Press. 102

Lafferty, J.D., McCallum, A. and Pereira, F.C.N. (**2001**). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In C.E. Brodley and A.P. Danyluk (eds.), *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pp. 282–289. Morgan Kaufmann. 296

Langley, P. (**1988**). Machine learning as an experimental science. *Machine Learning* 3:5–8. 359

Langley, P. (**1994**). *Elements of Machine Learning*. Morgan Kaufmann. 156

Langley, P. (**2011**). The changing science of machine learning. *Machine Learning* 82(3):275–279. 359

Lavrač, N., Kavšek, B., Flach, P.A. and Todorovski, L. (**2004**). Subgroup discovery with CN2-SD. *Journal of Machine Learning Research* 5: 153–188. 193

Lee, D.D., Seung, H.S. *et al*. **(1999)**. Learning the parts of objects by non-negative matrix factorization. *Nature* 401(6755):788–791. 328

Leman, D., Feelders, A. and Knobbe, A.J. (**2008**). Exceptional model mining. In W. Daelemans, B. Goethals and K. Morik (eds.), *Proceedings of the*

European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD 2008), Part II, LNCS, volume 5212, pp. 1–16. Springer. 103

Lewis, D. (**1998**). Naive Bayes at forty: The independence assumption in information retrieval. In *Proceedings of the Tenth European Conference on Machine Learning (ECML 1998)*, pp. 4–15. Springer. 295

Li, W., **Han, J. and Pei, J.** (**2001**). CMAR: Accurate and efficient classification based on multiple class-association rules. In N. Cercone, T.Y. Lin and X. Wu (eds.), *Proceedings of the IEEE International Conference on Data Mining (ICDM 2001)*, pp. 369–376. IEEE Computer Society. 193

Little, R.J.A. and Rubin, D.B. (**1987**). *Statistical Analysis with Missing Data*. Wiley. 296

Liu, B., Hsu, W. and Ma, Y. (**1998**). Integrating classification and association rule mining. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD 1998)*, pp. 80–86. AAAI Press. 193

Lloyd, J.W. (**2003**). *Logic for Learning – Learning Comprehensible Theories from Structured Data*. Springer. 193

Lloyd, S. (**1982**). Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28(2):129–137. 261

Mahalanobis, P.C. (1936). On the generalised distance in statistics. *Proceedings of the National Institute of Science, India* 2(1):49–55. 260

Mahoney, M.W. and Drineas, P. (**2009**). CUR matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences* 106(3):697. 329

McCallum, A. and Nigam, K. (**1998**). A comparison of event models for naive Bayes text classification. In *Proceedings of the AAAI-98 Workshop on Learning for Text Categorization*, pp. 41–48. 295

Michalski, R.S. (**1973**). Discovering classification rules using variable-valued logic system VL_1 .In *Proceedings of the Third International Joint Conference*

on Artificial Intelligence, pp. 162–172. Morgan Kaufmann Publishers. 127

Michalski, R.S. (**1975**). Synthesis of optimal and quasi-optimal variable-valued logic formulas. In *Proceedings of the 1975 International Symposium on Multiple-Valued Logic*, pp. 76–87. 192

Michie, D., Spiegelhalter, D.J. and Taylor, C.C. (**1994**). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood. 342

Miettinen, P (**2009**). Matrix decomposition methods for data mining: Computational complexity and algorithms. Ph.D. thesis, University of Helsinki. 329

Minsky, M and Papert, S (**1969**). *Perceptrons: An Introduction to Computational Geometry*. MIT Press. 228

Mitchell, T M (**1977**). Version spaces: A candidate elimination approach to rule learning. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pp. 305–310. Morgan Kaufmann Publishers. 127

Mitchell, T.M. (**1997**). *Machine Learning*. McGraw-Hill. 128

Muggleton, S. (**1995**). Inverse entailment and Progol. *New Generation Computing* 13(3&4):245–286. 193

Muggleton, S., De Raedt, L., Poole, D., Bratko, I., Flach, P.A., Inoue, K. and Srinivasan, A. (**2012**). ILP turns 20 – biography and future challenges. *Machine Learning* 86(1):3 – 23. 193

Muggleton, S. and Feng, C. (**1990**). Efficient induction of logic programs. In *Proceedings of the International Conference on Algorithmic Learning Theory (ALT 1990*), pp. 368–381. 193

Murphy, A.H. and Winkler, R.L. (**1984**). Probability forecasting in meteorology. *Journal of the American Statistical Association* pp. 489–500. 80

Nelder, J.A. and Wedderburn, R.W.M. (**1972**). Generalized linear models. *Journal of the Royal Statistical Society, Series A (General)* pp. 370–384. 296

Novikoff, A.B. (**1962**). On convergence proofs on perceptrons. In *Proceedings*

of the Symposium on the Mathematical Theory of Automata, volume 12, pp. 615–622. Polytechnic Institute of Brooklyn, New York. 228

Pasquier, N., Bastide, Y., Taouil, R. and Lakhal, L. (**1999**). Discovering frequent closed itemsets for association rules. In *Proceedings of the International Conference on Database Theory (ICDT 1999)*, pp. 398–416. Springer. 127

Peng, Y., Flach, P.A., Soares, C. and Brazdil, P. (**2002**). Improved dataset characterisation for meta-learning. In S. Lange, K. Satoh and C.H. Smith (eds.), *Proceedings of the Fifth International Conference on Discovery Science (DS 2002), LNCS*, volume 2534, pp. 141–152. Springer. 342

Pfahringer, B., Bensusan, H. and Giraud-Carrier, C.G. (**2000**). Metalearning by land-marking various learning algorithms. In P. Langley (ed.), *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pp. 743–750. Morgan Kaufmann. 342

Platt, J.C. (**1998**). Using analytic QP and sparseness to speed training of support vector machines. In M.J. Kearns, S.A. Solla and D.A. Cohn (eds.), *Advances in Neural Information Processing Systems 11 (NIPS 1998)*, pp. 557–563. MIT Press. 229

Plotkin, G.D. (**1971**). Automatic methods of inductive inference. Ph.D. thesis, University of Edinburgh. 127

Provost, F.J. and Domingos, P. (2003). Tree induction for probability-based ranking. *Machine Learning* 52(3):199–215. 156

Provost, F.J. and Fawcett, T. (**2001**). Robust classification for imprecise environments. *Machine Learning* 42(3):203–231. 79

Quinlan, J.R. (**1986**). Induction of decision trees. *Machine Learning* 1(1):81–106. 155

Quinlan, J.R. (**1990**). Learning logical definitions from relations. *Machine Learning* 5: 239–266. 193

Quinlan, J.R. (**1993**). *C4.5: Programs for Machine Learning*. Morgan Kaufmann. 156

Ragavan, H. and Rendell, L.A. (**1993**). Lookahead feature construction for learning hard concepts. In *Proceedings of the Tenth International Conference on Machine Learning (ICML 1993)*, pp. 252–259. Morgan Kaufmann. 328

Rajnarayan, D.G. and Wolpert, D. (**2010**). Bias-variance trade-offs: Novel applications. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 101–110. Springer. 103

Rissanen, J. (**1978**). Modeling by shortest data description. *Automatica* 14(5):465–471. 297

Rivest, R.L. (**1987**). Learning decision lists. *Machine Learning* 2(3):229–246. 192

Robnik-Sikonja, M. and Kononenko, I. (**2003**). Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning* 53(1–2):23–69. 328

Rosenblatt, F. (**1958**). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* 65(6):386. 228

Rousseeuw, P.J. (**1987**). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics* 20(0):53–65. 261

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (**1986**). Learning representations by back-propagating errors. *Nature* 323(6088):533–536. 229

Schapire, R.E. (**1990**). The strength of weak learnability. *Machine Learning* 5: 197–227. 341

Schapire, R.E. (**2003**). The boosting approach to machine learning: An overview. In *Nonlinear Estimation and Classification*, pp. 149–172. Springer. 341

Schapire, R.E., Freund, Y., Bartlett, P. and Lee, W.S. (**1998**). Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics* 26(5):1651–1686. 341

Schapire, R.E. and Singer, Y. (**1999**). Improved boosting algorithms using confidence-rated predictions. *Machine Learning* 37(3):297–336. 341

Settles, B. (**2011**). *Active Learning*. Morgan & Claypool. 361

Shawe-Taylor, J. and Cristianini, N. (**2004**). *Kernel Methods for Pattern Analysis*. Cambridge University Press. 230

Shotton, J., Fitzgibbon, A.W., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A. and Blake, A. (**2011**). Real-time human pose recognition in parts from single depth images. In *Proceedings of the Twenty-Fourth IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2011)*, pp. 1297–1304. 155

Silver, D. and Bennett, K. (**2008**). Guest editor's introduction: special issue on inductive transfer learning. *Machine Learning* 73(3):215–220. 361

Solomonoff, R.J. (**1964***a*). A formal theory of inductive inference: Part I. *Information and Control* 7(1):1–22. 297

Solomonoff, R.J. (**1964***b*). A formal theory of inductive inference: Part II. *Information and Control* 7(2):224–254. 297

Srinivasan, A. (**2007**). The Aleph manual, version 4 and above. Available online at `www.cs.ox.ac.uk/activities/machlearn/Aleph/`. 193

Stevens, S.S. (**1946**). On the theory of scales of measurement. *Science* 103(2684):677–680. 327

Sutton, R.S. and Barto, A.G. (**1998**). *Reinforcement Learning: An Introduction*. MIT Press. 361

Tibshirani, R. (**1996**). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B (Methodological)* pp. 267–288. 228

Todorovski, L. and Dzeroski, S. (**2003**). Combining classifiers with meta decision trees. *Machine Learning* 50(3):223–249. 342

Tsoumakas, G., Zhang, M.L. and Zhou, Z.H. (**2012**). Introduction to the special issue on learning from multi-label data. *Machine Learning* 88(1-2):1–4. 361

Tukey, J.W. (**1977**). *Exploratory Data Analysis*. Addison-Wesley. 103

Valiant, L.G. (**1984**). A theory of the learnable. *Communications of the ACM* 27(11):1134–1142. 128

Vapnik, V.N. and Chervonenkis, A.Y. (**1971**). On uniform convergence of the frequencies of events to their probabilities. *Teoriya Veroyatnostei I Ee Primeneniya* 16(2):264–279. 128

Vere, S.A. (**1975**). Induction of concepts in the predicate calculus. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, pp. 281–287. 127

von Hippel, P.T. (**2005**). Mean, median, and skew: Correcting a textbook rule. *Journal of Statistics Education* 13(2). 327

Wallace, C.S. and Boulton, D.M. (**1968**). An information measure for classification. *Computer Journal* 11(2):185–194. 297

Webb, G.I. (**1995**). Opus: An efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research* 3: 431–465. 192

Webb, G.I., Boughton, J.R. and Wang, Z. (**2005**). Not so naive Bayes: Aggregating one-dependence estimators. *Machine Learning* 58(1):5–24. 295

Winston, P.H. (**1970**). Learning structural descriptions from examples. Technical report, MIT Artificial Intelligence Lab. AITR-231. 127

Wojtusiak, J., Michalski, R.S., Kaufman, K.A. and Pietrzykowski, J. (**2006**). The AQ21 natural induction program for pattern discovery: Initial version and its novel features. In *Proceedings of the Eighteenth IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2006)*, pp. 523–526. 192

Wolpert, D.H. (**1992**). Stacked generalization. *Neural Networks* 5(2):241–259. 342

Zadrozny, B. and Elkan, C. (**2002**). Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD 2002)*, pp. 694–699. ACM Press. 80, 229

Zeugmann, T. (**2010**). PAC learning. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 745–753. Springer. 128

Zhou, Z.H. (**2012**). *Ensemble Methods: Foundations and Algorithms*. Taylor & Francis. 341

Index

$\sqrt{\text{Gini}}$, *see* impurity measure, $\sqrt{\text{Gini}}$
0–1 loss, **62**
0-norm, **234**
1-norm, **234**, 239
2-norm, **234**, 239

abstraction, **306**
accuracy, **19**, **54**, 57

as a weighted average
Example 2.1, 56

expected
Example 12.3, 347
Example 12.1, 346
macro-averaged, **60**
ranking, *see* ranking accuracy
active learning, **122, 361**
adjacent violators, **77**
affine transformation, **195**
agglomerative merging, **312**
AggloMerge(S, f, Q)
Algorithm 10.2, 312

aggregation
    in structured features, **306**
Aleph, *see* ILP systems
analysis of variance, **355**
anti-unification, **123**
Apriori, *see* association rule algorithms
AQ, *see* rule learning systems

arithmetic mean
minimises squared Euclidean distance, 291
Theorem 8.1, 238
association rule, **15, 184**

association rule algorithms
Apriori, 193
Warmr, 193

association rule discovery
    Example 3.12, 102
AssociationRules(D, f_0, c_0)
    Algorithm 6.7, 185
at least as general as, **105**
attribute, *see* feature
AUC, **67**
    multi-class
        Example 3.5, 88
average recall, *see* recall, average, 101

backtracking search, **133**
bag of words, **41**
bagging, 156, 331–333
Bagging(D, T, $\mathscr{A}$)
    Algorithm 11.1, 332
basic linear classifier, **21**, 207–228, 238, 241, 242, 269, 317, 321, 332, 334–336, 339, 364
    Bayes-optimality, 271
    kernel trick, 43
Bayes' rule, **27**
Bayes-optimal, **29**, 30, **265**
    basic linear classifier, 271
beam search, **170**
Bernoulli distribution, 148, **274**
    multivariate, **273**
Bernoulli trial, 148, **273**
Bernoulli, Jacob, 45, 274
BestSplit-Class(D, F)
    Algorithm 5.2, 137
bias, **94**
bias–variance dilemma, **93**, 338
big data, 362
bigram, 322
bin, **309**
binomial distribution, **274**
bit vector, **273**
Bonferroni–Dunn test, **357**
boosting, 63, 334–338

weight updates
Example 11.1, 334
Boosting(D, T, $\mathscr{A}$)
Algorithm 11.3, 335
bootstrap sample, **331**
breadth-first search, 183
Brier score, **74**

C4.5, *see* tree learning systems
calibrating classifier scores, 223, 316
calibration, 220
isotonic, **78**, 223, 286, **318**
logistic, 286, **316**
loss, **76**
map, **78**
CART, *see* tree learning systems
Cartesian product, **51**
categorical distribution, **274**
CD, *see* critical difference
central limit theorem, **220**, 350
central moment, **303**
centre around zero, 24, **198, 200**, 203, 324
centre of mass, 24
centroid, **97, 238**
characteristic function, **51**
Chebyshev distance, **234**
Chebyshev's inequality, **301**
Chervonenkis, Alexey, 125
chicken-and-egg problem, 287
cityblock distance, *see* Manhattan distance

class

imbalance
Example 2.4, 67
label, **52**
ratio, 57, 58, 61, 62, 67, 70, 71, 142, 143, 147, 221
class probability estimation, 360
class probability estimator, **72**

squared error
Example 2.6, 74
tree, 141

classification
    binary, **52**
    multi-class, **14**, 82
classifier, **52**
clause, **105**
cloud computing, 362
clustering, **14**
    agglomerative, **255**, 310
    descriptive, **18**

evaluation
Example 3.10, 99
predictive, **18**

representation
Example 3.9, 98
stationary point, **249**
Example 8.5, 249
clustering tree, 253

using a dissimilarity matrix
Example 5.5, 153

using Euclidean distance
Example 5.6, 155
CN2, *see* rule learning systems
CNF, *see* conjunctive normal form
comparable, **51**
complement, 178
component, **266**
computational learning theory, 124
concavity, **76**
concept, 157
closed, **117**, 183
complete, **113**
conjunctive, 106
Example 4.1, 106
consistent, **113**
concept learning, **52, 104**

negative examples
Example 4.2, 110
conditional independence, 281
conditional likelihood, **283**
conditional random field, 296
confidence, 57, **184**
confidence interval, **351**
Example 12.5, 352
confusion matrix, **53**
conjugate prior, **265**
conjunction $\wedge$, **34, 105**
conjunctive normal form, **105**, 119
conjunctively separable, **115**
constructive induction, **131**
contingency table, **53**
continuous feature, *see* feature, quantitative
convex, **213, 241**
hull, **78**
lower, **309**
loss function, **63**
ROC curve, **76**, 138
set, **113, 183**
correlation, 151
correlation coefficient, **45, 267**, 323
cosine similarity, **259**
cost ratio, **71**, 142
count vector, **275**
counter-example, **120**
covariance, **45**, 198
covariance matrix, **200**, 202, 204, 237, 267, 269, 270
coverage counts, **86**

as score
Example 3.4, 87
coverage curve, **65**
coverage plot, **58**
covering algorithm, **163**
weighted, *see* weighted covering
covers, **105, 182**
critical difference, **356**
critical value, **354**
cross-validation, 19, **349**
Example 12.4, 350
internal, **358**
stratified, **350**
curse of dimensionality, **243**

d-prime, **316**
data mining, 182, **362**
data set characteristics, 340
data streams, **361**
De Morgan laws, **105**, 131
decile, **301**
decision boundary, 4, 14
decision list, **34**, 192
decision rule, **26**
decision stump, **339**

decision threshold

tuning
Example 2.5, 71
decision tree, **33**, 53, 101, 314
decoding, **83**
loss-based, **85**
deduction, **20**
deep learning, **361**
default rule, **35, 161**

degree of freedom
in *t*-distribution, **353**
in contingency table, **58**
dendrogram, **254**
Definition 8.4, 254
descriptive clustering, **96**, 245
descriptive model, **17**
dimensionality reduction, 326
Dirichlet prior, **75**
discretisation, 155
agglomerative, **310**

agglomerative merging
Example 10.7, 313
bottom–up, **310**
divisive, **310**
equal-frequency, **309**
equal-width, **310**

recursive partitioning
Example 10.6, 311
top–down, **310**
disjunction v , **34, 105**
disjunctive normal form, **105**
dissimilarity, 96, **152**
cluster, **152**
split, **152**
distance, **23**

elliptical
Example 8.1, 237
Euclidean, **23**, 305
Manhattan, **25**
distance metric, **235**, 305
Definition 8.2, 236
distance weighting, **244**
divide-and-conquer, 35, **133**, 138, 161
DKM, **156**
DNF, *see* disjunctive normal form
dominate, **59**
DualPerceptron(*D*)
Algorithm 7.2, 209

Eddington, Arthur, 343
edit distance, **235**
eigendecomposition, **325**
Einstein, Albert, 30, 343
EM, *see* Expectation-Maximisation
empirical probability, **75**, 133, 135, 138
entropy, 159, **294**
as an impurity measure, *see* impurity measure, entropy
equivalence class, **51**
equivalence oracle, **120**
equivalence relation, **51**
error rate, **54**, 57
error-correcting output codes, 102
estimate, **45**
Euclidean distance, **234**
European Conference on Machine Learning, 2
European Conference on Principles and Practice of Knowledge Discovery in Databases, 2
evaluation measures, 344
for classifiers, 57
example, **50**
exceptional model mining, 103
excess kurtosis, *see* kurtosis
exemplar, **25, 97**, 132, **238**
Expectation-Maximisation, 97, **289**, 322
expected value, **45**, 267
experiment, **343**
experimental objective, **344**
explanation, **35**
explanatory variable, *see* feature
exponential loss, **63**, 337
extension, **105**

F-measure, **99, 300**
insensitivity to true negatives, 346
false alarm rate, **55**, 57
false negative, **55**
false negative rate, **55**, 57

false positive, **55**
false positive rate, **55**, 57
feature, **13, 50**, 262
    binarisation, **307**
    Boolean, **304**
    calibration, **314**
    categorical, 155, **304**
    construction, **41** , 50
    decorrelation, 202, 237, 270, 271
    discretisation, **42, 309**
    discretisation, supervised, **310**
    discretisation, unsupervised, **309**
    domain, **39, 50**
    list, **33**
    normalisation, 202, 237, 270, 271, **314**
    ordinal, 233, **304**
    quantitative, **304**
    space, **225**
    structured, **306**
        Example 10.4, 306
    thresholding, **308**
    thresholding, supervised, **309**
    thresholding, unsupervised, **308**
    transformation, **307**
    two uses of, 41
        Example 1.8, 41
    unordering, **307**
feature calibration, 277

categorical
Example 10.8, 315

isotonic
Example 10.11, 321
Example 10.10, 320

logistic
Example 10.9, 318
feature selection, 243
backward elimination, **324**
filter, **323**
forward selection, **324**
Relief, **323**
wrapper, **324**
feature tree, **32**, **132**, 155
Definition 5.1, 132
complete, **33**

growing
Example 5.2, 139

labelling
Example 1.5, 33
first-order logic, **122**
FOIL, *see* ILP systems
forecasting theory, **74**
frequency, *see* support
FrequentItems(D, f_0)
Algorithm 6.6, 184
Friedman test, **355**
Example 12.8, 356
function estimator, **91**
functor, **189**

Gauss, Carl Friedrich, 102, 196
Gaussian distribution, **266**
Gaussian kernel, **227**
bandwidth, **227**
Gaussian mixture model, 266, **289**

bivariate
Example 9.3, 269
relation to K-means, 292

univariate
Example 9.2, 269
general, **46**
generalised linear model, 296
generality ordering, **105**
generative model, **29**
geometric median, **238**
Gini coefficient, **134**
Gini index, 159
as an impurity measure, *see* impurity measure, Gini index
Gini, Corrado, 134
glb, *see* greatest lower bound
Golem, *see* ILP systems
Gosset, William Sealy, 353
gradient, **213**, 238
grading model, 52, 92
Gram matrix, **210**, 214, 325
greatest lower bound, **108**
greedy algorithm, 133
grouping model, 52, 92
GrowTree(D, F), 152
Algorithm 5.1, 132
Guinness, 353

HAC(D, L)
Algorithm 8.4, 255
Hamming distance, **84, 235**, 305
harmonic mean, 99
Hernández-Orallo, José, xvi
hidden variable, **16, 288**
hierarchical agglomerative clustering, 314
hinge loss, **63, 217**
histogram, **302**
Example 10.2, 303
homogeneous coordinates, 4, **24**, 195, 201
Horn clause, **105**, 119, **189**
Horn theory, 119

learning
Example 4.5, 122
Horn(*Mb*, *Eq*)
Algorithm 4.5, 120
Horn, Alfred, 105
Hume, David, 20
hyperplane, **21**
hypothesis space, **106**, 186

ID3, *see* tree learning systems
ILP, *see* inductive logic programming

ILP systems
Aleph, 193
FOIL, 193
Golem, 193
Progol, 35, 193
implication → , **105**

impurity
Example 5.1, 136
relative, **145**
impurity measure, 158
$\sqrt{\text{Gini}}$, **134**, 145, 147, 156, 337
entropy, **134**, 135, 136, 144, 147, 294
Gini index, **134**, 135, 136, 144, 147
as variance, 148
minority class, **134**, 135, 159
imputation, **322**
incomparable, **51**
independent variable, *see* feature
indicator function, **54**
induction, **20**
problem of, xvii, **20**
inductive bias, **131**
inductive logic programming, **189**, 307
information content, **293**
Example 9.7, 293
information gain, **136**, 310, 323
information retrieval, 99, 300, 326, 346
input space, **225**
instance, **49**
labelled, **50**
instance space, **21**, 39, 40, **49**
segment, **32**, 51, 104, **132**
intercept, **195**, 198
internal disjunction, **110**, 162
Example 4.3, 112
interquartile range, **301**, 314

isometric
$\sqrt{\text{Gini}}$, 145
accuracy, 61, 69, 77, 116
average recall, **60**, 72
entropy, 145
Gini index, 145
impurity, 159
precision, 167
precision (Laplace-corrected), 170
splitting criteria, 145
item set, **182**
closed, **183**
frequent, **182**

Jaccard coefficient, **15**
jackknife, 349

K-means, 25, 96, 97, **247**, 259, 289, 310
problem, **246**, 247
relation to Gaussian mixture model, 292
K-medoids, **250**, 310
k-nearest neighbour, **243**
Karush–Kuhn–Tucker conditions, **213**
kernel, **43**, 323

quadratic
Example 7.8, 225
kernel perceptron, **226**
kernel trick, **43**
Example 1.9, 44
Kernel-KMeans(D, K)
Algorithm 8.5, 259
KernelPerceptron(D, κ)
Algorithm 7.4, 226
Kinect motion sensing device, 129, 155
KKT, *see* Karush–Kuhn–Tucker conditions
KMeans(D, K)
Algorithm 8.1, 248
KMedoids(D, K, Dis)
Algorithm 8.2, 250
kurtosis, **303**

L_0 norm, *see* 0-norm
label space, **50**, 360
Lagrange multiplier, **213**
landmarking, 342
Langley, Pat, 359
Laplace correction, **75**, 138, 141, 147, 170, 265, 274, 279, 286
lasso, **205**
latent semantic indexing, **327**
latent variable, *see* hidden variable
lattice, **108**, 182, 186
law of large numbers, **45**
learnability, **124**
learning from entailment, **128**
learning from interpretations, **128**
learning model, **124**
learning rate, **207**
LearnRule(D), 169, 190
Algorithm 6.2, 164
LearnRuleForClass(D, C_i)
Algorithm 6.4, 171
LearnRuleList(D), 169, 190
Algorithm 6.1, 163
LearnRuleSet(D)
Algorithm 6.3, 171
least general generalisation, **108**, 112, 115–117, 131
least upper bound, **108**
least-squares classifier, **206**, 210

univariate
Example 7.4, 206
least-squares method, **196**
ordinary, **199**
total, **199**, 273
least-squares solution to a linear regression problem, 272
leave-one-out, **349**
Lebowski, Jeffrey, 16
level-wise search, 183
Levenshtein distance, **235**
LGG, *see* least general generalisation
LGG-Conj(x, y)
Algorithm 4.2, 110
LGG-Conj-ID(x, y)
Algorithm 4.3, 112
LGG-Set(D)
Algorithm 4.1, 108
lift, **186**
likelihood function, **27**, 305
likelihood ratio, **28**

linear
approximation, **195**
combination, **195**
function, **195**
model, **194**
transformation, **195**
linear classifier, 5, 21, 38, 40, 43, 81, 82, 207–223, 263, 282, 314, 333–340
Example 1, 3
coverage curve, 67
general form, 207, 364
geometric interpretation, 220

logistic calibration
Example 7.7, 223
margin, 22
VC-dimension, 126
linear discriminants, 21
linear regression, 92, 151

bivariate
Example 7.3, 203

univariate
Example 7.1, 198
linear, piecewise, **195**
linearly separable, 207
linkage function, **254**
Definition 8.5, 254
Example 8.7, 256
monotonicity, **257**
literal, **105**
Lloyd's algorithm, 247
local variables, **189, 306**
log-likelihood, 271
log-linear models, **223**
log-odds space, 277, **317**
logistic function, **221**
logistic regression, 223, **282**

univariate
Example 9.6, 285
loss function, **62**, 93
loss-based decoding
Example 3.3, 86
L_p norm, *see* p-norm
LSA, *see* latent semantic indexing
lub, *see* least upper bound

m-estimate, **75**, 141, 147, **279**
Mach, Ernst, 30
machine intelligence, 361

machine learning
    definition of, 3
    univariate, **52**
Mahalanobis distance, **237**, 271
majority class, **33**, 35, **53, 56**
Manhattan distance, **234**
manifold, **243**
MAP, *see* maximum a posteriori

margin
of a classifer, 62
of a decision boundary, **211**
of a linear classifier, **22**
of an example, **62**, 211, 339
margin error, **216**
marginal, **54, 186**
marginal likelihood, **29**
Example 1.4, 30
market basket analysis, 101

matrix
diagonal, 202
inverse, 267
rank, **326**
matrix completion, **327**
matrix decomposition, 17, 97, 324–327
Boolean, 327
non-negative, 328
with constraints, 326
maximum a posteriori, **28**, 263
maximum likelihood, **28**
maximum-likelihood estimation, **271**, 287
in linear regression, 199
maximum-margin classifier
Example 7.5, 216

soft margin
Example 7.6, 219
mean, 267, **299**
arithmetic, **300**
geometric, **300**
harmonic, **300**
mean squared error, **74**
median, 267, **299**
medoid, 153, **238**
membership oracle, **120**
meta-model, **339**
MGConsistent(C, N)
Algorithm 4.4, 116
midrange point, **301**

minimum description length
Definition 9.1, 294
Minkowski distance, **234**
Definition 8.1, 234

minority class
as an impurity measure, *see* impurity measure, minority class
missing values, 322
Example 1.2, 27
mixture model, **266**
ML, *see* maximum likelihood

MLM data set
Example 1.7, 40

clustering
Example 8.4, 249

hierarchical clustering
Example 8.6, 254
mode, 267, **299**
model, **13, 50**
declarative, **35**
geometric, **21**
grading, **36**
grouping, **36**
logical, 32
parametric, **195**
probabilistic, 25
univariate, **40**
model ensemble, **330**
model selection, **265**
model tree, **151**
monotonic, **182, 305**
more general than, **105**
MSE, *see* mean squared error
multi-class classifier

performance
Example 3.1, 82
multi-class probabilities
Example 3.7, 91
multi-class scores
from decision tree, 86
from naive Bayes, 86

reweighting
Example 3.6, 90
multi-label classification, **361**
multi-task learning, **361**
multinomial distribution, **274**
multivariate linear regression, 277

multivariate naive Bayes
decomposition into univariate models, **32**
multivariate normal distribution, 289

multivariate regression
decomposition into univariate regression, 203, 364

n-gram, **322**
naive Bayes, **30**, 33, 203, 278, 315, 318, 322
assumption, **275**
categorical features, 305
diagonal covariance matrix, 281
factorisation, 277, 281
ignores feature interaction, 44
linear in log-odds space, 277
multi-class scores, 86

prediction
Example 9.4, 276
recalibrated decision threshold, 277, 365
Scottish classifier, 32, 281
skewed probabilities, 277

training
Example 9.5, 280
variations, 280
nearest-neighbour classifier, **23, 242**
nearest-neighbour retrieval, **243**
negation ¬, **105**
negative recall, 57
neighbour, **238**
Nemenyi test, **356**
neural network, 207
Newton, Isaac, 30
no free lunch theorem, **20**, 340
noise, **50**
instance, **50**
label, **50**
nominal feature, *see* feature, categorical

normal distribution, 220, **266**, 305
multivariate, **267**
multivariate standard, **267**
standard, **267**, 270
normal vector, **195**
normalisation, **198**
row, **90**
null hypothesis, **352**

objective function, 63, **213**
Occam's razor, **30**
one-versus-one, **83**
one-versus-rest, **83**
online learning, **361**
operating condition, **72**
operating context, **345**

optimisation
constrained, 212, **213**
dual, **213**, 214
multi-criterion, **59**
primal, **213**
quadratic, 212, 213
Opus, *see* rule learning systems
ordinal feature, *see* feature, ordinal
ordinal, **299**
outlier, **198**, 238, **302**
Example 7.2, 199
output code, **83**
output space, **50**, 360
overfitting, 19, 33, 50, 91, 93, 97, 131, 151, 196, 210, 285, 323
Example 2, 6

p-norm, **234**
p-value, **352**
PAC, *see* probably approximately correct
paired *t*-test, **353**
Example 12.6, 353
PAM, *see* partitioning around medoids
PAM(D, K, Dis)
Algorithm 8.3, 251
Pareto front, **59**
partial order, **51**
partition, **51**
partition matrix, **97**
partitioning around medoids, **250**, 310
PCA, *see* principal component analysis
Pearson, Karl, 299
percentile, **301**
Example 10.1, 302
percentile plot, **301**
perceptron, **207**, 210
online, **208**
Perceptron(D, η)
Algorithm 7.1, 208
Perceptron Regression(D, T)
Algorithm 7.3, 211
piecewise linear, *see* linear, piecewise
population mean, **45**
post-hoc test, **356**
post-processing, **186**
posterior odds, **28**
Example 1.3, 29
posterior probability, **26**, 262
powerset, **51**
Príncipe, 343
precision, **57**, 99, **167**, 186, 300
Laplace-corrected, 170
predicates, *see* first-order logic

predicted positive rate, **347**
predictive clustering, **96**, 245, 289
predictive model, **17**
predictor variable, *see* feature
preference learning, **361**
principal component analysis, 24, 243, **324**
prior odds, **28**
prior probability, **27**

probabilistic model
    discriminative, **262**
    generative, **262**

probability distribution
    cumulative, **302**
    right-skewed, **303**
probability estimation tree, 73–76, 141, 147, 262, 263, 265
probability smoothing, **75**
probability space, **317**
probably approximately correct, **124**, 331
Progol, *see* ILP systems
projection, 219
Prolog, *see* query languages
propositional logic, **122**
propositionalisation, **307**
PruneTree(T, D)
    Algorithm 5.3, 144
pruning, **33, 142**
pruning set, **143**
pseudo-counts, **75**, 172, **274, 279**
pseudo-metric, **236**
pure, **133**
purity, 35, 158

quantile, **301**
quartile, **301**
query, **306**

query languages
    Prolog, 189–191, 193, 306
    SQL, 306

Rand index, **99**
random forest, 129, **333**, 339
random variable, **45**
RandomForest(D, T, d)
    Algorithm 11.2, 333
range, **301**
ranking, **64**
    Example 2.2, 64

accuracy
Example 2.3, 65
ranking accuracy, **64**
ranking error, **64**
ranking error rate, **64**
recalibrated likelihood decision rule, **277**
recall, 57, **58**, 99, 300
average, **60, 178**
receiver operating characteristic, **60**
RecPart(S, f, Q)
Algorithm 10.1, 311
recursive partitioning, **310**
reduced-error pruning, **143**, 147, 151
incremental, 192
refinement, **69**
refinement loss, **76**
regression, **14**, 64
Example 3.8, 92
isotonic, 78
multivariate, **202**
univariate, **196**
regression coefficient, **198**

regression tree
    Example 5.4, 151
regressor, **91**
regularisation, **204**, 217, 294
reinforcement learning, **360**
reject, **83**
relation, **51**
    antisymmetric, **51**
    equivalence, *see* equivalence relation
    reflexive, **51**
    symmetric, **51**
    total, **51**
    transitive, **51**
residual, **93, 196**
ridge regression, **205**
Ripper, *see* rule learning systems
ROC curve, **67**
ROC heaven, **69**, 145
ROC plot, **60**
Rocchio classifier, **241**
rotation, **24**
rule, **105**
    body, **157**
    head, **157**
    incomplete, **35**
    inconsistent, **35**

overlap
Example 1.6, 35

rule learning systems
AQ, 192
CN2, 192, 357
Opus, 192
Ripper, 192, 341
Slipper, 341
Tertius, 193
rule list, 139, **157**
Example 6.1, 159

as ranker
Example 6.2, 165
rule set, **157**
Example 6.3, 167

as ranker
Example 6.4, 173
rule tree, **175**
Example 6.5, 175

sample complexity, **124**
sample covariance, **45**
sample mean, **45**
sample variance, **45**
scale, **299**
reciprocal, **300**
scaling, **24**
uniform, **24**
scaling matrix, 201
scatter, **97, 246**
Definition 8.3, 246

reduction by partitioning
Example 8.3, 247
within-cluster, **97**
scatter matrix, **200, 245**, 325
between-cluster, **246**
within-cluster, **246**
scoring classifier, **61**
Scottish classifier, *see* naive Bayes
SE, *see* squared error
search heuristic, 63
seed example, **170**
segment, **36**
semi-supervised learning, **18**
sensitivity, **55**, 57

separability

conjunctive
Example 4.4, 116
separate-and-conquer, 35, **161**, 163
sequence prediction, **361**
sequential minimal optimisation, 229
set, **51**
cardinality, **51**
complement, **51**
difference, **51**
disjoint, **51**
intersection, **51**
subset, **51**
union, **51**
Shannon, Claude, 294
shatter, **126**

shattering a set of instances
    Example 4.7, 126
shrinkage, **204**
sigmoid, **221**
signal detection theory, **60**, 316
significance test, **352**
silhouette, **252**
similarity, 72
    Example 1.1, 15
singular value decomposition, **324**
skewness, **303**
    Example 10.3, 304
slack variable, **216**, 294
Slipper, *see* rule learning systems
slope, **195**
soft margin, **217**
SpamAssassin, 1–14, 61, 72
sparse data, 22
sparsity, **205**
specific, **46**
specificity, **55**, 57
split, **132**
    binary, **40**

splitting criterion
cost-sensitivity
Example 5.3, 144
SQL, *see* query languages
squared error, **73**
squared Euclidean distance, 238
stacking, **339**
standard deviation, **301**

statistic
of central tendency, **299**
of dispersion, **299**
shape, **299**, 303
stop word, **280**
stopping criterion, **163, 310**
structured output prediction, **361**
sub-additivity, 236
subgroup, **100**, 178

evaluation
Example 6.6, 180
extension, **100**
subgroup discovery, 17
Example 3.11, 100
subspace sampling, **333**
supervised learning, **14**, 17, 46
support, **182**
support vector, **211**
support vector machine, 22, 63, 211, 305
complexity parameter, **217**
SVD, *see* singular value decomposition
SVM, *see* support vector machine

t-distribution, **353**
target variable, **25**, 91, 262
task, **13**
terms, *see* first-order logic
Tertius, *see* rule learning systems
test set, **19, 50**
Texas Instruments TI-58 programmable calculator, 228
text classification, 7, 11, 22

thresholding
Example 10.5, 309
total order, **51**
training set, 14, **50**
transaction, **182**
transfer learning, **361**
translation, **24**

tree learning systems
    C4.5, 156
    CART, 156
    ID3, 155
triangle inequality, **236**
trigram, 322
true negative, **55**
true negative rate, **55**, 57
true positive, **55**
true positive rate, **55**, 57, 99
turning rankers into classifiers, 278

underfitting, **196**
unification, **123**
    Example 4.6, 123
unigram, 322
universe of discourse, **51**, 105
unstable, **204**
unsupervised learning, **14**, 17, 47

Vapnik, Vladimir, 125
variance, 24, **45, 94, 149**, 198, **200, 301**, 303
    Gini index as, 148
VC-dimension, **125**
    linear classifier, 126
version space, **113**
    Definition 4.1, 113
Viagra, 7–44
vocabulary, 7
Voronoi diagram, **98**
Voronoi tesselation, **241**

voting
one-versus-one
Example 3.2, 85

Warmr, *see* association rule algorithms
weak learnability, **331**
weighted covering, **181**, 338
Example 6.7, 181
weighted relative accuracy, **179**
Weighted Covering(*D*)
Algorithm 6.5, 182
Wilcoxon's signed-rank test, **354**
Example 12.7, 355
Wilcoxon-Mann-Whitney statistic, 80

χ^2 statistic, 101, **312**, 323
Xbox game console, 129

z-score, **267**, 314, 316

Machine Learning and IoT

A Biological Perspective

Edited by
Shampa Sen
Leonid Datta
Sayak Mitra

CRC Press
Taylor & Francis Group

Machine Learning and IoT

A Biological Perspective

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Machine Learning and IoT

A Biological Perspective

Edited by
Shampa Sen
Leonid Datta
Sayak Mitra

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

© 2019 by Taylor & Francis Group, LLC
CRC Press is an imprint of Taylor & Francis Group, an Informa business

No claim to original U.S. Government works

Printed on acid-free paper

International Standard Book Number-13: 978-1-138-49269-1 (Hardback)

This book contains information obtained from authentic and highly regarded sources. Reasonable efforts have been made to publish reliable data and information, but the author and publisher cannot assume responsibility for the validity of all materials or the consequences of their use. The authors and publishers have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission to publish in this form has not been obtained. If any copyright material has not been acknowledged please write and let us know so we may rectify in any future reprint.

Except as permitted under U.S. Copyright Law, no part of this book may be reprinted, reproduced, transmitted, or utilized in any form by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying, microfilming, and recording, or in any information storage or retrieval system, without written permission from the publishers.

For permission to photocopy or use material electronically from this work, please access www.copyright.com (http://www.copyright.com/) or contact the Copyright Clearance Center, Inc. (CCC), 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. CCC is a not-for-profit organization that provides licenses and registration for a variety of users. For organizations that have been granted a photocopy license by the CCC, a separate system of payment has been arranged.

Trademark Notice: Product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation without intent to infringe.

Library of Congress Cataloging-in-Publication Data

Names: Sen, Shampa, editor.
Title: Machine learning and IoT : a biological perspective / [edited by] Shampa Sen, Leonid Datta, Sayak Mitra.
Description: Boca Raton : Taylor & Francis, 2019. | Includes bibliographical references and index.
Identifiers: LCCN 2018014315 | ISBN 9781138492691 (hardback)
Subjects: LCSH: Biology--Data processing--Methodology. | Bioinformatics--Methodology.
Classification: LCC QH324.2 .M34 2019 | DDC 570.285--dc23
LC record available at https://lccn.loc.gov/2018014315

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

I would like to dedicate this book to my dearest niece and light of my life, Neha

Shampa Sen

To my beloved late grandma Bimalprabha, my father Haranath, my mother Mallika, my uncle Ramanath, and aunt Karuna Datta, who have always been my inspirations in life.

Leonid Datta

Dedicated to all the teachers, my mother and my grandmother, who continue to have a profound effect on my life. Also dedicated to my late grandfather Dr. Bimalendu Majumdar, who would have been very happy to see me reach for the stars

Sayak Mitra

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Contents

Foreword xi
Preface xiii
About the Editors xv
Contributors xvii

Chapter 1 Machine Learning: A Powerful Tool for Biologists 1

Mohd Zafar, Ramkumar Lakshmi Narayanan, Saroj K. Meher, and Shishir K. Behera

Chapter 2 Mining and Analysis of Bioprocess Data 29

Prerna Grover, Aditya Shah, and Shampa Sen

Chapter 3 Data Mining in Nutrigenomics 43

Avipsha Sarkar, Shreyasi Kundu, Shakti Singh, and Shampa Sen

Chapter 4 Machine Learning in Metabolic Engineering 53

Sayak Mitra

Chapter 5 Big Data and Transcriptomics 73

Sudharsana Sundarrajan, Sajitha Lulu, and Mohanapriya Arumugam

Chapter 6 Comparative Study of Predictive Models in Microbial-Induced Corrosion 91

Nitu Joseph and Debayan Mandal

Chapter 7 Application of Data Mining Techniques in Autoimmune Diseases Research and Treatment 113

Sweta Bhattacharya and Sombuddha Sengupta

Chapter 8 Data Mining Techniques in Imaging of Embryogenesis 133

Diptesh Mahajan and Gaurav K. Verma

Chapter 9 Machine Learning Approach to Overcome the Challenges in Theranostics: A Review 145

Bishwambhar Mishra, Sayak Mitra, Karthikeya Srinivasa Varma Gottimukkala, and Shampa Sen

Chapter 10 Emotion Detection System 167

Adrish Bhattacharya, Vibhash Chandra, and Leonid Datta

Chapter 11 Segmentation and Clinical Outcome Prediction in Brain Lesions 181

Sharmila Nageswaran, S. Vidhya, and Deepa Madathil

Chapter 12 Machine Learning Based Hospital-Acquired Infection Control System 193

Sehaj Sharma, Prajit Kumar Datta, and Gaurav Bansal

Chapter 13 No Human Doctor: Learning of the Machine 217

Leonid Datta, Emilee Datta, and Shampa Sen

Chapter 14 The IoT Revolution 229

Adrish Bhattacharya and Denim Datta

Chapter 15 Healthcare IoT (H-IoT): Applications and Ethical Concerns 247

Srijita Banerjee, Adrish Bhattacharya, and Shampa Sen

Chapter 16 Brain–Computer Interface 265

Abhishek Mukherjee, Madhurima Gupta, and Shampa Sen

Chapter 17 IoT-Based Wearable Medical Devices 287

Avitaj Mitra, Abanish Roy, Harshit Nanda, Riddhi Srivastava, and Gayathri M.

Chapter 18 People with Disabilities: The Helping Hand of IoT 299

Ashmita Das, Sayak Mitra, and Shampa Sen

Chapter 19 Smart Analytical Lab 313

Subhrodeep Saha, Sourish Sen, Bharti Singh, and Shampa Sen

Chapter 20 Crop and Animal Farming IoT (CAF-IoT) 327
Neha Agnihotri, Soumyadipto Santra, and Shampa Sen

Index 341

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Foreword

Machine learning and IoT are presently two of the most rapidly developing aspects in the field of computer science. Machine learning deals with the ability of the computers to learn and predict an output without being explicitly programmed. IoT, on the other hand, deals with connecting everything and/or everybody to each other using sensors. Both machine learning and IoT find numerous interdisciplinary applications in various fields, especially biology. The present book deals with how machine learning and IoT complement research in the field of biotechnology and health care. At present, biology is the foremost field generating enormous amounts of data needing to be sorted. These data also needs to be analyzed for a better understanding of the research at hand.

It is often hard for budding researchers to visualize how to bring together the two very contrasting fields of computer science and biology. This book beautifully unifies these two fields and provides an outlook on the ways biology has and can be improved when computer science technique intervenes. While there are many publications in this regard, most of them focus on the already established field of bioinformatics. But there is more to this interdisciplinary wedding than bioinformatics, and those aspects have been covered by the book. The topics chosen in the book are present day hot topics, thus making them relatable to undergraduate students and researchers of both fields alike.

What adds to the glamour of the book is how two of the editors—Leonid Datta and Sayak Mitra—both being undergraduate students, with hardly any experience in their respective core backgrounds of computer science and biology have effectively merged the two fields, while being guided by the senior editor Dr. Shampa Sen. Even the list of editors showcases a colorful array of undergraduate and postgraduate students guided by eminent researchers, along with industrial experts who are constantly updated on the current technologies used in industries through their technological demands.

I highly recommend this valuable and timely book. I heartily congratulate the editors and contributing authors for their immense efforts in producing this book.

Prof. Sarit K. Das
Director
Indian Institute of Technology Ropar

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Preface

The present era of research and development is all about interdisciplinary studies attempting to better comprehend and model our understanding of this vast universe. The fields of biology and computer science are no exception. This book discusses some of the innumerable ways in which computational methods can be used to facilitate research in biology and medicine—from storing enormous amounts of biological data to solving complex biological problems and enhancing the treatment of various diseases.

In 1959, Arthur Samuel defined machine learning as the "field of study that gives computers the ability to learn without being explicitly programmed". Since then, this field of study has undergone many advancements to make machine learning a combination of several disciplines such as statistics, information theory, theory of algorithms, probability, and functional analysis. Another aspect of machine learning is the extraction of common patterns from sets of data, known as data mining. Hence, a more inclusive definition of machine learning would be design and testing of algorithms facilitating pattern recognition, classification, and prediction based on existing data.

Internet of Things (IoT) is based on the concept of various hardware devices, including sensors, being embedded with software and apps able to transmit and process data. These new easy-to-use devices have innumerable applications, especially in the field of biotechnology and medicine. Any object, outfitted with the right sensors, can observe and interact with its environment, allowing hard-working individuals to control their personal and professional responsibilities from far away, using only a voice command in smartphone.

The relationship between biology and machine learning has existed for decades. The present status of research in the field of biotechnology calls for sophisticated methods to store the large amounts of data generated, and to ease the task of the researcher by going so far as to predict certain outcomes. In short, biological research has become much dependent on computational algorithms. Likewise, the management of the health care system also becomes easier with the intervention of machine learning and other computational tools.

In this book, we have started our discussion with how various machine learning algorithms can be used as a powerful tool for aiding biologists in their research work. In the first half of the book, we have talked about how different aspects of machine learning can complement bioprocess engineering, bioinformatics, and biofouling. We have also discussed how health care facilities and treatments can benefit from machine learning, covering a wide range of aspects such as autoimmune diseases, nosocomial infections, brain lesions, and even theranostics. The latter half of the book talks about how IoT can facilitate many aspects of biology, starting from health care systems to analytical labs, even revolutionizing traditional farming practices.

In conclusion, this book gives a comprehensive overview of some of the innumerable ways in which computational methods can aid research in biology and medicine to allow us to live a better life.

Shampa Sen
Leonid Datta
Sayak Mitra

About the Editors

Dr. Shampa Sen earned her PhD in environment from Indian Institute of Technology, Guwahati, India. She is the Associate Professor at School of Bio-Sciences and Technology, VIT, Vellore, India. With extensive experience in academia, she has 60 publications in the fields of biotechnology, drug design, nanobiotechnology, and nutraceuticals. She was actively involved in many professional development activities. Her research interests include biosynthesis of metallic nanoparticles, nanoparticles in biomedical and environmental applications, metabolic engineering, drug design, and computational biology. She is a life member of the Biotech Research Society, India (BRSI) and the Environmental Mutagen Society of India (EMSI), as well as a member of the International Neural Network Society (INNS). She is also a fellow of the Royal Society of Biology.

Leonid Datta has earned his BTech degree in computer science and engineering from VIT, Vellore, India. He has various publications in reputed peer-reviewed international journals and books. His research interests include data mining and machine learning. His most notable project works include designing truncation techniques for a search engine, developing a system that runs based on RFID scanning for attendance purposes or maintaining other records, and automating the processing of big data analysis through development of a portal. He is also a student member of INNS, USA.

Sayak Mitra is currently pursuing his BTech degree in Biotechnology at VIT, Vellore, India. At present, he has publications in the fields of nanoremediation and metabolic engineering in international peer-reviewed journals and conferences. His research interests include biochemical engineering, bioprocess engineering and metabolic engineering. He was awarded Best Oral Presentation Award at the 7th International Science Congress conference (ISCA-2017) held in Bhutan. He is also a fellow under the Khorana Program for Scholars (an Indo-U.S. Science and Technology collaborative research scheme). His avid interest in research motivates him to pursue a PhD in the upcoming years.

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Contributors

Neha Agnihotri
School of Bio Sciences and Technology
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Mohanapriya Arumugam
School of Bio Sciences and Technology
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Srijita Banerjee
School of Bio Sciences and Technology
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Gaurav Bansal
Bank of America
Hyderabad, India

Shishir K. Behera
Department of Chemical Engineering
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Adrish Bhattacharya
Department of Computer Science and Technology
Indian Institute of Engineering Science and Technology
Shibpur, India

and

Department of Electronics and Telecommunication Engineering
Indian Institute of Engineering Science and Technology
Howrah, West Bengal, India

Sweta Bhattacharya
School of Information Technology and Engineering
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Vibhash Chandra
Department of Computer Science and Technology
Indian Institute of Engineering Science and Technology
Howrah, West Bengal, India

Ashmita Das
School of Bio Sciences and Technology
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Denim Datta
CIDSE (School of Computing Informatics and Decision Systems Engineering)
Arizona State University
Tempe, Arizona

Emilee Datta
R G Kar Medical College and Hospital
Kolkata, India

Leonid Datta
Department of Software Systems
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Prajit Kumar Datta
Bank of America
Hyderabad, India

Gayathri M.
Department of Biotechnology
School of Bio Sciences and Technology
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Karthikeya Srinivasa Varma Gottimukkala
Department of Biotechnology
Sreenidhi Institute of Science and Technology
Hyderabad, Telangana, India

Prerna Grover
School of Bio Sciences and Technology
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Madhurima Gupta
School of Bio Sciences and Technology
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Nitu Joseph
School of Mechanical and Building Sciences
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Shreyasi Kundu
School of Bio Sciences and Technology
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Sajitha Lulu
School of Bio Sciences and Technology
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Deepa Madathil
Department of Sensor and Biomedical Technology
School of Electronics Engineering
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Diptesh Mahajan
School of Bio Sciences and Technology
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Debayan Mandal
School of Mechanical and Building Sciences
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Saroj K. Meher
Systems Science and Informatics Unit
Indian Statistical Institute
Bangalore, India

Bishwambhar Mishra
Department of Biotechnology
Sreenidhi Institute of Science and Technology
Hyderabad, Telangana, India

Avitaj Mitra
Department of Biomedical Engineering
School of Electronics Engineering
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Sayak Mitra
School of Bio Sciences and Technology
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Abhishek Mukherjee
School of Information Technology and Engineering
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Sharmila Nageswaran
Department of Sensor and Biomedical Technology
School of Electronics Engineering
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Harshit Nanda
Department of Biotechnology
School of Bio Sciences and Technology
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Ramkumar Lakshmi Narayanan
Department of Information Technology
Sur College of Applied Sciences
Ministry of Higher Education
Sultanate of Oman

Abanish Roy
Department of Biotechnology
School of Bio Sciences and Technology
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Subhrodeep Saha
School of Bio Sciences and Technology
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Soumyadipto Santra
School of Bio Sciences and Technology
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Avipsha Sarkar
School of Bio Sciences and Technology
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Shampa Sen
School of Bio Sciences and Technology
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Sourish Sen
School of Bio Sciences and Technology
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Sombuddha Sengupta
School of Bio Sciences and Technology
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Aditya Shah
School of Bio Sciences and Technology
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Sehaj Sharma
GE Healthcare
Bangalore, India

Bharti Singh
Abbott
California

Shakti Singh
Los Angeles Biomedical Research Institute Harbor-UCLA Medical Center
Torrence, California

Riddhi Srivastava
Department of Biotechnology
School of Bio Sciences and Technology
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Sudharsana Sundarrajan
School of Bio Sciences and Technology
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Gaurav K. Verma
EBS University of Business and Law
Wiesbaden, Germany

S. Vidhya
Department of Sensor and Biomedical Technology
School of Electronics Engineering
Vellore Institute of Technology
Vellore, Tamil Nadu, India

Mohd Zafar
Department of Applied Biotechnology
Sur College of Applied Sciences
Ministry of Higher Education
Sultanate of Oman

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

1 Machine Learning
A Powerful Tool for Biologists

Mohd Zafar, Ramkumar Lakshmi Narayanan, Saroj K. Meher, and Shishir K. Behera

CONTENTS

1.1 Machine Learning (ML): An Overview 2
1.1.1 What is ML? 2
1.1.2 ML versus Other Computing Environments 2
1.1.2.1 ML versus Deep Learning 2
1.1.2.2 ML versus Statistics 2
1.2 Various Approaches to ML 3
1.2.1 Decision Tree (DT) 3
1.2.2 Artificial Neural Networks 4
1.2.3 Deep Learning 8
1.2.3.1 Deep Learning Algorithm 9
1.2.4 Support Vector Machines 9
1.2.5 Clustering 10
1.2.6 Bayesian Networks 10
1.2.7 Mining Methods 12
1.3 Data Mining 13
1.3.1 Knowledge Discovery Process 14
1.3.2 Classification 15
1.3.3 Clustering 15
1.3.4 Regression 16
1.3.5 Outlier Analysis 16
1.4 Big Data 17
1.4.1 Why Big Data? 17
1.4.2 Big Data Implementation Framework 17
1.4.3 MapReduce 18
1.4.4 Big Data and Biology 19
1.4.4.1 Cluster Computing in System Biology 20
1.4.4.2 Grid Computing in System Biology 20
1.4.4.3 General Purpose Graphic Processing Units (GPGPUs) Computing in System Biology 20
1.4.4.4 Cloud Computing in System Biology 20
References 22

1.1 MACHINE LEARNING (ML): AN OVERVIEW

1.1.1 What is ML?

ML is a method of data analysis dealing with the construction and evaluation of algorithms. It is the science that gives computers and computing machines the ability to act without being explicitly being programmed. It is defined by the ability to choose effective features for pattern recognition, classification, and prediction based on the models derived from existing data (Tarca et al. 2007). A suitably programmed computing machine is required to perform the aforementioned tasks with the help of an efficient data analysis algorithm in such a way that the classifier itself is highly mechanized without the involvement of human input.

1.1.2 ML versus Other Computing Environments

Data mining is an advanced technique used for data exploration in these domains and is broadly categorized into descriptive and predictive approaches. Under the descriptive approaches, the interesting patterns (i.e., relations) in the dataset can be identified and clustered into meaningful groups. The predictive approaches refer to supervised learning which establishes the relationship between input (independent) variables and target (dependent) variables through a structural presentation, that is, a model. Among different ML algorithms, a supervised learning algorithm includes a set of mathematical instructions which can derive a typical, high-dimensional model in the form of input-output relationships. The developed supervised learning model help in identification of mapping of input variables to output variables based on a given example of joint observations of the values of these variables (Geurts et al. 2009).

Data science compares and overlaps with many related fields such as ML, deep learning, artificial intelligence, statistics, operations research, and applied mathematics. ML is largely a hybrid field, taking its motivation and practices from both computational science and mathematics. It is the core part of both data mining and predictive analytics.

1.1.2.1 ML versus Deep Learning

The two main supervised models of ML are classification and regression. The regression model is used to predict the demand of a given product in relation to its characteristics. However, the classification model maps the input variables into predefined classes. The widely used classifiers under ML approaches are support vector machine (SVM), artificial neural network (ANN), and decision trees (DT). Deep learning is referred to as the intersection between ML and artificial intelligence. It is a subfield of ML focusing narrowly on a subset of ML tools and techniques supported by neural network inspired algorithms.

1.1.2.2 ML versus Statistics

The main focus of ML is the study and design of systems which can learn from data. On the other hand, statistical modeling represents the mathematical equation revealing the relationships between variables of a data set. ML is all about predictions, classification, regression, forecasting, clustering, and association models, whereas

statistics is concerned about sample, population, and hypothesis testing. ML techniques and approaches completely rely on computing power. On the other hand, statistical techniques can be developed where computing power is not visualized as an option.

1.2 VARIOUS APPROACHES TO ML

1.2.1 Decision Tree (DT)

The application of DTs is very popular in biological data mining because of its simplicity and transparency (Table 1.1). The constructed DT is self-explanatory in nature and can efficiently explain and interpretate the output result without the help of a data mining expert.

The structure of a DT has much resemblance in real life, as well as in artificial intelligence life. It is an intersect between artificial intelligence and statistics, which can be used to build the predictive model from the observation of a system (Geurts et al. 2009). In its structure, the input and output variables are known as attributes or features, while joint observations of these values are called objects. The given example of objects, which is supposed to be used as a model, is known as a learning sample.

A DT is made up of various nodes, which can appear in the form of rooted tree having no incoming edges. Besides, the other nodes comprise exactly one incoming edge. A node with outgoing edges is called an "internal" or "test" node, whereas all other nodes are known as "leaves" (also called "terminal" and "decision" nodes) (Figure 1.1). As per the specific discrete function of the input variables, each internal node can split the case space into two or more subspaces.

TABLE 1.1
Application of DT in Biological Science

Type of DT	Study Focus	References
Genetic algorithm based DT and neural network	Prediction of benthic macroinvertebrates	D'heygere et al. (2003); D'heygere et al. (2006)
Genetic algorithm based DTs	Prediction of microRNA target genes	Behzad et al. (2015)
Wavelets and genetic algorithm	Extraction of cancer data classification using mass spectrometry	Nguyen et al. (2015)
DT	DT for binding of dipeptides to the thermally fluctuating surface of cathepsin K	Nishiyama (2016)
Genetic algorithm based DTs	Ensemble model (iACP-GAEnsC) proposed for classification of anticancer peptides	Akbar et al. (2017)
Discrete bacterial algorithm based DTs	Selection of feature in classification of microarray gene expression cancer data	Wang et al. (2017b)

FIGURE 1.1 Typical diagram of DT.

The complexity of a DT depends on available variables in data and the total number of nodes, leaves, and tree-depth. It can be controlled by using a suitable pruning method and stopping criteria (Geurts et al. 2009).

The conventional greedy inducers such as ID_3, C4.5, and CART (Rokach and Maimon 2008) are widely used in DT, which help in construction of a DT in recursive manner, that is, divide and conquer. The greedy nature of DT induction usually fails to elucidate underlying relationships between the datasets and endpoints (output). However, using evolutionary programing, the induction of a DT is more significant than the induction by recursive portioning. It has been reported that the evolutionary programming induced DTs to exhibit a significant accuracy on previously unseen data (DeLisle and Dixon 2004). They have produced less complex classifiers with an increase in predictive accuracy of 5% to 10% over the traditional method (DeLisle and Dixon 2004). Evolutionary Algorithm (EA) is a search heuristic mimicking the process of natural biological evolution. During induction using EA, a collective learning process within the selected population of individuals is randomly initiated. Using the EAs, the problem of local minima in the search space can be overcome by randomization process (Zafar et al. 2010). Multiobjective algorithm comprises different objectives in aggregated form and combined into one objective using a fixed weight. Ant colony optimization (ACO) algorithm is another heuristic-based algorithm successfully applied to extract classification rules, and DT induction in unexplored research fields. Bacterial-based optimization algorithms with embedded weighted feature selection strategy has been developed with the aim to reduce the feature dimension in classification. The feature selection algorithm-based optimization processes possess low computational complexity and improve search ability, even in discrete optimization problems (Wang et al. 2017a).

1.2.2 Artificial Neural Networks

The inspiration for computing with ANNs came from various studies on biological neurons. The major components of a neuron are shown in Figure 1.2. They consists of

FIGURE 1.2 Typical biological neuron. (From Rene, E. R., "Removal of volatile organic compounds using compost biofiters." PhD Thesis, Submitted to the Dept Chem. Eng., IIT Madras, 2005.)

dendrites, soma, axon, and synapses. While dendrites are the receptive zones (input surface of the neuron), the axon is the transmission (output) line. Synapses connect the axon of one neuron to various parts of other neurons. When the input signals come in contact with these synapses, it results in local changes in the input potential in the cell body of receiving neurons, which eventually spread through the main body of the cell, and are then weighted.

ANNs possess neurons joined together by variable strength connections (synaptic weights) to form a highly interconnected information system, which helps in the modeling and control of nonlinear systems (Hornik et al. 1989).

The input signals from an artificial neuron (Figure 1.3) flow through a gain or weight called synaptic weight, whose function is analogous to that of the synaptic junction in a biological neuron. The weights can be positive or negative depending on the acceleration or inhibition of the flow of signals in a biological cell. The summing node accumulates all the input weighted signals, adds a bias signal, and then passes to the output through the activation function. The most commonly used activation function is the sigmoidal function also called the log-sigmoid or logistic function, which is defined by:

$$f(x) = \frac{1}{1+e^{-x}} \tag{1.1}$$

A generalized example of a sigmoid function with inputs ranging between -1 to $+1$ is shown in Figure 1.4.

The interconnections of these neurons results in an ANN, and its prime objective is to emulate the human brain and to solve scientific, engineering, and many other real life problems. Various neural network models have been developed in the

FIGURE 1.3 Schematic of an artificial neuron.

recent past due to the fuzzy understanding about the interconnections between the biological neurons. These networks are generally classified as feed forward and recurrent types. Of considerable interest is the feed forward network, where signals from one neuron to another flows only in the forward direction. Some of the common network models developed are perceptron, back propagation networks, radial bias function networks, modular neural networks, etc. (Haykin 1994). Many ANN topologies have been proposed (Widrow and Sterns 1985; Rumelhart et al. 1986; Kohonen 1988).

A feed forward multilayer perceptron (MLP) consists of two or more layers of processing elements, which are linked by weighted connections. Neural networks can be thought of as "black box" devices, which accept inputs and produces the desired outputs. The architecture of a three layered ANN with four input and two output parameters is shown in Figure 1.5. It consists of (1) an input layer receiving

FIGURE 1.4 Sigmoid transfer function.

FIGURE 1.5 Basic structure of a multilayered perceptron (χ_1–χ_4: input vector; y_1, y_2: output vector).

information from external sources, and passing this information to the network for processing; (2) a hidden layer receiving information from the input layer and processing them in a hidden way; (3) an output layer receiving processed information and signaling the desired output; and (4) a bias acting on a neuron like an offset. The function of the bias is to provide a threshold for the activation of neurons. The bias input is connected to each of the hidden and output neurons in a network. The network is operated in two distinct phases called training and testing.

More commonly, ANNs have been widely studied and successfully employed in diverse applications, such as pattern recognition (Mah and Chakravarthy 1992), fault detection (Hoskins and Himmelblau 1988; Sharma et al. 2004), weather forecasting (Foster et al. 1990; Ramírez et al. 2005), and image analysis (Nattkemper 2004). In addition to this, ANNs have also been widely used for prediction purposes (Maier and Dandy 2000; Al-Shayji and Liu 2002; Zhang et al. 2002; Sözen et al. 2005).

For simplicity, and since the connection weights are the most influential variable, Rumelhart et al. (1986) proposed the back error propagation (BEP) training algorithm, which uses the gradient descent technique to adjust the weights in which the global error function, E, is minimized by modifying the weights using the Equation 1.2:

$$\Delta W_{ji} = -\eta \frac{\partial E}{\partial W_{ji}} \tag{1.2}$$

where, ΔW_{ji} is the weight increment from node i to node j; and η is the learning rate, by which the size of the step taken along the error surface is determined.

The weights are then updated by adding the delta weight, ΔW_{ji}, to the corresponding previous weight as follows:

$$W_{ji}(n+1) = W_{ji}(n) + \Delta W_{ji}(n+1) \tag{1.3}$$

TABLE 1.2
Applications of ANNs in the Field of Biotechnology

Type of Neural Network	Study Focus	References
Deep neural network	Cell mitosis detection	Zhou et al. (2017)
ANN	Assessment of organic and inorganic contaminants in agricultural soils	Bonelli et al. (2017)
ANN-GA	Optimization of wheat germ fermentation condition	Zheng et al. (2017)
MLP NN	Graft survival prediction in liver transplantation	Raji and Vinod Chandra (2016)
Feed forward NN	Prediction of bioethanol production from sugar beet processing waste	Grahovac et al. (2016)
MLP NN	Prediction of volatile organic compound removal in bioreactors	López et al. (2017)
Back propagation NN	Prediction of methane percentage in biogas recovered from a landfill	Behera et al. (2015)

Where, $W_{ji}(n)$ = the value of a weight from node i to node j at step n (before adjustment); and $W_{ji}(n + 1)$ = the value of the weight at step (n + 1) (after adjustment). First of all, the weights between the hidden layer and the output layer are adjusted and then the weights between the hidden layer and the input layer are adjusted. The choice of the learning rate is determined by trial-and-error. Rumelhart et al. (1986) described a process that adds a momentum term (μ) to the weight adjustment. The momentum term may be considered to increase the effective step size in shallow regions of the error surface (Hassoun, 1995) and can speed up the training process. Once an adjustment is carried out, it is saved and used to modify all subsequent weight adjustments. The modified adjustment of the delta weight, ΔW_{ji}, is given in Equation 1.4

$$\Delta W_{ji} = -\eta \frac{\partial E}{\partial W_{ji}} + \mu \Delta W_{ji} \tag{1.4}$$

The above process is repeated, and once the desired learning is achieved, the weights are fixed and the neural network can be used in practice. Some typical examples of the application of neural networks for prediction purposes in the field of biotechnology are given in Table 1.2.

1.2.3 Deep Learning

Deep learning process enables the computer to build a complex concept using many small concepts (related to various biological activities, e.g., transcription, translation, replication, etc.) to solve the associated problem in the biological world. The deep learning model is based on a feedforward deep network, which is also known as an

MLP. In order to resolve the difficulties of data expression profile in the deep learning process, the desired complicated map is broken into different nested series of simple maps and represented in different layers of a model.

1.2.3.1 Deep Learning Algorithm

Deep learning is a collection of new techniques including the recent class of ML applied in biology, health care, and drug discovery. After 2012, deep learning received more attention in computational biology (Rampasek and Goldenberg 2016). Recently, the deep learning-based methods became the state-of-the-art in prediction of various regulatory roles and expression pattern identification directly from DNA sequences (Park and Kellis 2015; Angermueller et al. 2016) (Table 1.3).

1.2.4 Support Vector Machines

SVM is a relatively new supervised ML statistical approach, which can be used for both the classification and regression challenges. SVM models the situation

TABLE 1.3
Application of Deep Learning in the Field of Biology

Type of Deep Learning	Study Focus	References
Deep learning	Prediction of sequence specific DNA- and RNA-binding proteins	Alipanahi et al. (2015)
Deep learning algorithm framework deepSEA	Prediction of eQTLs and disease associated variants from large scale chromatin profile data	Zhou and Troyanskaya (2015)
Deep learning	Prediction of single-cell DNA methylation	Angermueller et al. (2016)
Deep convolutional neural network (CNNs)	Prediction of regulatory code of genome wide SNPs	Kelley et al. (2016)
Deep convolutional neural network—deepLoc	Automated classification of protein subcellular localization	Kraus et al. (2017)
Deep learning—neural network	Protein subcellular localization classification from throughput microscopy images	Pärnamaa and Parts (2017)
Gene network (GNet)—LMM	Modeling and prediction of expression QTLs	Rakitsch and Stegle (2016)
Convolutional neural network based deep learning	Identification of Autism Spectrum Disorder (ASD) patient	Heinsfeld et al. (2018)
Deep PSL using stacked auto—encoder (SAE)	Prediction of human protein subcellular localization	Wei et al. (2017)
Convolutional neural network (CNN) based deep learning	Prediction of cancer using RNA sequence data sets	Xiao et al. (2018)
Deep transfer learning	Prediction of membrane protein & 3-D structure	Wang et al. (2017b)

TABLE 1.4
Application of SVM in the Field of Biological Science

Type of SVM	Study Focus	References
SVM Kernel	Prediction of DNA- and RNA-binding protein	Shao et al. (2009)
Hybrid fuzzy support vector regression	Quantitative prediction of peptide binding affinity	Uslan and Huseyin (2016)
SVM	Analysis of protein-protein interactions related to diabetes mellitus	Vyas et al. (2016)
SVM based gDNA-Prot	Prediction of DNA binding protein	Zhang et al. (2016)
SVM with nested cross-validation method	Protein–sugar binding site prediction	Banno et al. (2017)
SVM	Cancer subtype prediction based on integrated gene expression and protein network	Hung and Chiu (2017)
SVM using wavelet denoising	Prediction subcellular localization of gram-negative bacterial proteins	Yu et al. (2017)

by creating a feature space, which is a finite dimensional vector space. The fundamental principle of SVM is to separate the feature space into two classes by finding the hyper-plane differentiating the two classes very well. SVM methods are robust to study the interaction among common variants and rare variants in a small sample family. An efficient and effective method for SVM classifier using a coarse-grained parallel genetic algorithm (CGPGA) has been developed by Chen et al. (2016). CGPGA can be widely used to jointly select the feature subset and optimize the parameters for SVM in many practical applications of biological science (Table 1.4).

1.2.5 Clustering

Cluster analysis is a data explorative technique aiming to divide a multivariate dataset into clusters (groups) based on a set of measured variables in same group, which have similar subjects. Cluster analysis played an important tool in data analysis in a wide variety of fields including medicine, pharmaceutical science, biological science, statistics, and computer and information science. The classical k-means algorithm have efficiency in clustering large data sets and is used extensively in many applications (Milone et al. 2013; Wang et al. 2014). Several heuristic methods have been used to improve the accuracy and efficiency of the k-mean clustering algorithm for clustering the biological data (Wang et al. 2014) (Table 1.5).

1.2.6 Bayesian Networks

Bayesian networks (BNs) are probabilistic graphical distributions of a set of data with a possible mutual causal relationship also known as belief networks. This is one of the most popular supervised ML approaches based on the probabilistic classifier.

TABLE 1.5
Application of Clustering in the Field of Biological Science

Type of Clustering	Study Focus	References
Kohonen self organizing maps (SOP)	Clustering biological data: topology preservation in nonlinear dimension reduction	Milone et al. (2013)
Parallel clustering algorithm	Clustering large-scale biological data set	Wang et al. (2014)
Multiple kernel clustering (MKC) algorithms	Clustering with corrupted kernels data in biological applications	Li et al. (2017)
New hierarchical clustering algorithms	Prediction of coexpressed genes based on new cluster validation indices	Pagnuco et al. (2017)
Particle swarm clustering	Comparison and clustering of basal cell carcinoma (BCC), squamous cell carcinoma of the skin (SCC), and actinic keratosis (AK)	Poswar et al. (2017)

The network consists of nodes (each node in the graph represent a random variable) representing the probabilistic dependencies among the corresponding random variables (Figure 1.6). The conditional dependence between the nodes in the graph can be estimated by using other known statistical and computational methods.

Due to the complexity of learning a globally optimal network structure, vague empirical algorithms based on overlapping swarm (OS) intelligence (a modification of particle swarm optimization) have been developed to address the inference, structure-learning, and parameter estimation in BNs. The computational complexity of the

Gene 1	Gene 2 = 1 | Gene 1
−1	0.1
0	0.2
1	0.7

FIGURE 1.6 Baysian network example. (From Djebbari, A., Quackenbush, J., *BMC Systems Biology* 2, 2008: 57.). Here, each random variable corresponding to a gene showing three states corresponding to its transcriptional response: (i) −1 for under-expressed, (ii) 0 for unchanged, and (iii) +1 for overexpressed. The conditional probabilities for Gene 2 indicating the likelihood that Gene 2 is up-regulated given the transcriptional state of Gene 1.

TABLE 1.6
Application of BNs in the Field of Biological Science

Type of SVM	Study Focus	References
BNs-based Gaussian process regression (GPR) model	Process modeling and optimization	Yan et al. (2011)
Bayesian inference with support vector regression (SVR)	Soft sensor predictions in batch bioprocess	Yu (2012)
BNs-based Markov Chain Monto Carlo (MCMC) algorithm	Prediction of cancer related gene in acute myeloma leukemia cases	Wang et al. (2015)
Standard clustering techniques: PCA—BNs	Differential gene expression and gene ontology analysis of microarray data	Lowes et al. (2017)

Bayesian classifier learning structure can increase exponentially with the increase of descriptive variables in task. This can be overcome by learning of Bayesian classifier structure with the use of sophisticated algorithms such as genetic algorithm (Table 1.6).

1.2.7 Mining Methods

Data mining is the process of scrutinizing large database in different fields to identify specific patterns and to establish relationships among datasets. This is the computational process of analyzing big data to extract the pattern and get useful information (Gullo 2015). The widely used data mining techniques are induced learning based where a model is constructed explicitly or implicitly by generalization from the training of a data set. In the knowledge discovery database (KDD) process, data follows different phases like selection, preprocessing, transformation, and mining (Rokach and Maimon 2008; Gullo 2015). Data mining is a central step in the overall KDD process (Figure 1.7). The objective of data mining is to reduce the complexity of datasets and to extract the useful information from a large dataset to its maximum extent (Rokach and Maimon 2008).

Data mining task can be both predictive and descriptive in nature. In the predictive task, a model is build to predict the future behavior or certain features, whereas in the descriptive task the built model describes the data in an efficient and easily

FIGURE 1.7 The KDD process. (From Gullo, F., *Physics Procedia* 62, 2015: 18–22.)

TABLE 1.7
Applications of Different Data Mining Methods in Biological Science

Data Mining Methods	Study Focus	References
ROSEFW-RF map reduce approach—random forest model	Large scale (big data) analysis of bioinformatics classification problem	Triguero et al. (2015)
DT and k-nearest neighbor algorithm	Improve the prediction accuracy of DNA variants identification in biological big data	Farid et al. (2016)
DAMIS (cloud technology) web based	Data mining classification, clustering and dimension reduction in biological datasets	Medvedev et al. (2017)
Graingenes—database for data mining	Inflow of data into meaningful metadata; curration of gene, QTLs, and genomic elements	Odell et al. (2017)
Classification, and clustering techniques	Discover valuable patterns as the process ontology	Khanbabaei et al. (2018)

understandable way. The descriptive tasks can perform the data characterization, data discrimination, association-rule discovery from data, and data clustering. The association analysis is used in computational biology for DNA microarray analysis, including monitoring of expression of thousands of genes or an entire genome (Table 1.7).

1.3 DATA MINING

Advancement of modern technology in medical science and biotechnology has acquired a great scale of biorelated data, among others. These data demand for thorough analysis in order to extract possible interesting pattern/information, which can be useful for various studies and investigations. Data mining is the computing process of extracting nonobvious and hidden information from a largely available data warehouse, which involves ML, statistical analysis, and database systems. As the name indicates, data mining is not to mine (extraction) the data itself but the discovery of knowledge from it. The apparent task here is to establish the association of huge amount of biorelated data and data mining methodologies in order to develop algorithms/models to solve complex biological problems. This section provides a general overview of some open challenges in data mining approaches motivating the development of new and effective tools, which can be used for the analysis of biorelated data sets.

Data mining is one of the important steps in the overall knowledge extraction/discovery process. The term knowledge discovery process (KDP) often gets confused with terminologies like knowledge discovery in databases, knowledge discovery, data mining, information extraction process, etc. However, researchers consider them as the synonyms and use them based on the convenient domain and its applications.

1.3.1 Knowledge Discovery Process

While executing the task of knowledge discovery for a particular decision-making problem, one has to perform some predefined steps. The number of steps usually varies based on models described by the developer, but mostly depends on the particular application. However, the present discussion is limited to some popular literature (Fayyad et al. 1996; Anand and Buchner 1998; Cabena et al. 1998; Cios et al. 2000) which has been employed successfully in several knowledge discovery tasks. All these models are often iterative in nature for certain steps initiated by a revision process, as per the requirement. In this direction Fayyad et al. (1996) proposed a primitive model consisting of nine-steps performed in succession. In this process, each step takes as input the information obtained as output from the successful completion of the preceding step. The graphical representation of the model proposed is illustrated in Figure 1.8.

In the step 1 (Figure 1.8), the KDD aims to understand the domain of application. In addition, the KDD tries to gather the prior information of the task and spot its prime objective(s). The second step performs the selection operation to figure out the most suitable data points from the available set. To some extent, variables/attributes of these points are also scrutinized in this step. In the third step, some preprocessing operations, such as cleaning and quality assessment of data points are carried out. This step also addresses the issue of missing data variables. The fourth step provides an appropriate representation of data points based on the requirement in hand. The transformation operations for data representation makes the feature reduction task more convenient and provides a feature subset, which cut down the computational burden incurred by the model. In the fifth step, the task is to make a suitable coordination of the goals of a problem at hand with the data mining technique. A complete analysis of the model is performed in the sixth step, and subsequently a hypothesis selection criterion is considered here. The most important component of the KDD process, called data mining operation, is performed in the seventh step for discovering interesting patterns of interest. The eighth step is to provide a proper

FIGURE 1.8 Schematic block diagram of the knowledge discovery process. (From Fayyad, U. et al. "*Advances in Knowledge Discovery and Data Mining*." AAAI Press, Cambridge, 1996.)

reasoning of the discovered information from the preceding step. One may go for the visualization of extracted information for a deeper understanding and further processing. Once the discovered information is obtained in the form of knowledge, the ninth step acts on it and provides the directions for possible utilization of the information.

The above-mentioned KDD model is then revised to come up with reduced steps of operation, such as eight, six, and five steps. In this process, Anand and Buchner (1998) described an eight step-based model, Cios et al. (2000) presented a six step-based model, and Cabena et al. (1998) proposed a five step-based model. Although there are similarities in these models each of them has its advantages and disadvantages, and their suitability also depend on the application area and the purpose of their use.

1.3.2 Classification

Classification is one of the most important decision-making tools in data mining where the aim is to craft some rules in order to get a "nearly approximated answer" for a particular question. The answer may be a decision, predicting a behavior, the outcome of a process, or an information, which can act as an input to the next step of reasoning. The term "nearly approximated answer" indicates that the classification method normally finds a near optimal (based on the strength/efficiency of the rules) solution—out of many—for a particular problem.

At the beginning, the developer has to collect some prior information in the form of training samples, which should have a set of predefined meaningful attributes, including the most appropriate outcomes normally called class label/ground truth. The objective is to develop a hypothesis mapping the set of input attributes to an outcome. Once the classification model is developed, it is validated with an unseen data set, which comprises the same attributes as the training data set, but without class labels. In the operational step, the hypothesis or the algorithm analyzes the input variables and yields its prediction. The accuracy level of this prediction tells the goodness of the algorithm.

The principle of classification techniques is used for various activities. For example, financial forecasting in stock market might use a classification model to decide which stock is good to buy or sell. Medical practitioners might use this technique to get a solution for health issues. Although this approach is very convenient and easy to realize, it poses several challenges for many real-time problems. The model demands for a large number of authenticate training samples to estimate the model parameters accurately, and they should contain precise information with less ambiguity, uncertainties, and redundancy. Additionally, most of the real-time data do not have label information, which restrict their use for classification models. This particular challenge can be handled by clustering algorithms, as discussed in the following section.

1.3.3 Clustering

Clustering is the method of organizing a set of samples and identifying its subsets in such a way samples in a subset are similar to each other and dissimilar with the

samples of other subsets. Clustering is an *unsupervised learning* phenomenon, where its goal is to discover a *structure* in the set of data points without class labels. In other words, the objective here is to partition the whole data set into nonoverlapping groups with the samples of similar behaviors. Let us understand this process with an example. Consider a deck of 52 numbers of *playing cards and the task is to group them using the clustering method. This particular problem can have many solutions because we can group the whole deck into many ways with similarly behaved cards. For example, a group of two (color-based), or four (shape-based), or thirteen (number-based) cards. Again, among each of these groups there may be several possible solutions. For example, in a two group-based solution, one may have groups with color and another with image. These possibilities lead to another task of finding the optimum one. These issues end up indicating that this task, although looking simple, is the most difficult one.* It can be proved that there exists no absolute "best" criterion, which would be independent of the final aim of clustering. Therefore, the developer—in concurrence with the user—has to provide a criterion of interest so that the result of the clustering will suit their needs. These criteria are broadly categorized into four types, such as connectivity-based (hierarchical clustering), centroid-based, distribution-based, and density-based clustering.

1.3.4 Regression

Regression analysis is a statistical technique for predictive modeling, which investigates and discovers the mapping between input and output variables. Input and output variables are termed as independent (or predictor) and dependent (or target variables), respectively. In a regression analysis, the number of dependent variables is normally less than the independent variables. This analysis is useful for naturally occurring variables rather than the manipulated variables obtained through an experiment. In addition, regression analysis is not suitable for investigating the causal relationships of variables. For that reason, it is logical to state that a variable "I predicts J," but not to state "I causes J." Regression analysis is mostly used for time series modeling, forecasting, and finding the appropriate relationship between the variables. With this method, one can handle the task of curve fitting, where the requirement is to estimate the occurrence of data points so that the predicted curve goes along with data points. Advantages of using regression analysis includes: discovering/indicating probable relationships between predictor and target variables, and estimating the level of influence of predictor variables on target variables. Similar to the classification, regression analysis also passes through two basic steps, such as training and testing. With the available training data points the regression model is developed, which is normally used to predict the dependent variables of the test data points through their independent variables.

1.3.5 Outlier Analysis

An outlier is an event/object/observation not conforming to an anticipated example or sample, and appearing to deviate noticeably from others in a data set, as if it

were generated by a different mechanism. Most of the outlier analysis approaches remove the outlier with an assumption it is a noise. These types of samples do not hold on the conventional statistical explanation of an outlier as a rare object, and the outlier detection methods would unable to detect them. The existence of an outlier can have several reasons, for example, measurement or execution error, human error, natural deviations, or contamination with elements from outside the data set under consideration. Of late, outliers are considered as bad samples in a set of good samples, which are normally due to an incorrect assumption or the result of incorrectly coded experiments. Under such situations, the outliers should be discarded from the analysis. The objective of an outlier analysis method is to detect, rectify, or reduce the influence for reasonable analysis. However, in many cases it was observed that the consequences of the outlier removal result in the loss of important hidden information. These aspects thus alert us to handle the outliers in a strategic manner.

1.4 BIG DATA

In today's technological development, there are numerous transactions of data taking place every second, which are not in a specific format. This gives rise to a huge semistructured or unstructured datset. Big data is a very large, loosely structured dataset defining traditional storage. With the features available it is possible to store biology data repositories, data and back-ups about genes, proteins, and small molecules. The basic idea is to use the big data techniques in processing of unstructured data; performing biological analytics will solve the issues faced due to traditional data mining and processing (Bhosale and Gadekar 2014).

1.4.1 Why Big Data?

The necessity of big data lies in the requirement of different applications to process large volumes of data. The institutions need to employ data-driven decision-making to provide better results. Processing and integration of large volumes of data should make it a valid data, providing both panoramic and granular views to support the decision-making. It is accomplished with the support of big data using affordable computational and storage resources (Tekiner and Keane 2013). The challenges of handling large quantities of data when combining datasets across different archives, and cross-referencing datasets with patient records, treatments, and outcomes can be handled by analysis techniques using two primary methods of processing data, namely, cloud-based computing and big data (Lakshminarayanan et al. 2013).

1.4.2 Big Data Implementation Framework

There are wide varieties of big data implementation frameworks, and examples of famous open source frameworks are Hadoop, Flink, Apache Storm, and ApacheSamza (Matei et al. 2016). Hadoop is an open source implementation framework of Google Distributed computing. It uses a MapReduce pattern for processing data and can

TABLE 1.8
Functionalities of Different Components of Hadoop Framework Used for Big Data Implementation

Project	Functionalities of the Project	References
Ambari™	A web based tool to view the Pig, Hive and MapReduce applications	http://ambari.apache.org/
Avro™	Implementation for data serialization using Python, PHP, and most popular languages	http://avro.apache.org/
Cassandra™	A NoSQL database management system to handle large amount of data across many servers	http://cassandra.apache.org/
Chukwa™	A data collection system for managing large distributed systems	http://chukwa.apache.org/
HBase™	A structured data storage for large tables supported by a scalable distributed database	http://hbase.apache.org/
Hive™	To provides data summarization and ad hoc querying in a data warehouse infrastructure	http://hive.apache.org/
Mahout™	A Scalable ML and data mining library	http://mahout.apache.org/
Pig™	Apache Pig is an extension of Hadoop and is a high-level data processing language by maintaining scalability and reliability of Hadoop framework	http://pig.apache.org/
Spark™	Spark is a flexible in-memory framework that allows it to handle batch and real-time analytic and data processing workload with Hadoop data	http://spark.apache.org/
Tez™	Tez is a low level, distributed execution framework targeted towards data processing applications	http://tez.apache.org/
ZooKeeper™	ZooKeeper is a fault tolerant group of service across Hadoop cluster for configuration management and administration providing distributed synchronization	http://zookeeper.apache.org/

handle large volumes of data (http://hadoop.apache.org). It is associated with different components such as Ambari™, Sqoop, Avro™, Cassandra™, Chukwa™, HBase™, Hive™, Mahout™, Pig™, Spark™, Tez™, and ZooKeeper™. The most widespread components are Hadoop Distributed File System (HDFS) and MapReduce (Table 1.8).

1.4.3 MapReduce

Hadoop MapReduce is the processing component of Apache Hadoop, which processes data sets in a parallel, distributed system. The operations of MapReduce include: specify computation in terms of Map and Reduce function; parallel computation across large-scale clusters of machine; handling machine failures and performance issues; and ensuring efficient communication between the nodes. The key reason to perform mapping and reducing is to accelerate the execution of a specific process through splitting the process into a number of tasks, thereby enabling parallel work (Figure 1.9).

The map execution process consists of many phases for mapping, partitioning, shuffling, sorting, and reduction (Jiang et al. 2010). During the mapping phase, the

FIGURE 1.9 MapReduce operations.

assigned inputs are read from HDFS, followed by the processing of inputs into records (key/value pairs). The map function is applied to each record and the information is stored in MasterNode, at the end of the phase. In the partition phase, the reducer receiving each of the outputs is determined by each mapper. The number of partitions is equal to the number of reducers. The input data is fetched from all map tasks for the portion corresponding to the reduced task's bucket in the partition phase. In the sorting phase, the merge-sorts all map outputs into a single run. In the next phase, a user-defined reduction function is applied to the merged run and the output file is received in HDFS.

1.4.4 Big Data and Biology

Nowadays, biology is considered as "big-data science" with the recent advancement in high-throughput experimental technologies in last 20 years (Altaf-Ul-Amin et al. 2014). The hierarchy shown in Figure 1.10 summarizes the major types of molecules studied in system biology.

The significant advancement in next generation sequencing (NSG) and single-cell capture (SCC) technologies have increased interest in single-cell studies and thus enabled the generation of huge amounts (several Gigabytes) of sequencing data from a single-cell run (Yang et al. 2017). These samples of big data are generated by a number of large consortia including the Encyclopedia of DNA elements (ENCODE) project (www.encodeproject.org), the GTeX project (www.gtexportal.org), and LINCS program (www.lincsproject.org) (Dolinski and Troyanskaya 2015). Scalability and validation are the two fundamental issues, which must

FIGURE 1.10 Hierarchy of different bio-molecules of omics technology at various level of organization in a biological system. (From Altaf-Ul-Amin, M. et al. *BioMed Research International* 2014, 2014: Article ID 428570.)

be kept in mind during running and designing data analysis in system biology. Software scalability is the challenge since the early days of system biology due to, for example, the high algorithmic complexity associated with multiple sequence alignment (Yang et al. 2017). Several efforts towards making a suitable computing environment for biological big data have been made with the development of new hardware technologies such as cluster computing, grid computing, graphical processing unit (GPU) technology, and cloud computing (Altaf-Ul-amin et al. 2014; Yang et al. 2017).

1.4.4.1 Cluster Computing in System Biology

Cluster computing is an early attempt to scaling the biological big data by networking individual computers into clusters through formation of a parallelized distributed-memory machine. For developing cluster-based software environments for big data analysis in system biology, communications protocols and software tools such as message passing interface (MPI) and parallel virtual machine (PVM) have been widely exploited (Table 1.9).

1.4.4.2 Grid Computing in System Biology

Grid computing forms a massively distributed high performance environment for handling big data in biology. Jacq et al. (2004) have reported their experience in the deployment of bioinformatic grid applications within the framework of the DataGrid project. The performed grid computing tests showed how resources (e.g., web portal, update, and distribution of biological databases) sharing improved the practices of bioinformatics (Table 1.9).

1.4.4.3 General Purpose Graphic Processing Units (GPGPUs) Computing in System Biology

GPGPUs are most popular among the scientific community as they allow more intensive investigations of biological systems with a simultaneous reduction of running time (Nobile et al. 2017). Different GPU-powered CUDA-based software tools have been developed for different computational analyses, for example, sequence alignment, molecular dynamics, molecular docking, prediction and searching of molecular structures, simulation of temporal dynamics of cellular processes, and analysis methods in system biology (Dematté and Prandi 2010; Nobile et al. 2017; Yang et al. 2017) (Table 1.9).

1.4.4.4 Cloud Computing in System Biology

Cloud computing is defined as a model enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources such as networks, servers, storage, applications, and services, which can be rapidly provisioned and released with minimal management efforts or service provider interaction. The wider applicability of cloud computing lies in the development of a software framework for big data analysis in different fields including computational biology, bioinformatics, and system biology (O'Driscoll et al. 2013) (Table 1.9).

TABLE 1.9
Recent Applications of Different Computing Platform in Computational Biology

Type of Approaches	Study	References
	Cluster Computing	
MyGrid	An open source high-level middleware to support in silico experiments in biology on a Grid	Stevens et al. (2003)
MASON	Performing multiple sequence alignment of a number of sequences	Ebedes and Datta (2004)
MPI based software using ClustalW algorithm		
MC64-NW/SW for multicore64 Needleman–Wunsch/Smith–Waterman algorithm	Parallelizing/optimizing pairwise sequence alignment algorithm	Díaz et al. (2011)
MicroClAn framework	Evaluation of clustering algorithms for microarray data analysis	Bruno and Fiori (2013)
	Grid Computing	
GridBLAST	High-throughput implementation of BLAST based on Globus	Krishnan (2005)
Grid Technology-based European EGEE project infrastructure	Development of a high-performance pipeline for analyzing cDNA sequences	Trombetti et al. (2006)
Grid computing	Building phylogenetic or evolutionary trees for a set of DNA sequences	Joshi and Vadhiyar (2009)
	GUGUP Computing	
BarraCUDA	Sequence alignment based on Burrow-Wheeler Transform (BWT)	Klus et al. (2012)
GPU-REMuSic	Multiple sequence alignment based on Clustal with regular expression constraints	Lin and Lin (2014)
Molecular dynamics (MD) Program with myPresto/psygene algorithm	NonEwald scheme for long-range electrostatic interactions in biological macromolecules	Mashimo et al. (2013)
Nvidia GPUs—Hex spherical polar Fourier protein docking algorithm	Developing a fast protein docking tools for large scale protein–protein interactions (PPIs) networks	Ritchie and Venkatraman (2010)
	MEGADOCK	
(GPU—Katchalski-Katzir algorithm with Fast Fourier transform rigid docking scheme)	Application in protein–protein docking in large-scale interactome analysis	Ohue et al. (2014)
GPU-based CASSERT algorithm	Searching for similar 3D protein structures	Mrozek et al. (2014)
GPU—simulated annealing algorithm	Search of protein structure database for structural motif	Stivala et al. (2010)

(*Continued*)

TABLE 1.9 (*Continued*)
Recent Applications of Different Computing Platform in Computational Biology

Type of Approaches	Study	References
	Cloud Computing	
HadoopMapReduce programing model–cloud computing	Analysis of next-generation sequencing (NSG) data in timely and cost effective way	Shi et al. (2017)
Cloud computing with scheduling algorithm	Analysis of NSG data with reducing workflow execution time	Senturk et al. (2018)
CloudMap: A cloud-based pipeline	Mutant genome sequence analysis	Minevich et al. (2012)
Hadoop Blast (HBlast)—a parallelized BLAST algorithm	Proposal of a flexible method for virtual partitioning of databases and input query sequences	O'Driscoll et al. (2013)
Open source cloud environment with auto-scaling resources	Execution of bioinformatics and biomedical workflows	Krieger et al. (2017)

REFERENCES

Akbar, S., Hayat, M., Iqbal, M., Jan, M. A., "iACP-GAEnsC: Evolutionary genetic algorithm based ensemble classification of anticancer peptides by utilizing hybrid feature space." *Artificial Intelligence in Medicine* 79, 2017: 62–70.

Alipanahi, B., Delong, A., Weirauch, M. T., Frey, B. J., "Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning." *Nature Biotechnology* 33, 2015: 831–838.

Al-Shayji, K. A., Liu, Y. A., "Predictive modeling of large-scale commercial water desalination plants: Data-based neural network and model-based process simulation." *Industrial & Engineering Chemistry Research* 41, no. 25, 2002: 6460–6474.

Altaf-Ul-Amin, M., Afendi, F. M., Kiboi, S. K., Kanaya, S., "Systems biology in the context of big data and networks." *BioMed Research International* 2014, 2014: Article ID 428570.

Anand, S., Buchner, A., "*Decision Support Using Data Mining*." Financial Times Pitman Publishers, London, 1998.

Angermueller, C., Lee, H., Reik, W., Stegle, O., "*Accurate prediction of single-cell DNA methylation states using deep learning*." bioRxiv (2016) doi: 10.1101/055715.

Banno, M., Komiyama, Y., Caoa, W., Okua, Y., Ueki, K., Sumikoshi, K., Nakamura, S., Teradaa, T., Shimizu, K., "Development of a sugar-binding residue prediction system from protein sequences using support vector machine." *Computational Biology and Chemistry* 66, 2017: 36–43.

Behera, S. K., Meher, S. K., Park, H-S., "Artificial neural network model for predicting methane percentage in biogas recovered from a landfill upon injection of liquid organic waste." *Clean Technology and Environment Policy* 17, 2015: 443–453.

Behzad R-G., Rafiei, F., Niknafs, A. A., Behzad, Z., "Prediction of microRNA target genes using an efficient genetic algorithm based decision tree." *FEBS Open Bio* 5, 2015: 877–884.

Bhosale, H. S., Gadekar, D. P., "A review paper on big data and Hadoop." *International Journal of Scientific and Research Publications* 4, no. 10, 2014: ISSN 2250-3153.

Bonelli, M. G., Ferrini, M., Manni, A., "Artificial neural networks to evaluate organic and inorganic contamination in agricultural soils." *Chemosphere* 186, 2017: 124–131.

Bruno, G., Fiori, A., "MicroClAn: Microarray clustering analysis." *Journal of Parallel and Distributed Computing* 73, no. 3, 2013: 360–370.

Cabena, P., Hadjinian, P., Stadler, R., Verhees, J., Zanasi, A., "*Discovering Data Mining: From Concepts to Implementation*," Prentice-Hall Saddle River, New Jersey, 1998.

Chen, Z., Lin, T., Tang, N., Xia, X., "A parallel genetic algorithm based feature selection and parameter optimization for support vector machine." *Scientific Programming*, Article ID 2739621 2016: 1–10.

Cios, K., Teresinska, A., Konieczna, S., Potocka, J., Sharma, S., "Diagnosing myocardial perfusion from SPECT bull's-eye maps—a knowledge discovery approach." *IEEE Engineering in Medicine and Biology Magazine, special issue on Medical Data Mining and Knowledge Discovery* 19(4), 2000: 17–25.

D'heygere, T., Goethals, P. L. M., Niels, D. P., "Genetic algorithm for optimisation of predictive ecosystems models based on decision trees and neural networks." *Ecological Modelling* 195, 2006: 20–29.

DeLisle, R. K., Dixon, S. L., "Induction of Decision Trees via Evolutionary Programming." *Journal of Chemical Information and Computational Science* 44, no. 3, 2004: 862–870.

Dematté, L., Prandi, D., "GPU computing for systems biology." *Brief Bioinformatics* 11, no. 3, 2010: 323–333.

D'heygere, T., Goethals, P. L. M., De Pauw, N., "Use of genetic algorithm to select input variables in decision tree models for the prediction of benthic macro invertebrates." *Ecological Modelling* 160, 2003: 291–300.

Díaz, D., Esteban, F. J., Hernández, P., Caballero, J. A., Dorado, G., Gálvez, S., "Parallelizing and optimizing a bioinformatics pairwise sequence alignment algorithm for many-core architecture." *Parallel Computing* 37, no. 4–5, 2011: 244–259.

Djebbari, A., Quackenbush, J., "Seeded Bayesian Networks: Constructing genetic networks from microarray data." *BMC Systems Biology* 2, 2008: 57.

Dolinski, K., Troyanskaya, O. G., "Implications of big data for cell biology." *Molecular Biology of Cell* 26(14), 2015: 2575–2578.

Ebedes, J., Datta, A., "Multiple sequence alignment in parallel on a workstation cluster." *Bioinformatics* 20, no. 7, 2004: 1193–1195.

Farid, D. M., Al-Mamun, M. A., Manderick, B., Nowe, A., "An adaptive rule-based classifier for mining big biological data." *Expert Systems with Applications* 64, 2016: 305–316.

Fayyad, U., Piatesky-Shapiro, G., Smyth, P., Uthurusamy, R. (Eds.), "*Advances in Knowledge Discovery and Data Mining*." AAAI Press, Cambridge, 1996.

Foster, B., Collope, F., Ungar, L. H., "*Forecasting Using Neural Networks*." *AIChE Annual Meeting*, Chicago, USA, 1990.

Geurts, P., Irrthum, A., Wehenkel, L., "Supervised learning with decision tree-based methods in computational and systems biology." *Molecular Biosystems* 12, 2009: 1593–1605.

Grahovac, J., Jokić, A., Dodić, J., Vučurović, D., Dodić, S., "Modelling and prediction of bioethanol production from intermediates and byproduct of sugar beet processing using neural networks." *Renewable Energy* 85, 2016: 953–958.

Gullo, F., "From patterns in data to knowledge discovery: What data mining can do?" *Physics Procedia* 62, 2015: 18–22.

Hassoun, M. H., "*Fundamentals of Artificial Neural Networks*." MIT Press, Cambridge, 1995.

Haykin, S., "*Neural Networks: A Comprehensive Foundation*." Macmillan College Co., New York, 1994.

Heinsfeld, A. S., Franco, A. R., Craddock, R., Buchweitz, A., Meneguzzi, F., "Identification of autism spectrum disorder using deep learning and the ABIDE dataset." *NeuroImage: Clinical* 17, 2018: 16–23.

Hornik, K., Stinchcombe, M., White, H., "Multilayer feedforward networks are universal approximators." *Neural Network* 2, 1989: 359–366.

Hoskins, J. C., Himmelblau, D. M., "Artificial neural network models of knowledge representation in chemical engineering." *Computers & Chemical Engineering* 12, 1988: 881–890.

Hung, F-H., Chiu, H-W., "Cancer subtype prediction from a pathway-level perspective by using a support vector machine based on integrated gene expression and protein network." *Computer Methods and Programs in Biomedicine* 141, 2017: 27–34.

Jacq, N., Blanchet, C., Combet, C., Cornillot, E., Duret, L., Kurata, K., Nakamura, H., Silvestre, T., Breton, V., "Grid as a bioinformatic tool." *Parallel Computing* 30, no. 9–10, 2004: 1093–1107.

Jiang, D., Ooi, B. C., Shi, L., Wu, S., "The performance of MapReduce: An in-depth study." *Proceedings of the VLDB Endowment* 3, no. 1–2, 2010: 472–483.

Joshi, Y., Vadhiyar, S., "Analysis of DNA sequence transformations on grids." *Journal of Parallel and Distributed Computing* 69, no. 1, 2009: 80–90.

Kelley, D. R., Snoek, J., Rinn, J., "Basset: Learning the regulatory code of the accessible genome with deep convolutional neural networks." *Genome Research* 26, no. 7, 2016: 990–999.

Khanbabaei, M., Sobhani, F. M., Alborzi, M., Radfar, R., "Developing an integrated framework for using data mining techniques and ontology concepts for process improvement." *Journal of Systems and Software* 137, 2018: 78–95.

Klus, P., Lam, S., Lyberg, D., et al., "BarraCUDA—a fast short read sequence aligner using graphics processing units." *BMC Research Notes* 5, 2012: 27.

Kohonen, T., "*Self-Organization and Associative Memory*." Springer-Verlag, Berlin, 1988.

Kraus, O. Z., Grys, B. T., Ba, J., Chong, Y., Frey, B. J., Boone, C., Andrews, B. J., "Automated analysis of high-content microscopy data with deep learning." *Molecular System Biology* 13, no. 4, 2017: 924.

Krieger, M. T., Torreno, O., Trelles, O., Kranzlmüller, D., "Building an open source cloud environment with auto-scaling resources for executing bioinformatics and biomedical workflows." *Future Generation Computer Systems* 67, 2017: 329–340.

Krishnan, A., "GridBLAST: A Globus-based high-throughput implementation of BLAST in a Grid computing framework." *Concurrency and Computations Practice and Experience* 17, no. 13, 2005: 1607–1623.

Lakshminarayanan, R., Kumar, B., Raju, M., "*Cloud computing benefits for educational institutions*." arXiv preprint arXiv:1305.2616 (2013).

Li, T., Dou, Y., Liu, X., Zhao, Y., Lv, Q., "Multiple kernel clustering with corrupted kernels." *Neurocomputing* 267, 2017: 447–454.

Lin, C. Y., Lin, Y. S., "Efficient parallel algorithm for multiple sequence alignments with regular expression constraints on graphics processing units." *International Journal of Computational Science and Engineering* 9, no. 1, 2014: 11–20.

López, M. E., Rene, E. R., Boger, Z., Veiga, M. C., Kennes, C., "Modelling the removal of volatile pollutants under transient conditions in a two-stage bioreactor using artificial neural networks." *Journal of Hazardous Materials* 324, 2017: 100–109.

Lowes, D. A., Galley, H. F., Moura, A. P. S., Webster, N. R., "Brief iso-flurane anaesthesia affects differential gene expression, gene ontology and gene networks in rat brain." *Behavioural Brain Research* 317, 2017: 453–460.

Mah, R. S., Chakravarthy, H. V., "Pattern recognition using artificial neural networks." *Computers & Chemical Engineering* 16, no. 4, 1992: 371–377.

Maier, H. R., Dandy, G. C., "Neural networks for the prediction and forecasting of water resources variables: A review of modelling issues and applications." *Environmental Modelling & Software* 15, 2000: 101–124.

Mashimo, T., Fukunishi, Y., Kamiya, N., Takano, Y., Fukuda, I., Nakamura, H., "Molecular dynamics simulations accelerated by GPU for biological macromolecules with a non-Ewald scheme for electrostatic interactions." *Journal of Chemical Theory and Computation* 9, no. 12, 2013: 5599–609.

Matei, Z., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X. et al., "Apache Spark: A unified engine for big data processing." *Communications of the ACM* 59.11, 2016: 56–65, 2016.

Medvedev, V., Kurasova, O., Bernatavičienė, J., Treigys, P., Marcinkevičius, V., Dzemyda, G., "A new web-based solution for modelling data mining processes." *Simulation Modelling Practice and Theory* 76, 2017: 34–46.

Milone, D. H., Stegmayer, G., Kamenetzky, L., López, M., Carrari, F., "Clustering biological data with SOMs: On topology preservation in non-linear dimensional reduction." *Expert Systems with Applications* 40, no. 9, 2013: 3841–3845.

Minevich, G., Park, S. D., Blankenberg, D., Poole, J. R., Hobert, O., "CloudMap: A cloud-based pipeline for analysis of mutant genome sequences." *Genetics* 192, no. 4, 2012: 1249–1269.

Mrozek, D., Brozek, M., Małysiak-Mrozek, B., "Parallel implementation of 3D protein structure similarity searches using a GPU and the CUDA." *Journal Molecular Modeling* 20, no. 2, 2014: 1–17.

Nattkemper, T. W., "Multivariate image analysis in biomedicine." *Journal of Biomedical Informatics* 37, no. 5, 2004: 380–391.

Nguyen, T., Nahavandi, S., Creighton, D., Khosravi, A., "Mass spectrometry cancer data classification usingwavelets and geneticalgorithm." *FEBS Letters* 589, 2015: 3879–3886.

Nishiyama, K., "Decisiontree for the binding of dipeptides to the thermally fluctuating surface of cathepsin K." *Chemical Physics Letters* 647, 2016: 42–45.

Nobile, M. S., Cazzaniga, P., Tangherloni, A., Besozzi, D., "Graphics processing units in bioinformatics, computational biology and systems biology." *Brief Bioinformatics* 18, no. 1, 2017: 870–885.

Odell, S. G., Lazo, G. R., Woodhouse, M. R., Hane, D. L., Sen, T. Z., "The art of curation at a biological database: Principles and application." *Current Plant Biology* 2017, https://doi.org/10.1016/j.cpb.2017.11.001.

O'Driscoll, A., Daugelaite, J., Sleator, R. D., "Big data, Hadoop and cloud computing in genomics." *Journal of Biomedical Informatics* 46, no. 5, 2013: 774–781.

Ohue, M., Shimoda, T., Suzuki, S., Matsuzaki, Y., Ishida, T., Akiyama, Y., "MEGADOCK 4.0: An ultra–high-performance protein–protein docking software for heterogeneous supercomputers." *Bioinformatics* 30, no. 22, 2014: 3281–3283.

Pagnuco, I. A., Pastore, J. I., Abras, G., Brun, M., Ballarin, V. L., "Analysis of genetic association using hierarchical clustering and cluster validation indices." *Genomics* 109, no. 5–6, 2017: 438–445.

Park, Y., Kellis, M., "Deep learning for regulatory genomics." *Nature Biotechnology* 33, 2015: 825–826.

Pärnamaa, T., Parts, L., "Accurate classification of protein subcellular localization from high throughput microscopy images using deep learning." *G3 (Bethesda)* 7, no. 5, 2017: 1385–1392.

Poswar, F. O., Santos, L. I., Farias, L. C., Guimarães, T. A., Santos, S. H. S., Jones, K. M., Paula, A. M. B., et al., "An adaptation of particle swarm clustering applied in basal cell carcinoma, squamous cell carcinoma of the skin and actinic keratosis." *Meta Gene* 12, 2017: 72–77.

Raji, C. G., Vinod Chandra, S. S., "Graft survival prediction in liver transplantation using artificial neural network models." *Journal of Computational Science* 16, 2016: 72–78.

Rakitsch, B., Stegle, O., "Modelling local gene networks increases power to detect trans-acting genetic effects on gene expression." *Genome Biology* 17, 2016: 33.

Ramírez, M. C. V., Velho, H. F. de C., Ferreira, N. J., "Artificial neural network technique for rainfall forecasting applied to the São Paulo region." *Journal of Hydrology* 301, no. 1–4, 2005: 146–162.

Rampasek, L., Goldenberg, A., "TensorFlow: Biology's Gateway to Deep Learning?" *Cell Systems* 2, no. 1, 2016: 12–14.

Rene, E. R., "Removal of volatile organic compounds using compost biofiters." Ph.D Thesis, Submitted to the Dept Chem. Eng., IIT Madras, 2005.

Ritchie, D. W., Venkatraman, V., "Ultra-fast FFT protein docking on graphics processors." *Bioinformatics* 26, no. 19, 2010: 2398–2405.

Rokach, L., Maimon, O., "Data Mining with Decision Trees: Theory and Applications." In: Bunke, H. and Wang, P. S. P. (Eds.), *Series: Machine Perception and Artificial Intelligence*. Vol. 69. World Scientific Press, Singapore, 2008.

Rumelhart, D. E., Hinton, G. E., Williams, R. J., "Learning Internal Representations by Error Propagation." In: Rumelhart, D. E. and McClelland, J. L. (Eds.), *Parallel Distributed Processing*. MIT Press, Cambridge, Chapter 8, 318–362, 1986.

Senturk, I. F., Balakrishnan, P., Abu-Doleh, A., Kaya, K., Malluhi, Q., Çatalyürek, Ü. V., "A resource provisioning framework for bioinformatics applications in multi-cloud environments." *Future Generation Computer Systems* 78, no. 1, 2018: 379–391.

Shao, X., Tian, Y., Wu, L., Wang, Y., Jing, L., Deng, N., "Predicting DNA- and RNA-binding proteins from sequences with kernel methods." *Journal of Theoretical Biology* 258, no. 2, 2009: 289–293.

Sharma, R., Singh, K., Singhal, D., Ghosh, R., "Neural network applications for detecting process faults in packed towers." *Chemical Engineering and Processing* 43, no. 7, 2004: 841–847.

Shi, L., Wang, Z., Yu, W., Meng, X., "A case study of tuning MapReduce for efficient Bioinformatics in the cloud." *Parallel Computing* 61, 2017: 83–95.

Sözen, A., Arcaklioglu, E., Özalp, M., Çaglar, N., "Forecasting based on neural network approach of solar potential in Turkey." *Renewable Energy* 30, no. 7, 2005: 1075–1090.

Stevens, R. D., Robinson, A. J., Goble, C. A., "myGrid: Personalised bioinformatics on the information grid." *Bioinformatics* 19, 2003: i302–i304.

Stivala, A. D., Stuckey, P. J., Wirth, A. I., "Fast and accurate protein substructure searching with simulated annealing and GPUs." *BMC Bioinformatics* 11, 2010: 446.

Tarca, A. L., Carey, V. J., Chen, X. W., Romero, R., Drăghici, S., "ML and its application in biology." *PLoS Computational Biology* 3, no. 6, 2007: e116.

Tekiner, F., Keane, J. A., "Big data framework." *Proceeding of the 2013 IEEE International Conference on Systems, Man, and Cybernetics*, (2013): 1494–1499.

Triguero, I., Río, S., López, V., Bacardit, J., Benítez, J. M., Herrera, F., "ROSEFW-RF: The winner algorithm for the ECBDL'14 big data competition: An extremely imbalanced big data bioinformatics problem." *Knowledge-Based Systems* 87, 2015: 69–79.

Trombetti, G. A., Merelli, I., Milanesi, L., "High performance cDNA sequence analysis using grid technology." *Journal of Parallel and Distributed Computing* 66, no. 12, 2006: 1482–1488.

Uslan, V., Huseyin, S., "Quantitative prediction of peptide binding affinity by using hybrid fuzzy support vector regression." *Applied Soft Computing Journal* 43, 2016: 210–221.

Vyas, R., Bapat, S., Jain, E., Karthikeyan, M., Tambe, S., Kulkarni, B. D., "Building and analysis of protein-protein interactions related to diabetes mellitus using support vector machine, biomedical text mining and network analysis." *Computational Biology and Chemistry* 65, 2016: 37–44.

Wang, H., Huang, L., Jing, R., Yang, Y., Liu, K., Li, M., Wen, Z., "Identifying oncogenes as features for clinical cancer prognosis by Bayesian nonparametric variable selection algorithm." *Chemometrics and Intelligent Laboratory Systems* 146, 2015: 464–471.

Wang, H., Jing, X., Niu, B., "A discrete bacterial algorithm for features election in classification of micro array gene expression cancer data." *Knowledge-Based Systems* 126, 2017a: 8–19.

Wang, S., Li, Z., Yu, Y., Xu, J., "Folding membrane proteins by deep transfer learning." *Cell Systems* 5, no. 3, 2017b: 202–211.

Wang, M., Zhang, W., Ding, W., Dai, D., Zhang, H., Xie, H., et al., "Parallel clustering algorithm for large-scale biological data sets." *PLOS ONE* 9, no. 4, 2014: e91315.

Wei, L., Ding, Y., Su, R., Tang, J., Zou, Q., "Prediction of human protein subcellular localization using deep learning." *Journal of Parallel and Distributed Computing*, Article in Press (2017): https://doi.org/10.1016/j.jpdc.2017.08.009

Widrow, B., Sterns, S. D., "*Adaptive Signal Processing*." Prentice-Hall, Englewood Cliffs, New Jersey, 1985.

Xiao, Y., Wu, J., Lin, Z., Zhao, X., "A deep learning-based multi-model ensemble method for cancer prediction." *Computer Methods and Programs in Biomedicine* 153, 2018: 1–9.

Yan, W., Hu, S., Yang, Y., Gao, F., Chen, T., "Bayesian migration of Gaussian process regression for rapid process modeling and optimization." *Chemical Engineering Journal* 166, no. 3, 2011: 1095–1103.

Yang, C., Huang, Q., Li, Z., Liu, K., Hu, F., "Big Data and cloud computing: Innovation opportunities and challenges." *International Journal of Digital Earth* 10, no. 1, 2017: 13–53.

Yu, J., "A Bayesian inference based two-stage support vector regression framework for soft sensor development in batch bioprocesses." *Computers & Chemical Engineering* 41, 2012: 134–144.

Yu, B., Li, S., Chen, C., Xu, J., Qiu, W., Wu, X., Chen, R., "Prediction subcellular localization of gram-negative bacterial proteins by support vector machine using wavelet denoising and chou's pseudo amino acid composition." *Chemometrics and Intelligent Laboratory Systems* 167, 2017: 102–112.

Zafar, M., Kumar, S., Kumar, S., "Optimization of napthalene biodegradation by a genetic algorithm based response surface methodology." *Brazilian Journal of Chemical Engineering* 27, no. 1, 2010: 89–99.

Zhang, Y-P., Wuyunqiqige, Zheng, W., Liu, S., Zhao, C., "gDNA-Prot: Predict DNA-binding proteins by employing support vector machine and a novel numerical characterization of protein sequence." *Journal of Theoretical Biology* 406, 2016: 8–16.

Zhang, Q., Xang, S. X., Mittal, G. S., Ti, S., "AE-automation and emerging technologies: Prediction of performance indices and optimal parameters of rough rice drying using neural networks." *Biosystems Engineering* 83, no. 3, 2002: 281–290.

Zheng, Z-Y., Guo, X-N., Zhu, K-X., Peng, W., Zhou, H-M., "Artificial neural network – Genetic algorithm to optimize wheat germ fermentation condition: Application to the production of two anti-tumor benzoquinones." *Food Chemistry* 227, 2017: 264–270.

Zhou, Y., Mao, H., Yi, Z., "Cell mitosis detection using deep neural networks." *Knowledge-Based Systems* 137, 2017: 19–28.

Zhou, J., Troyanskaya, O. G., "Predicting effects of noncoding variants with deep learning-based sequence model." *Nature Methods* 12, 2015: 931–934.

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

2 Mining and Analysis of Bioprocess Data

Prerna Grover, Aditya Shah, and Shampa Sen

CONTENTS

2.1 Introduction 29
2.2 Characteristics and Types of Bioprocess Data 30
2.3 Traditional Methods Used for Bioprocess Optimization 31
2.4 Illustration of Bioprocess Data Analysis on Small-Scale Model 31
2.5 Analysis of Bioprocess Data 34
2.5.1 Data Preprocessing 34
2.5.2 Feature Selection 35
2.5.3 Data Mining 36
2.5.3.1 Descriptive Approaches 36
2.5.3.2 Predictive Approaches 36
2.6 Softwares and Mathematical Models Used 37
2.6.1 Multiple Linear Regression 37
2.6.2 Regularized Regression 38
2.6.3 Random Forests 38
2.7 Case Studies 38
2.7.1 Artificial Neural Networks for Analyzing Wine Production 38
2.7.2 Multivariate Data Analysis for Recombinant Immunoglobulin Production 39
2.8 Conclusion 39
References 39

2.1 INTRODUCTION

It is a well-known fact that the demand and production of biopharmaceuticals has been increasing exponentially over the past few decades. Since the production of the earliest biopharmaceuticals like antibiotics began on a commercial scale, scientists and engineers have been trying to record and analyze data related to process parameters in order to optimize it. Optimization of these parameters is integral to the production process, as they affect the growth of the microorganisms employed by bioprocess industries, and the quality as well as quantity of the product formed. The most primitive form of data collection consisted of machine operators manually recording data like temperature, pH, and pressure and personally noting them down. Since then, several developments have been made and now the facilities manufacturing these products are equipped with sophisticated control systems, as well as systems for collecting and archiving the data. This

archived data provides a wide scope for data mining, which in turn helps in enhancing the efficiency, consistency, and performance of the production process. This technique has offered a huge benefit to manufacturing units in the terms of the ease of characterizing and analyzing the process outcome, thereby improving the process economy [1].

In this chapter, the challenges associated with storage and analysis of data obtained from biotechnological processes has been discussed with special emphasis on various software designed and mathematical models proposed in bioprocess data mining. Applications of bioprocess data mining have been envisaged as an emerging discipline here.

2.2 CHARACTERISTICS AND TYPES OF BIOPROCESS DATA

Bioprocess data can be classified based on the frequency of measurement of parameters associated with the process carried out to produce the desired product of interest. Such data can be either monitored *online* or *off-line*. Online monitoring is vital for rapidly changing reaction parameters, while off-line monitoring is done at a slower pace [2].

Bioprocess data recording and handling is done electronically at every step, from material inputs to system outputs, control activities, system properties, and environmental constraints. Bioprocess data are heterogeneous in terms of time scale and data types with respect to the production cycle of the desired product. With respect to time scale, online parameters are calculated continuously while off-line parameters are measured within defined periods (as is required for system control actions). In addition to these measurements, some of the process parameters, such as concentration of product and quality related aspects are measured at a single time point only. Heterogeneity of process data with respect to data types can be reflected by the continuous or discrete nature of the process parameters. For example, parameters such as biomass concentration, pH, and temperature are regarded as continuous, whereas others such as feed input valve settings or gas sparging are considered to be discrete or binary, where an ON or OFF state is to be determined [3] (Table 2.1).

TABLE 2.1
Examples of Online and Off-line Bioprocess data

Online Monitored Data	Off-line Monitored Data
• pH	• Nutrient concentration
• Dissolved oxygen	• Biomass concentration
• Pressure	• Protein assays
• Temperature	• Cell types
• Flow rate	• Rheology
• Foam level	
• Stirrer speed	
• Broth weight	
• Gas composition	

FIGURE 2.1 Traditional methods for bioprocess data analysis and optimization.

2.3 TRADITIONAL METHODS USED FOR BIOPROCESS OPTIMIZATION

Prior to the development of advanced data mining methods, four traditional methods have generally been used: mathematical modeling, single variable optimization, fractional factorial experimental designs, and response-surface methodology.

These methods have their own shortcomings like being too tedious, lacking automation, and the inability to perform advanced multivariable data analysis. Thus, with the advancements in computer science and in the field of data mining, several other methods like artificial neural networks are being experimented with and tried out for the purpose of bioprocess optimization (Figure 2.1).

2.4 ILLUSTRATION OF BIOPROCESS DATA ANALYSIS ON SMALL-SCALE MODEL

Before moving on to the actual industrial scale scenario of complex bioprocess data mining, let us first examine a small-scale example.

An aerobic fermentation of a particular species of bacteria in a simple batch mode is considered. The biomass concentration (x) in g/l for 10 different intervals of time is the input data. The data obtained is as follows (Table 2.2):

The data is analyzed to calculate the maximum specific growth rate (μ), which can then be used to calculate several other parameters. The relation between time (t) and biomass concentration (X), is given by the following equation [4]:

$$\ln X = \ln X_o + \mu t$$

TABLE 2.2
Sample Data for Illustration

Time (t)	Biomass Concentration (X)	ln X
0 s	0.21 g/l	−1.56
2 s	0.25 g/l	−1.39
4 s	0.31 g/l	−1.17
6 s	0.95 g/l	−0.05
8 s	1.71 g/l	0.54
10 s	3.24 g/l	1.17
12 s	5.54 g/l	1.71
14 s	6.12 g/l	1.81
16 s	6.24 g/l	1.83
18 s	7.05 g/l	1.95

Where X_o is the initial biomass concentration, and μ is the maximum specific growth rate.

Thus, by obtaining the equation of the line of best fit for the graph of ln X versus t, its slope may be estimated and thus, the value of μ.

Given below is a code in C++ to find out the equation of the line of best fit for any given data of time and biomass: the comment lines written in blue briefly explain the purpose of some particular lines of the code (Figure 2.2)

- Variable "n:" stores the number of data pairs;
- X[n]: an array consisting of n elements used to store the values of time intervals;
- Y[n]: an array consisting of n elements used to store the values of ln of Biomass concentrations (Figures 2.3 and 2.4).

When the code is compiled and executed, the following results are obtained in the output window (Figure 2.5).

From the output window, the equation of line of best fit is obtained:

$$Y = 0.226182x - 1.54164$$

```
//Linear Fit
#include<iostream>
#include<iomanip>
#include<cmath>
using namespace std;
int main()
{
    int i,j,k,n;
    cout<<"\nEnter the no. of data pairs to be entered:\n";        //To find the size of arrays
    cin>>n;
    double x[n],y[n],a,b;
    cout<<"\nEnter the values of Time intervals:\n";                //Input x-values
    for (i=0;i<n;i++)
        cin>>x[i];
    cout<<"\nEnter the values of Bio-mass concentration:\n";              //Input y-values
    for (i=0;i<n;i++)
        cin>>y[i];
```

FIGURE 2.2 First part of the program allotted to the input of data.

```
18      double xsum=0,x2sum=0,ysum=0,xysum=0;                    //variables for sums/sigma of xi,yi,xi^2,xiyi etc
19      for (i=0;i<n;i++)
20      {
21          xsum=xsum+x[i];                                  //calculate sigma(xi)
22          ysum=ysum+y[i];                                  //calculate sigma(yi)
23          x2sum=x2sum+pow(x[i],2);                          //calculate sigma(x^2i)
24          xysum=xysum+x[i]*y[i];                              //calculate sigma(xi*yi)
25      }
26      a=(n*xysum-xsum*ysum)/(n*x2sum-xsum*xsum);                //calculate slope
27      b=(x2sum*ysum-xsum*xysum)/(x2sum*n-xsum*xsum);              //calculate intercept
28      double y_fit[n];                                  //an array to store the new fitted values of y
29      for (i=0;i<n;i++)
30          y_fit[i]=a*x[i]+b;                            //to calculate y(fitted) at given x points
```

FIGURE 2.3 Portion of the program used to calculate the slope of the line and fitted values of the points.

```
31      cout<<"S.no"<<setw(5)<<"x"<<setw(19)<<"y(observed)"<<setw(19)<<"y(fitted)"<<endl;
32      cout<<"-------------------------------------------------------------\n";
33      for (i=0;i<n;i++)                                          //print a table of x,y(obs.) and y(fit.)
34          cout<<i+1<<"."<<setw(8)<<x[i]<<setw(15)<<y[i]<<setw(18)<<y_fit[i]<<endl;
35      cout<<"\nThe linear fit line is of the form\n\n"<<a<<"x + "<<b<<endl;        //print the best fit line
36      return 0;
37  }
```

FIGURE 2.4 Display the results.

```
Enter the no. of data pairs to be entered:
10

Enter the values of Time intervals:
0
2
4
6
8
10
12
14
16
18

Enter the values of Bio-mass concentration:
-1.56
-1.39
-1.17
0.05
0.54
1.17
1.71
1.81
1.83
1.95
S.no     x          y(observed)            y(fitted)
-------------------------------------------------------------
1.        0           -1.56            -1.54164
2.        2           -1.39            -1.08927
3.        4           -1.17           -0.636909
4.        6            0.05           -0.184545
5.        8            0.54            0.267818
6.       10            1.17            0.720182
7.       12            1.71             1.17255
8.       14            1.81             1.62491
9.       16            1.83             2.07727
10.       18            1.95             2.52964

The linear fit line is of the form:

0.226182x + -1.54164

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

FIGURE 2.5 Output window.

From the slope of the above equation, the value of the maximum specific growth rate (μ) is obtained as:

$$\mu = 0.226182$$

Thus, an important biological parameter was calculated from the basic data input using computational methods.

2.5 ANALYSIS OF BIOPROCESS DATA

The ultimate objective of bioprocess data mining is to maximize the efficiency of the process with respect to the qualitative aspects, such as product quality and product titer, as well as quantitative aspects such as product yield and efficiency of production. Analysis of the data collected from an array of process runs helps in achieving these goals.

Knowledge discovery with respect to bioprocess data helps in standardizing the process protocols and thereby helping the industries to enhance revenue generation. A typical knowledge discovery process consists of several iterative steps [5]. These steps include: *data preprocessing, feature selection*, and *data mining* (Figure 2.6).

Data preprocessing is done to convert this precursor data into feature. In the following step, the dimensions are reduced by selecting a set of data, which complies best with data mining. The data mining step involves automated or computed methods, like machine learning, to establish correlations between the parameters selected in the previous step, and for developing mathematical models to estimate the product yield.

2.5.1 Data Preprocessing

With the advent of technology in bioprocess production units, the scope of human errors and difficulties in characterization of data has been greatly reduced. This significantly reduces the need for carrying out the preprocessing techniques, such as data cleaning and denoising. However, the dynamic nature of data changing with time in culture processes, downstream processes, and fermentation poses a great deal of instigations, which need to be sought and sorted before data mining. Data preprocessing can be carried out through several approaches, two of the most common being: (1) Segmenting a temporal parameter profile into different time intervals by utilizing the triangular representation method. The evaluated first order and second order derivative of the profile for each interval deduce an increasing or decreasing trend [6,7]; and (ii) selection of temporal features with the help of wavelet decomposition [8,9]. Several other approaches including piecewise estimation of

FIGURE 2.6 3 steps of data analysis.

features with the use of discrete Fourier transform methods and symbolic aggregate approximation (SAX) can be employed besides the most common ones [10,11].

Another problem arising in the preliminary stages of data analysis is the lack of alignment of the time series. This usually happens because of the presence of a lag phase or fluctuating growth rates [12,13]. This can lead to wrong interpretations and data modeling because the system avoids the possibility that the similar states do not represent the identical time points. This problem requires a preprocessing step to be rectified involving alignment of the time series of different processes [14,15]. This is done by a dynamic time wrapping strategy, which was initially developed for speech recognition [16] (Table 2.3).

2.5.2 Feature Selection

Feature selection is done in order to reduce the dimensions. In simpler terms, this stage consists of selecting features relevant to the process outcome from the set defined in data preprocessing. There are various approaches to carry out feature selection, the popular ones being wrapper and filter approaches. In the wrapper approach, the crucial parameters are defined by making a decision tree (DT), which predicts the low or high productivity of the process [8,17–19]. On the other hand, in the filter approach, the key features are selected on the basis of hypothesis testing [9,10]. All the approaches used for feature selection help in building mathematical models to predict the outcome of the process [20–22].

In addition to employing different approaches for feature selection, the sequence of events also needs to be taken care of because of the dynamic nature of process data. To prevent any redundancy, statistical methods are introduced to assess the importance of the selected features. There are chances different parameters or features from single runs, such as oxygen uptake rate and biomass growth, are often correlated. This relation may cause redundancy in the prediction of process outcomes. In order to avoid this, feature selection techniques are often used to choose from the entire feature set the features that are independent of each other. Many methods, such as nonnegative matrix factorization (NMF) and principle component analysis (PCA)

TABLE 2.3
Aspects of Data Preprocessing

Data Cleaning	Data Integration and Transformation	Data Reduction
• Fill in missing values • Smooth noisy data • Identify and remove outliers • Resolve inconsistencies	• Integrate data from multiple sources in a coherent format • Resolve data value conflicts arising due to variations in scale, units and representations	• Reduction in size of representation but retaining important information • Data compression • Discretization

are utilized to carry out this function [23]. These selected dimensions or features are further segregated into their respective groups, which belong to certain classes.

2.5.3 Data Mining

Data mining approaches can be broadly classified into *descriptive* or *predictive* approaches. Descriptive approaches look for motifs that help in carrying out data characterization, whereas predictive approaches depend upon previously observed parameters, which help in building mathematical models to predict the outcome of the process [23].

2.5.3.1 Descriptive Approaches

Descriptive methods can be carried out by either looking for patterns of interests and their temporal profile during multiple runs or data clustering.

In the first method stated, data mining is carried out by creating and executing algorithms to identify a data pattern from a large dataset. This approach assesses the pattern among features from multiple process runs. Any pattern occuring frequently is considered a pattern of interest as it renders significance to further statistical analysis [24,25]. The correlation among different features can be identified by tracking down these patterns, which further helps in framing association rules. For example, identification of patterns from multiple runs can help in determining the specific rates for particular metabolic processes.

In the second method of clustering, different runs can be clustered or grouped on the basis of similarity in the behavioral characteristics of certain features. With the help of some cluster visualization tools, such as Spot fire [26], the distinguishing features can also be determined. The element of utter importance in clustering approaches is the selection of method, which estimates the degree of similarity between two different runs on the basis of their time-dependent features. The sensitivity of this element is taken care of by carrying out the estimation of similarity in two steps. In the first step, the similarity between corresponding time dependent features of any two parallel runs is estimated [27], which is followed by the next step where the feature similarities obtained from the first step are collaborated to establish the total similarity between the process runs. The methods employed for establishing similarities between the runs, such as cosine similarity, Euclidean distance, Pearson's correlation coefficient, and information theory-based methods (e.g., mutual information) can be utilized for comparison of the same feature in multiple runs (to identify patterns), or to compare different features in a single run (to reduce the dimensions) [28].

2.5.3.2 Predictive Approaches

Predictive methods are based on assessing segregated runs and finding a valid relationship between the features and process outcomes. These established relationships, usually called models, are used to predict the process outcome and provide an easier way to describe how the predicted outcome can affect the process features. The methods employed in predictive approaches are support vector machines (SVM), DTs, regression and artificial neural networks (ANN). These methods are only helpful when the outcome has a definite discrete value. In the process runs

where the outcome value falls in a continuous range, an outcome can be segregated to different classes [29].

Predictive approaches have found extensive application in the analysis of bioprocess data. For example, ANNs can be used to predict the outcomes of a fermentation process as a nonlinear function of experimental inputs [30–32]. DTs have also been used when the process outcomes can be segregated as a high productivity process or a low productivity process [33–35]. These methods are also used in association with optimization methods to help in maximizing the desired outcome [36]. Regression models based on partial least squares (PLS) have also been found to detect errors in the process, in addition to establishing a correlation between the inputs and outputs of the process [37–39].

Furthermore, the advancements in the application of predictive approaches for bioprocess data mining have been the introduction of the structural risk minimization (SRM) principle [40–43], which is extensively useful in determining the upper limit of the generalization error and kernel-based learning, mainly used in quantifying nonlinear and linear process data in conjunction with a sensitive model construction in which the features are sorted according to their significance [41]. These methods also form the foundation for SVM, which has found its application in analyzing certain data-rich concepts. [44]

2.6 SOFTWARES AND MATHEMATICAL MODELS USED

Software commonly used for data mining purposes can be either free open-source or proprietary data mining software. Free open-source data mining software offers free online access to data mining tools. These are mainly Carrot2, Chemicalize. org, ELKI, GATE, KNIME, Massive Online Analysis (MOA), MEPX, ML-Flex, MLPACK library, NLTK (Natural Language Toolkit), Open NN, Orange, R, scikit-learn, Torch, UIMA and Weka [45]. Proprietary data mining software levy subscriptions and monetary charges to its customers before handing any rights to access the data mining tools. The examples of such software packages are Angoss Knowledge STUDIO, Clarabridge, KXEN Modeler, LIONsolver, Megaputer Intelligence, Microsoft Analysis Services, Net Owl, Open Text Big Data Analytics, Oracle Data Mining, PSeven, Qlucore Omics Explorer, Rapid Miner, SAS Enterprise Miner, SPSS Modeler, STATISTICA Data Miner, Tanagra and Vertica [46–47].

The three most common approaches to model Bioprocess data are: multiple linear regression, regularized regression, and random forests.

2.6.1 Multiple Linear Regression

In this model, a combination of multiple predictor variables is used to model the response variable. The generalized equation for this model is:

$$\gamma = \beta_0 + \alpha_1\beta_1 + \alpha_2\beta_2 + \alpha_3\beta_3 + \cdots + \alpha_p\beta_p$$

where γ is the response variable, α_i(i = 1, 2, …, p) are the predictor variables, and β_i(i = 1, 2, …, p) are their respective coefficients. β_0 represents the intercept. This

equation can alternatively be represented in vector notation. It gives, $\gamma = H\theta$, where, H represents augmented predictor and θ represents parameter vector [48–49].

2.6.2 Regularized Regression

In this model, a penalty on the coefficient size is added to the error function. LASSO, standing for least absolute shrinkage and selection operator [50], is a widely accepted concept in which L_1 norm of coefficients is used to yield sparse solutions (models having multiple coefficients equal to zero).

From the generalized equation of multiple linear regression model, the L_1 norm can be depicted as:

$$\Phi = \|y - H\theta\|^2{}_2 + \lambda \|\theta\|_1$$

where $\|\theta\|_1$ represents L_1 norm and λ represents regularization parameter. The solutions for various different values of regularization parameters can be found by the use of numerous efficient existing algorithms [51].

2.6.3 Random Forests

Random forests rely upon DTs for modeling bioprocess data. DTs can fall under the category of either *classification tree* or *regression tree*. A common term used to represent both these types together is *Classification and Regression Tree* (CART). DTs show hierarchy in their structures, in which decisions can be made either on the basis of process outcome (in regression tree) or on the basis of class (in classification tree). The concept behind using DTs as a mathematical model is that the foresaid method compares values of the features with a threshold selected by the user. This returns a set of if-then rules arranged in a hierarchy. These hierarchical rules are then represented in the form of a tree, in which the comparison of each input feature value is equivalent to a node. Ultimately, the output value is depicted by the leaves [52–53].

2.7 CASE STUDIES

2.7.1 Artificial Neural Networks for Analyzing Wine Production

In wine production, the quality of wine is determined by its taste and smell and is an essential process outcome to be considered apart from the yield. In a study by Sophocles Vlassides and his team at the University of California, in 2000, they analyzed historical data obtained from wines produced in 1996. Three parameters were analyzed: grape maturity at harvest (2 levels), fermentation temperature (3 levels), and skin contact time prior to inoculation (3 levels). Data from 31 batches was analyzed. First, a sensory analysis was done with the help of 15 professional wine tasters who rated the wine on the basis of sweetness, sourness, and viscosity on a scale of 1 to 9. An ANN was then developed to obtain three outputs: fermentation kinetics, chemical characteristics, and sensory characteristics of the wine. The results

showed that the ANN could predict the outcome fairly accurately in accordance with the results given by the wine tasters [54].

2.7.2 Multivariate Data Analysis for Recombinant Immunoglobulin Production

In a study published in 2012 by Huong Le et al. of the University of Minnesota, data obtained from 134 process parameters was analyzed from 243 runs of a production process. The study was based on Genentech's Vacaville manufacturing facility, which produces a recombinant immunoglobulin. For the analysis, kernel-based support vector regression and partial least square regression methods were used to predict the yield. Another important parameter analyzed was lactate consumption. The results accurately predicted the output and established the relationship between lactate consumption and yield [55].

2.8 CONCLUSION

In contrast to traditional fermentation and bioprocess plants, modern plants ensure control systems are highly sophisticated to maintain the consistency of the product and the sensitivity of the process. However, fluctuations are inevitable in a process plant. Keen understanding of the factors causing these fluctuations can significantly help in enhancing the performance and controlling the outcome of the process. With the expansion of the bioprocess industry, the archived process data is also expanding, which has an enormous scope in identifying the patterns and relationships between the process features and outcomes. The objective of bioprocess data mining is mainly to find out ways for process innovation or process advancement. Since in a bioreactor the process outcome ultimately depends on the physiology of the cells, interpretations by the field experts can greatly help in relating the patterns discovered to the outcomes. This can further be used in modulating the cellular physiology according to different process parameters.

The opportunities lying in bioprocess data mining and the benefits gained are enormous. This is the consequence of advancements in the tools of data mining, and insights into knowledge discovery of bioprocess functions, which have been present since the past decade. These advanced tools holding wide applications for exploring bioprocess data are considered to be potentially highly and invariably rewarding in the future.

REFERENCES

1. Walsh, G. "Biopharmaceutical benchmarks 2006." *Nature Biotechnology* 24, no. 7, 2006: 769–776.
2. Teixeira, A. P., R. Oliveira, P. M. Alves, and M. J. T. Carrondo. "Advances in on-line monitoring and control of mammalian cell cultures: Supporting the PAT initiative." *Biotechnology Advances* 27, no. 6, 2009: 726–732.
3. Kirchman, D., H. Ducklow, and R. Mitchell. "Estimates of bacterial growth from changes in uptake rates and biomass." *Applied and Environmental Microbiology* 44, no. 6, 1982: 1296–1307.

4. Ashoori, A., B. Moshiri, A. Khaki-Sedigh, and M. R. Bakhtiari. "Optimal control of a nonlinear fed-batch fermentation process using model predictive approach." *Journal of Process Control* 19, no. 7, 2009: 1162–1173.
5. Piatetsky-Shapiro, G. *Advances in Knowledge Discovery and Data Mining*. Edited by Usama M. F., Padhraic S., and Ramasamy U. Vol. 21. Menlo Park: AAAI press, 1996.
6. Cheung, J. T.-Y. and G. Stephanopoulos. "Representation of process trends—Part II. The problem of scale and qualitative scaling." *Computers & Chemical Engineering* 14, no. 4–5, 1990: 511–539.
7. Cheung, J. T.-Y. and G. Stephanopoulos. "Representation of process trends—Part I. A formal representation framework." *Computers & Chemical Engineering* 14, no. 4–5, 1990: 495–510.
8. Bakshi, B. R. and G. Stephanopoulos. "Representation of process trends—IV. Induction of real-time patterns from operating data for diagnosis and supervisory control." *Computers & Chemical Engineering* 18, no. 4, 1994: 303–332.
9. Bakshi, B. R. and G. Stephanopoulos. "Representation of process trends—III. Multiscale extraction of trends from process data." *Computers & Chemical Engineering* 18, no. 4, 1994: 267–302.
10. Moult, J. "Rigorous performance evaluation in protein structure modelling and implications for computational biology." *Philosophical Transactions of the Royal Society of London B: Biological Sciences* 361, no. 1467, 2006: 453–458.
11. Tompa, M., N. Li, T. L. Bailey, G. M. Church, B. De Moor, E. Eskin, and A. V. Favorov et al. "Assessing computational tools for the discovery of transcription factor binding sites." *Nature Biotechnology* 23, no. 1, 2005: 137–144.
12. Huang, J., H. Nanami, A. Kanda, H. Shimizu, and S. Shioya. "Classification of fermentation performance by multivariate analysis based on mean hypothesis testing." *Journal of Bioscience and Bioengineering* 94, no. 3, 2002: 251–257.
13. Kamimura, R. T., S. Bicciato, H. Shimizu, J. Alford, and G. Stephanopoulos. "Mining of biological data I: Identifying discriminating features via mean hypothesis testing." *Metabolic Engineering* 2, no. 3, 2000: 218–227.
14. Sakoe, H. and S. Chiba. "Dynamic programming algorithm optimization for spoken word recognition." *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26, no. 1, 1978: 43–49.
15. Keogh, E., K. Chakrabarti, M. Pazzani, and S. Mehrotra. "Locally adaptive dimensionality reduction for indexing large time series databases." *ACM Sigmod Record* 30, no. 2, 2001: 151–162.
16. Keogh, E. and C. A. Ratanamahatana. "Exact indexing of dynamic time warping." *Knowledge and Information Systems* 7, no. 3, 2005: 358–386.
17. Buck, K. K. S., V. Subramanian, and D. E. Block. "Identification of critical batch operating parameters in fed-batch recombinant *E. coli* fermentations using decision tree analysis." *Biotechnology Progress* 18, no. 6, 2002: 1366–1376.
18. Coleman, M. C., K. K. S. Buck, and D. E. Block. "An integrated approach to optimization of Escherichia coli fermentations using historical data." *Biotechnology and Bioengineering* 84, no. 3, 2003: 274–285.
19. Stephanopoulos, G., G. Locher, M. J. Duff, R. Kamimura, and G. Stephanopoulos. "Fermentation database mining by pattern recognition." *Biotechnology and Bioengineering* 53, no. 5, 1997: 443–452.
20. Tai, Y. C. and T. P. Speed. "A multivariate empirical Bayes statistic for replicated microarray time course data." *The Annals of Statistics* 34, no. 5, 2006: 2387–2412.
21. Storey, J. D., W. Xiao, J. T. Leek, R. G. Tompkins, and R. W. Davis. "Significance analysis of time course microarray experiments." *Proceedings of the National Academy of Sciences of the United States of America* 102, no. 36, 2005: 12837–12842.

22. Bar-Joseph, Z., G. Gerber, I. Simon, D. K. Gifford, and T. S. Jaakkola. "Comparing the continuous representation of time-series expression profiles to identify differentially expressed genes." *Proceedings of the National Academy of Sciences* 100, no. 18, 2003: 10146–10151.
23. Mercier, S. M., B. Diepenbroek, R. H. Wijffels, and M. Streefland. "Multivariate PAT solutions for biopharmaceutical cultivation: Current progress and limitations." *Trends in Biotechnology* 32, no. 6, 2014: 329–336.
24. Kamimura, R. T., S. Bicciato, H. Shimizu, J. Alford, and G. Stephanopoulos. "Mining of biological data II: Assessing data structure and class homogeneity by cluster analysis." *Metabolic Engineering* 2, no. 3, 2000: 228–238.
25. Agrawal, R. and R. Srikant. "Fast algorithms for mining association rules." In *Proc. 20th int. conf. very large data bases, VLDB*, Vol. 1215, pp. 487–499. 1994.
26. Seno, M. and G. Karypis. "Lpminer: An algorithm for finding frequent itemsets using length-decreasing support constraint." In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pp. 505–512. IEEE, 2001. Ahlberg, C., 1996 Spotfire: An information exploration environment. SIGMOD Rec. 25, 25–29.
27. D'haeseleer, P. "How does gene expression clustering work?" *Nature Biotechnology* 23, no. 12, 2005: 1499–1501.
28. Lapointe, J., C. Li, J. P. Higgins, M. Van De Rijn, E. Bair, K. Montgomery, and M. Ferrari et al. "Gene expression profiling identifies clinically relevant subtypes of prostate cancer." *Proceedings of the National Academy of Sciences of the United States of America* 101, no. 3, 2004: 811–816.
29. Papa, J. P., A. X. Falcao, and C. T. N. Suzuki. "Supervised pattern classification based on optimum-path forest." *International Journal of Imaging Systems and Technology* 19, no. 2, 2009: 120–131.
30. Read, E. K., R. B. Shah, B. S. Riley, J. T. Park, K. A. Brorson, and A. S. Rathore. "Process analytical technology (PAT) for biopharmaceutical products: Part II. Concepts and applications." *Biotechnology and Bioengineering* 105, no. 2, 2010: 285–295.
31. Glassey, J., G. A. Montague, A. C. Ward, and B. V. Kara. "Enhanced supervision of recombinant E. coli fermentation via artificial neural networks." *Process Biochemistry* 29, no. 5, 1994: 387–398.
32. Glassey, J., G. A. Montague, A. C. Ward, and B. V. Kara. "Artificial neural network based experimental design procedure for enhancing fermentation development." *Biotechnology and Bioengineering* 44, no. 4, 1994: 397–405.
33. Bachinger, T., U. Riese, R. K. Eriksson, and C.-F. Mandenius. "Electronic nose for estimation of product concentration in mammalian cell cultivation." *Bioprocess and Biosystems Engineering* 23, no. 6, 2000: 637–642.
34. Vlassides, S., J. G. Ferrier, and D. E. Block. "Using historical data for bioprocess optimization: Modeling wine characteristics using artificial neural networks and archived process information." *Biotechnology and Bioengineering* 73, no. 1, 2001: 55–68.
35. Coleman, M. C. and D. E. Block. "Retrospective optimization of time-dependent fermentation control strategies using time-independent historical data." *Biotechnology and Bioengineering* 95, no. 3, 2006: 412–423.
36. Kirdar, A. O., J. S. Conner, J. Baclaski, and A. S. Rathore. "Application of multivariate analysis toward biotech processes: Case study of a cell-culture unit operation." *Biotechnology Progress* 23, no. 1, 2007: 61–67.
37. Vapnik, V. N. and V. Vapnik. *Statistical learning theory.* Vol. 1. New York: Wiley, 1998.
38. Vapnik, V. *The nature of statistical learning theory.* New York: Springer science & business media, 2013.
39. Yang, L., R. Jin, and J. Ye. "Online learning by ellipsoid method." In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1153–1160. ACM, 2009.

40. Weinberger, K. Q. and L. K. Saul. "Distance metric learning for large margin nearest neighbor classification." *Journal of Machine Learning Research* 10, 2009: 207–244.
41. Lanckriet, G. R. G., T. De Bie, N. Cristianini, M. I. Jordan, and W. S. Noble. "A statistical framework for genomic data fusion." *Bioinformatics* 20, no. 16, 2004: 2626–2635.
42. Lanckriet, G. R. G., N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. "Learning the kernel matrix with semidefinite programming." *Journal of Machine learning Research* 5, 2004: 27–72.
43. Brown, M. P. S., W. N. Grundy, D. Lin, N. Cristianini, C. W. Sugnet, T. S. Furey, M. Ares, and D. Haussler. "Knowledge-based analysis of microarray gene expression data by using support vector machines." *Proceedings of the National Academy of Sciences* 97, no. 1, 2000: 262–267.
44. Charaniya, S., S. Mehra, W. Lian, K. P. Jayapal, G. Karypis, and W.-S. Hu. "Transcriptome dynamics-based operon prediction and verification in Streptomyces coelicolor." *Nucleic Acids Research* 35, no. 21, 2007: 7222–7236.
45. Tong, S. and D. Koller. "Support vector machine active learning with applications to text classification." *Journal of Machine Learning Research* 2, 2001: 45–66.
46. Tong, S. and E. Chang. "Support vector machine active learning for image retrieval." In *Proceedings of the ninth ACM international conference on Multimedia*, pp. 107–118. ACM, 2001.
47. Tibshirani, R. "Regression shrinkage and selection via the lasso." *Journal of the Royal Statistical Society. Series B (Methodological)* 58, 1996: 267–288.
48. Andersen, P. K. and L. T. Skovgaard. "Multiple regression, the linear predictor." In *Regression with Linear Predictors*, pp. 231–302. Springer, New York, 2010.
49. Miller, A. *Subset selection in regression*. CRC Press, 2002.
50. Mercier, S. M., B. Diepenbroek, M. C. F. Dalm, R. H. Wijffels, and M. Streefland. "Multivariate data analysis as a PAT tool for early bioprocess development data." *Journal of Biotechnology* 167, no. 3, 2013: 262–270.
51. Goldrick, S., R. Turner, M. Kuiper, K. Lee, R. Pradhan, and S. Farid. "*Application of multivariate data analysis in the monitoring and control of mammalian cell processes*," Paper Presented at: Cell Culture Engineering XV, Palm Springs, CA, USA, May 8–13, 2016.
52. Mercier, S. M., B. Diepenbroek, R. H. Wijffels, and M. Streefland. "Multivariate PAT solutions for biopharmaceutical cultivation: Current progress and limitations." *Trends in Biotechnology* 32, no. 6, 2014: 329–336.
53. Thibault, J., V. Van Breusegem, and A. Chéruy. "On-line prediction of fermentation variables using neural networks." *Biotechnology and Bioengineering* 36, no. 10, 1990: 1041–1048.
54. Syu, M.-J. and G. T. Tsao. "Neural network modeling of batch cell growth pattern." *Biotechnology and Bioengineering* 42, no. 3, 1993: 376–380.
55. Yang, X.-M. "Optimization of a cultivation process for recombinant protein production by Escherichia coli." *Journal of Biotechnology* 23, no. 3, 1992: 271–289.

3 Data Mining in Nutrigenomics

Avipsha Sarkar, Shreyasi Kundu, Shakti Singh, and Shampa Sen

CONTENTS

3.1 Introduction 43
3.2 Nutrigenomics 45
3.2.1 Natural Food Sources 45
3.2.2 Nutraceuticals 46
3.2.3 In Health Care 47
3.3 Data Mining 48
3.3.1 Nutrigenomics Database versus Data Mining 48
3.3.2 Data Mining Approaches 48
3.4 Future Prospects 50
References 50

3.1 INTRODUCTION

The concept of "personalized medicine" has evolved based on the predicted pharmacogenetic responses of individuals. These response predictions were greatly enhanced by the Human Genome Project and the resultant detection of single nucleotide polymorphisms (SNPs) within human societies. Nutritional genomics provide genetic uprising, which incorporates (Subbiah 2007):

1. *Nutrigenomics*: To understand the changes occurring in the proteome and metabolome of an individual after the dietary compounds interact with the genome.
2. *Nutrigenetics*: Involves an individual's gene-based reactions to various dietary compounds—thus, it helps in the development of a variety of nutraceuticals, well suited to the genetic makeup of the person.

Nutrigenomics explores the effects of nutrients on the genome, proteome and metabolome, and nutrigenetics. Figure 3.1 below depicts a scheme to estimate the risk of a particular disease and the health condition of an individual by utilizing several nutrigenomic techniques incorporating dietary, genetic, and metabolic knowledge. On the basis of more than one interrelated dimension, a "systems" biomarker has been designed by the amalgamation of various aforesaid data.

FIGURE 3.1 Nutrigenomic biomarker of health.

Nutraceuticals are the products derived from food sources having extra health benefits in addition to their basic nutritional value. They can be categorized as: dietary supplements, functional foods, medicinal foods, and pharmaceuticals. Nutrigenomics has a wide range of applications and can be used to understand the consequence of nutritional compounds on genetic constitution, function, and expression profile along with the transcriptome of an individual. Intake of nutraceutical substitutes in the form of tablets or pill capsules are also known to prevent multifactorial diseases like cancer, CVD, type II diabetes mellitus, and some monogenic disorders, for example, phenylketonuria, galactosemia, lactose intolerance, etc. Nutrigenetics and nutrigenomics are two different fields, which have two completely different approaches for understanding the genetic interaction with the dietary compounds. However, their common final aspiration is to personalize diet, understand genetic polymorphisms, and to offer potent gateways and hence augment human health (Mutch et al. 2005). Nutritional interventions like, antioxidants, vitamins (e.g., vitamin A, E, D and C, etc.), flavonoids, omega-3 fatty acids, etc., aim to prevent the pathogenesis of diabetes mellitus, metabolic syndromes, and their complications. Plant-derived food products show also positive effects on the reduction of chronic diseases due to the presence of phytochemicals. Various food materials like green tea, vitamin E, soy, vitamin D, lycopene, and selenium are taken from natural sources and are being used to alleviate human health (Cencic and Chingwaru 2010). So, inborn error metabolism can be rectified artificially by giving nutraceuticals, or personalized diet supports, which have future aspects of personalized nutrigenome medications. Data mining is the procedure of finding patterns in data to calculate a result, or to predict future outputs. Using data mining tools and techniques, huge amounts of data can be handled in a short period of time. Cluster detection, memory-based reasoning, market basket analysis, genetic algorithms, link analysis, decision trees, and neural nets are some of the powerful techniques used for data mining and evaluation purposes. Data mining is playing a major role in nutrigenomics analysis, as it helps to study the present status

of nutraceuticals in the market, as well as control the market value of those medicated products and the customers' responses to them.

3.2 NUTRIGENOMICS

Nutrigenomics deal with the study of the effects of nutrients on the expression of an individual's genetic makeup. This, in turn, provides information for the biochemical functions of the body. Better nutritional advice can be provided to research groups and subgroups, or the general population by utilizing nutrigenomics (Fenech 2015). Revolutionary changes in genetic research have occurred due to the novel techniques used to study and analyze the complete genome. This has also helped in understanding the exact mechanism of diseases. Genomic research has changed into statistical evaluation of large masses of data, otherwise known as big data (Palotie and Ripatti 2017). A personalized diet or nutrition can be important for its role in a healthy lifestyle, as well as in chronic and acute diseases. Due to the polymorphism in the genetic blueprint of our body, food-processing mechanisms are different for each individual. So, nutrigenomics is a powerful area of genomic study by which a relation between dietary intake and its effect can be analyzed, making disease prevention possible. "Omics" define different disciplines like transcriptomics, proteomics, and metabolomics, studying the way these genes–proteins–metabolites interact, which is useful for personalized nutrition—nutrigenomics (Affolter et al. 2009). Analytical softwares like genomics analysis tools (CGA™ Tools) can be used to analyze human genome.

The recent interest of nutrigenomics study can be used to evaluate the genetic variations of different individuals in the laboratory. By analyzing sophisticated analytical technologies of metadata we can analyze the systemic chemical process, and can identify and quantify cellular metabolites. For example, some nutrigenomic research tools, available in the field of nutrigenomics, are the synthesis of mRNA (transcriptomics), polymerase chain reaction (PCR), geneticp (SNPs), and protein synthesis (proteomics); additionally, microarray data in the field of nutritional genomics (nutrigenomics) are available. The nutrigenomics provide an approach to integrating the various available technologies for systems biology. *OMICS* provide reports on methods and research, including bioinformatics, computational biology, metadata, data standards, data sharing, databases, biomedical informatics, biobanks, and the methods along with the statistical and algorithmic developments of cloud computing (Dove et al. 2012). Understanding the function of metabolic stress and its link to various metabolic syndromes is a key objective nutrigenomics aspires to. It can be used for preventing disease by assembling and studying metabolic stress, diabetes, and phenotypes combining inflammation and insulin stress (Afman and Müller 2006). Improvements can be made in the field of nutrition exploration with novel tools like genome editing and stem cell-based techniques (Mathers 2017).

3.2.1 Natural Food Sources

In daily life we consume different food and food products form a variety of sources that has potential nutritional value. The following substances provide nutrition

TABLE 3.1
Nutraceuticals Contain Natural Food Source

Components	Sources	Health Benifits
Carotinoids • Beta–carotene • Lycopene	Carrots, various fruits Tomatoes and tomato products	Neutralize free radicals, boost the cellular antioxidant defense
Fatty acids	Tree nuts	Reduce coronary heart disease
Phenols Caffeic acid, ferullic acid	Apples, pears, citrus fruits etc.	Enhance detoxification property and boost cellular antioxidant defense
Flavonoids	Onions, apples, tea, broccoli	Neutralize free radicals, boost cellular antioxidant defense
Phytoestrogens Isoflavones (daizein, genistein)	Soybeans and soy-based foods	Maintain bone health, healthy brain and immune responses
Probiotics Biofidobacteria, Lactobacilli	Yogurt, dairy sources	Improve gastrointestinal health and immunity
Dietary fiber Insoluble fiber	Wheat bran	Maintenance of healthy digestive tract

to our body: carbohydrates, proteins, fats, vitamins, minerals, fiber, water, etc. A healthy human being contains roughly 1% carbohydrates, 16% proteins, 16% fats, 6% minerals, 62% water, and small amounts of vitamins. Food contains different nutritious components and after consuming it complex molecules are broken down as smaller absorbable molecules, with the help of different kinds of enzymes and juices, to restore energy to our body. This energy and nutrition production mechanism is different in all individuals due to the presence of different levels of polymorphism. Dietary gene interactions are highly complex and their regulations are difficult to understand. These interactions are controlled by environmental factors, the individual's genotype, and their diet (Mead 2007). Table 3.1 represents varieties of nutraceuticals, currently available in markets (Taylor 2004), which contain natural food components of benefit to human health.

3.2.2 Nutraceuticals

Nutraceuticals is the combination of nutrients and pharmaceuticals. Nutraceuticals provide health benefits and work as a modern medicine, which help in maintaining health and act against various diseases. Nutraceuticals contain various bioactive compounds like carotenoids, phytosterols, phytoestrogens, phenols, saponins, and cyclic phenolics, which are proven to be essential in disease prevention (Pradeep and Mallikarjuna 2012). Varieties of nutraceuticals available in markets are:

- *Dietary supplements*: Contain nutrients derived from food products available in liquid and capsule form.

- *Functional foods*: Contain natural dietary supplements, or sometimes dietary supplements, which are added to obtain a functional food; for example, vitamin D added to milk.
- *Medical food*: This is specially designed for the prevention of specific diseases, or some medicinal plants able to cure certain diseases.
- *Pharmaceuticals*: This refers to the medical applications of genetically modified animals and crops.

Nutrients, dietary supplements, and herbals are major contents of nutraceuticals (Kumari et al. 2015). After understanding an individual's nutritional needs by analyzing the data according to his/her genetic materials, nutritious components can be given artificially in the form of tablets or pills, such as vitamin, calcium tablets, etc. Potential databases are now available for human gene sequencing and play a significant role in nutrigenomics. So, by detecting an individual's nutrigenomic factors, medical practitioners can prescribe nutraceuticals to prevent diseases arising due to deficiencies in food nutritional value. Personalized nutrition will be the future—designing and prescribing a diet for an individual based on their genome and their genetic variations (Pavlidis et al. 2015). Nutrigenomic data can also predict digestive enzyme production rates with the help of transcriptomics data analysis of mass protein production, as well as the proper protein folding analysis based on an individual's genomic information.

3.2.3 In Health Care

Gene function is known to play a major role in our dynamic health states and DNA has an observable trait. On a molecular level nutrients transmit signals, which can be translated into changes in gene, protein, and metabolite expression. The phenotypic appearance of monogenic disorders, like lactose intolerance or phenylketonuria, is highly influenced by an individual's diet. Therefore nutrigenomics can enhance prevention since it helps in identifying haplotype arrangements of particular mutations altering an affected person's response to diet. In case of multifactorial disorders like obesity, cancer, or cardiovascular disorders (CVD), nutrigenomics can be utilized to alter the onset—as well as development—of the disease since reports suggest they are prone to dietary intrusion (Gorduza et al. 2008). Nutrigenomics could be applied to everyday life, as the future of nutrition science offers new tools for dieticians to design and prescribe diets for each person, based on their genome and their genetic variations. The two most known genes *FTO* rs9939609 and the *MC4R* rs17782313 polymorphisms were found to be associated with type 2 diabetes which can be altered by diet. These were studied while investigating the effects of the Mediterranean diet on two polymorphisms (rs9939609 and rs17782313, respectively), where no association to diabetes type 2 was found (Ortega-Azorín et al. 2012). Genetic variation may affect appetite and calorific intake, as well as insulin signaling, inflammation, adipogenesis (formation of fat cells), and lipid metabolism. This means individual variations seen in body weight and composition are likely influenced by genetic makeup, as well as diet and activity patterns. Dietary factors can work to stabilize the genome when genetic abnormalities have occurred (Mead 2007).

3.3 DATA MINING

3.3.1 Nutrigenomics Database versus Data Mining

Mısırlı et al. in 2016 have stated that the development of a new and conventional biological system (after combining the simpler devices and parts) has become one of the most important endeavors in synthetic biology. But the procedure is not very successful since there is a shortage of completely categorized and distinct devices. The information should be made available in a compact format through novel computational advancements, making data mining more accessible. Scientists have introduced SyBiOnt, which helps in gathering information on various biological elements and their relationships with each other. SyBiOntKB is developed using SyBiOnt and also offer the inclusion of already existing biological standards and ontologies (Mısırlı et al. 2016).

The field of bioinformatics has been actively trying to overcome the key challenge of biological data amalgamation. The conventional methods for data incorporation are:

- *Data warehousing*: Where data from several databases are combined together into a single database (Goble and Stevens 2008, Balakrishnan et al. 2012, Contrino et al. 2011).
- *Federated data integration*: Where the user can make parallel queries in multiple databases, but the results will appear in an integrated format (Belleau et al. 2008, Cheung et al. 2007, Lenzerini 2002, Stein 2003, Antezana et al. 2009).

Traditional statistical techniques cannot completely analyze "big data" sets stepwise and investigate their various patterns and relationships. However, this can be done by data mining, which is also an integral tool in nutritional epidemiology. Present day health care produces huge amounts of computerized data associated with patient care, keeping all the records for various regulatory needs (Raghupathi 2016). A new handy and convenient tool has been developed offering new ways to evaluate the patients' food choices and nutritional behavior (Ngo et al. 2009). Haslam and James (2005) performed data mining in the field of nutrition among 1140 children with respect to their obesity status. They investigated the dietary patterns possibly related to obesity among them. The methodical examination of the relationship between the metabolites 3D structures and their biological activities was performed using the "KNApSAcK Metabolite Activity" database (Afendi et al. 2013, Ohtana et al. 2014).

3.3.2 Data Mining Approaches

Nutrigenomics research is gaining momentum day by day, thereby augmenting the amount of microarray data associated to this field. This calls for an immediate and well-organized infrastructure for data storage to enhance the workflow. Saito et al. (2005) developed a web-based database assimilating all the microarray publications along with expression data related to nutrigenomics. The data present in this database

are linked to other databases like PubMed and other microarray databases (Saito et al. 2005).

Scientists developed a database, known as nutritional phenotype database (dbNP), to connect nutrition studies with all the information required for the study including the diet taken, genetic variation and other phenotypic information. The databases aid in describing the two axes of nutrition, namely the effects of food ingestion and exposure. The most important constituents of these axes are metabolomics, genetics, proteomics, food composition, functional assays, and intake of food, which should be adapted during research in the field. The complete removal of restrictions, with respect to information retrieval in dbNP, requires the inclusion of good quality data, disseminated networking, setting up of rigorous protocols, and annotations definite to nutrition (Ommen et al. 2010).

Scientists have developed a database known as the Soybean Genomics and Microarray Database, which aids in obtaining information about soybean genomics (Alkharouf and Matthews 2004). This database is available as an open source system and consists of all the gene expression reports, and the relationship of soybean with their cyst nematode.

The database developed by Lee et al. in 2011 comprises information about interactions between genes and the environment. It is known as GxE and aids in mining all the data related to particular interactions significant to cardiovascular disorders, nutrition, type 2 diabetes, and blood lipids (Lee et al. 2011). GxE shows the importance of nutrition-associated diseases in the field of medicine.

Woolf et al. (2011) developed vProtein, a database for the identification of essential amino acid supplements derived from plant-based nutrition. Determination of the quantity of each food necessary is the main aim of this database (Woolf et al. 2011). The database is based on the information already presented by the U.S. Department of Agriculture about various foods.

Shen et al. in 2005 have developed a specified plant microarray database for visualizing and analyzing the data statistically. It is known as BarleyBase, and is significant for both nutrigenomics as well as plant genomics.

De Santis et al. in 2015 aimed at characterizing the nutrigenomics profile of *Salmo salar*, that is, the scientists introduced nutritional stress using soybean meal derived from plant:

1. *GSEA (gene-set enrichment analysis)*: The software R function *gage* was used to analyze the sequences uniquely annotated. This software was taken from the package GAGE, that is, the generally applicable gene-set enrichment (Luo et al. 2009). The sole purpose of the analysis was the identification of the changes in mechanism, as were already suggested by the changes in coordinated expression in the sample gene sets.
2. *Hierarchical clustering*: The resemblance linking the samples were evaluated and analyzed by the R package *pvclust* (Suzuki and Shimodaira 2006) after the entire data was analyzed by hierarchical clustering.
3. *Differential expression summary*: R package *ggplot2* was used to plot all the genes having differential expression (Wickham 2016).

3.4 FUTURE PROSPECTS

One of the new fields of "omics" science research includes nutrigenomics. Present day medicine is highly dependent on nutrigenomics. In the future, the concept of nutrigenomics could be quite essential in controlling diseases related to nutrition (Wiwanitkit 2012), as data mining in nutrigenomics has a huge potential in the field of medicine. The scientific community has been focusing on the correlation of data mining and nutrigenomics to enhance the potential of this field. The utilization of genetic information for personalized dietary guidance can be applied in the field of medicine; however, it might give rise to diverse ethical and social concerns. The revelation of an individual's genetic constituents might not be desirable. Mathematical models can be designed by developing novel tools (statistical, bioinformatics), which can assimilate enormous data sets. These data sets should consider both diet and genetic outcomes in big or/and different population. Predictive tools like "decision trees" can be used for the prediction of target diseases affecting a person due to his/her diet. This can then be used by medical professionals, or nutritionists, to prevent a particular disease by altering a patient's diet. New tools should be developed to make these predictions more accurate.

REFERENCES

Afendi, F. M., N. Ono, Y. Nakamura, K. Nakamura, L. K. Darusman, N. Kibinge, and A. H. Morita et al. "Data mining methods for omics and knowledge of crude medicinal plants toward big data biology." *Computational and Structural Biotechnology Journal* 4, no. 5, 2013: 1–14.

Affolter, M., F. Raymond, and M. Kussmann. "Omics in nutrition and health research." *Nutrigenomics and Proteomics in Health and Disease: Food Factors and Gene Interactions* 2009: 11–29.

Afman, L. and M. Müller. "Nutrigenomics: From molecular nutrition to prevention of disease." *Journal of the American Dietetic Association* 106, no. 4, 2006: 569–576.

Alkharouf, N. W. and B. F. Matthews. "SGMD: The soybean genomics and microarray database." *Nucleic Acids Research* 32, no. suppl_1, 2004: D398–D400.

Antezana, E., M. Kuiper, and V. Mironov. "Biological knowledge management: The emerging role of the semantic web technologies." *Briefings in Bioinformatics* 10, no. 4, 2009: 392–407.

Balakrishnan, R., J. Park, K. Karra, B. C. Hitz, G. Binkley, E. L. Hong, J. Sullivan, G. Micklem, and J. Michael Cherry. "YeastMine—an integrated data warehouse for Saccharomyces cerevisiae data as a multipurpose tool-kit." *Database* 2012, 2012.

Belleau, F., M.-A. Nolin, N. Tourigny, P. Rigault, and J. Morissette. "Bio2RDF: Towards a mashup to build bioinformatics knowledge systems." *Journal of Biomedical Informatics* 41, no. 5, 2008: 706–716.

Cencic, A. and W. Chingwaru. "The role of functional foods, nutraceuticals, and food supplements in intestinal health." *Nutrients* 2, no. 6, 2010: 611–625.

Cheung, K.-H., A. K. Smith, K. Y. L. Yip, C. J. O. Baker, and M. B. Gerstein. "Semantic Web approach to database integration in the life sciences." *Semantic Web* 2007: 11–30.

Contrino, S., R. N. Smith, D. Butano, A. Carr, F. Hu, R. Lyne, and K. Rutherford et al. "modMine: Flexible access to modENCODE data." *Nucleic Acids Research* 40, no. D1, 2011: D1082–D1088.

De Santis, C., K. L. Bartie, R. E. Olsen, J. B. Taggart, and D. R. Tocher. "Nutrigenomic profiling of transcriptional processes affected in liver and distal intestine in response to a soybean meal-induced nutritional stress in Atlantic salmon (Salmo salar)." *Comparative Biochemistry and Physiology Part D: Genomics and Proteomics* 15, 2015: 1–11.

Dove, E. S., V. Özdemir, and Y. Joly. "Harnessing omics sciences, population databases, and open innovation models for theranostics—guided drug discovery and development." *Drug Development Research* 73, no. 7, 2012: 439–446.

Fenech, M. "Nutrigenomics and nutrigenetics: The new paradigm for optimising health and preventing disease." *Journal of Nutritional Science and Vitaminology* 61, no. Supplement, 2015: S209–S209.

Goble, C. and R. Stevens. "State of the nation in data integration for bioinformatics." *Journal of Biomedical Informatics* 41, no. 5, 2008: 687–693.

Gorduza, E. V., L. L. Indrei, and V. M. Gorduza. "Nutrigenomics in postgenomic era." *Revista medico-chirurgicala a Societatii de Medici si Naturalisti din Iasi* 112, no. 1, 2008: 152–164.

Haslam, D. W. and W. P. James. "Obesity." *Lancet* 366, 2005: 1197–1209. CrossRef| PubMed| Web of Science® Times Cited 834.

Kumari, M., J. Shashi, and S. Jagdeep. "Nutraceutical–medicine of future." *Journal of Global Biosciences* 4, no. 7, 2015: 2790–2794.

Lee, Y.-C., C.-Q. Lai, J. M. Ordovas, and L. D. Parnell. "A database of gene-environment interactions pertaining to blood lipid traits, cardiovascular disease and type 2 diabetes." *Journal of Data Mining In Genomics & Proteomics* 2, no. 1, 2011: 106.

Lenzerini, M. "Data integration: A theoretical perspective." In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 233–246. ACM, 2002.

Luo, W., M. S. Friedman, K. Shedden, K. D. Hankenson, and P. J. Woolf. "GAGE: Generally applicable gene set enrichment for pathway analysis." *BMC Bioinformatics* 10, no. 1, 2009: 161.

Mathers, J. C. Nutrigenomics in the modern era. *Proceedings of the Nutrition Society* 76, no. 3, 2017: 265–275.

Mead, M. N. "Nutrigenomics: The genome–food interface." *Environmental Health Perspectives* 115, no. 12, 2007: A582.

Mısırlı, G., J. Hallinan, M. Pocock, P. Lord, J. A. McLaughlin, H. Sauro, and A. Wipat. "Data integration and mining for synthetic biology design." *ACS Synthetic Biology* 5, no. 10, 2016: 1086–1097.

Mutch, D. M., W. Wahli, and G. Williamson. "Nutrigenomics and nutrigenetics: The emerging faces of nutrition." *The FASEB Journal* 19, no. 12, 2005: 1602–1616.

Ngo, J., A. Engelen, M. Molag, J. Roesle, P. García-Segovia, and L. Serra-Majem. "A review of the use of information and communication technologies for dietary assessment." *British Journal of Nutrition* 101, no. S2, 2009: S102–S112.

Ohtana, Y., A. A. Abdullah, Md. Altaf-Ul-Amin, M. Huang, N. Ono, T. Sato, and T. Sugiura et al. "Clustering of 3D-structure similarity based network of secondary metabolites reveals their relationships with biological activities." *Molecular Informatics* 33, no. 11–12, 2014: 790–801.

Ommen, B., J. Bouwman, L. O. Dragsted, C. A. Drevon, R. Elliott, P. Groot, and J. Kaput et al. "Challenges of molecular nutrition research 6: The nutritional phenotype database to store, share and evaluate nutritional systems biology studies." *Genes & Nutrition* 5, no. 3, 2010: 189.

Ortega-Azorín, C., D. Corella, E. Ros, E. Gómez-Gracia, E. M. Asensio, F. Arós, and G. Sáez-Tormo et al. "Associations of the FTO rs9939609 and the MC4R rs17782313 polymorphisms with type 2 diabetes are modulated by diet, being higher when adherence to the Mediterranean diet pattern is low." *Cardiovascular Diabetology* 11, no. 1, 2012: 137.

Palotie A. and Ripatti S. "Finland establishing the internet of genomics and health data." *Duodecim* 133, no. 8, 2017: 771–5.

Pavlidis, C., G. P. Patrinos, and T. Katsila. "Nutrigenomics: A controversy." *Applied & Translational Genomics* 4, 2015: 50–53.

Pradeep K. S. and R. Mallikarjuna. Phytochemicals in vegetables and their health benefits. *Asian J. Agric. Rural Dev* 2, 2012: 177–183.

Raghupathi, W. "Data mining in healthcare." *Healthcare Informatics: Improving Efficiency through Technology, Analytics, and Management* 2016: 353–372.

Saito, K., S. Arai, and H. Kato. "A nutrigenomics database–integrated repository for publications and associated microarray data in nutrigenomics research." *British Journal of Nutrition* 94, no. 4, 2005: 493–495.

Shen, L., J. Gong, R. A. Caldo, D. Nettleton, D. Cook, R. P. Wise, and J. A. Dickerson. "BarleyBase—an expression profiling database for plant genomics." *Nucleic Acids Research* 33, no. suppl_1, 2005: D614–D618.

Stein, L. D. "Integrating biological databases." *Nature Reviews Genetics* 4, no. 5, 2003: 337–345.

Subbiah, MTR. "Nutrigenetics and nutraceuticals: The next wave riding on personalized medicine." *Translational Research* 149, no. 2, 2007: 55–61.

Suzuki, R. and H. Shimodaira. "pvclust: Hierarchical clustering with P-values via multiscale bootstrap resampling." *Bioinformatics* 22, no. 12, 2006: 1540–2.

Taylor, C. L. "Regulatory frameworks for functional foods and dietary supplements." *Nutrition Reviews* 62, no. 2, 2004: 55–59.

Wickham, H. *ggplot2: elegant graphics for data analysis*. Springer, 2016.

Wiwanitkit, V. "Database and tools for nutrigenomics: A brief summary." *Journal of Medical Nutrition and Nutraceuticals* 1, no. 2, 2012: 87.

Woolf, P. J., L. L. Fu, and A. Basu. "vProtein: Identifying optimal amino acid complements from plant-based foods." *PLOS One* 6, no. 4, 2011: e18836.

4 Machine Learning in Metabolic Engineering

Sayak Mitra

CONTENTS

4.1 Metabolic Engineering: An Introduction 53
4.1.1 Why Engineer Metabolism? 54
4.1.2 Mathematical Models in Metabolic Rengineering 54
4.1.3 Metabolic Flux Analysis 57
4.1.4 Metabolic Control Analysis 58
4.2 Softwares Used in Metabolic Engineering 59
4.2.1 MatLab 59
4.2.2 FiatFlux 60
4.2.3 OptFlux 60
4.2.4 Other Softwares 60
4.3 Machine Learning Techniques 61
4.3.1 Decision Tree Learning 63
4.3.2 Association Rule Learning 64
4.3.3 Clustering 64
4.3.4 Bayesian Networks 66
4.3.5 Genetic Algorithms (GA) 66
4.4 Future Aspects 67
References 67

4.1 METABOLIC ENGINEERING: AN INTRODUCTION

With an ever-increasing demand for bioproducts in today's society, the main aim of biotechnological industries is to optimize their resources for maximum production at low cost. Most bioprocess industries employ microorganisms for production of primary metabolites (such as commercially important enzymes, ethanol, citric acid, nutritional amino acids, and many more), secondary metabolites (antibiotics, antioxidants, alkaloids, and many more), or biomass constituents (single cell protein, lipids, and others). However, living systems naturally produce a substance just in amounts required for their survival, after which, the cell employs certain mechanisms to stop the wastage of any more cellular resources in the production of the metabolite. The amount of metabolite thus obtained is obviously not sufficient for its industrial-scale production. To tackle this problem, the field of biochemical engineering has evolved, of which metabolic engineering is an inherent component.

4.1.1 Why Engineer Metabolism?

From the perspective of a biochemical engineer, the cell growing inside a bioreactor can be considered to be a factory, which takes in substrates and gives us products. This factory has its own production processes, which can be manipulated and optimized for enhanced production. These production processes are carried out using the metabolic pathways of the cell, in which the substrates get oxidized to produce energy and metabolites important for the growth and survival of the cell.

In any industry, one needs to be completely familiar with the process flow taking place, from the source of raw materials to the last product purification step being carried out. This needs extensive knowledge of the complete layout of the industry. Such information cannot be simply qualitative, but also needs to be quantitative so that the process can be optimized, thus preventing any wastage of resources. This is very similar to the cellular factory, which has a source of raw materials (uptake of substrates), a processing unit (the cellular metabolism), and a finished product dispatch unit (secretion of product), as depicted in Figure 4.1. In essence, metabolic engineering is the quantitative analysis and optimization of this factory. This entails a detailed knowledge of all the branch points and connecting nodes of the cell's metabolic network, and their influence on the flow and speed of the production process. In analogy to the various parameters affecting the production process in an industry, the metabolism of the cell is also affected by various parameters, which are expressed quantitatively by a set of control coefficients, as will be discussed later. Imbued with all this knowledge and the skills of genetically manipulating a cell, the biochemical engineer is able to reorient the metabolism of a cell to meet our specific needs.

4.1.2 Mathematical Models in Metabolic Rengineering

The metabolic pathway associated with the concerned production process is decided based upon a literature survey or from online metabolic databases, like KEGG (Gombert and Nielsen 2000; Kanehisa et al. 2017). A cell feeds on a substrate and utilizes it via metabolic pathways to give a product (secreted out of the cell) or biomass constituents (polymeric constituents present inside the cell, such as DNA, RNA, protein, and others). A metabolic pathway is, in essence, a linear/branched combination of chemical reactions. The individual compounds formed, and subsequently utilized via these reactions, are called intermediates. A general mass balance equation of all the aforementioned cellular components is given by (Stephanopoulos, Aristidou, and Nielsen 1998):

$$\sum Substrates = \sum Products + \sum Biomass\ constituents + \sum Intermediates \tag{4.1}$$

Each component of a chemical reaction has a stoichiometric coefficient. For substrates, this coefficient is −ve, as substrates are always consumed or depleted, while that for products is always +ve, as products are always formed. Biomass constituents and intermediates may have −ve or +ve stoichiometric coefficients, depending on whether they are being used up or produced in a particular reaction. To explain

this more clearly, let us take only the pathway metabolite into consideration of the equation, corresponding to metabolic intermediates. If there are "*i*" intermediates (*I*) being involved in "*j*" reactions, the last term on the RHS of the above equation is expanded as:

$$\begin{aligned} & g_{11}I_1 + g_{12}I_2 + g_{13}I_3 + \cdots + g_{1M}I_M \\ & \quad + g_{21}I_1 + g_{22}I_2 + g_{23}I_3 + \cdots + g_{2M}I_M \\ & \quad + \cdots + g_{j1}I_1 + g_{j2}I_2 + g_{j3}I_3 + \cdots + g_{jM}I_M \end{aligned}$$

FIGURE 4.1 Comparing cellular metabolism with the layout of a factory. (a) General Layout of a Pharmaceutical Industry. (*Continued*)

FIGURE 4.1 (Continued) Comparing cellular metabolism with the layout of a factory. (b) Overview of Cellular Metabolism.

where g is the stoichiometric coefficient of the respective intermediate. A better way to represent this is by using a matrix, as follows:

$$\begin{pmatrix} g_{11} & g_{12}\cdots & g_{1M} \\ g_{21} & g_{22}\cdots & g_{2M} \\ \vdots & \vdots & \vdots \\ g_{j1} & g_{j2} & g_{jM} \end{pmatrix} \cdot \begin{pmatrix} I_1 \\ I_2 \\ \vdots \\ I_M \end{pmatrix}$$

In a similar fashion, the other terms of the equation can be represented via matrices. This type of mathematical representation of cellular metabolism is called stoichiometric matrix model, where every row corresponds to a particular reaction, while every column corresponds to a particular metabolite. Thus, the above equation, in matrix notation is (Stephanopoulos, Aristidou, and Nielsen 1998):

$$AS + BP + \Gamma X_{macro} + GX_{met} = 0 \tag{4.2}$$

where A, B, Γ, and G represent the matrices of stoichiometric coefficients.

However, to alter the metabolism of cell, we also need to know the rates at which the metabolic intermediates are formed and/or consumed. This can be represented by rate matrix models as follows:

$$r_{met} = G^T \nu \tag{4.3}$$

where "r_{met}" is the net rate of formation or consumption of the intermediates, while "ν" is the rate matrix containing all the rates of the individual reactions of the metabolic pathway. The transpose of the stoichiometric coefficient matrix (G^T) is to be multiplied with the rate matrix to obtain the matrix r_{met}. Hence, the net rate of product formation can be expressed as:

$$r_{met} = \begin{pmatrix} g_{11} & g_{21} \cdots & g_{j1} \\ g_{12} & g_{22} \cdots & g_{j2} \\ \vdots & \vdots & \vdots \\ g_{1M} & g_{2M} & g_{jM} \end{pmatrix} \cdot \begin{pmatrix} \nu_1 \\ \nu_2 \\ \vdots \\ \nu_3 \end{pmatrix} \tag{4.4}$$

The stoichiometric matrix models and the rate matrix models are the first steps towards engineering cellular metabolism. It is on the basis of these models that we perform metabolic flux analysis and metabolic control analysis.

4.1.3 Metabolic Flux Analysis

Flux is the rate at which a metabolite flows into a particular metabolic pathway, which is influenced by the rate of the competing reactions (Stephanopoulos 1999). The rate matrix models discussed above are used to quantify the intracellular fluxes of the metabolic network. In any metabolic pathway, we need to identify fluxes, which we can measure experimentally, such as rate of secretion of certain intermediates into the medium or rate of formation of biomass constituents. We can thus segregate the rate matrix obtained above into two sections—one containing experimentally measured fluxes ($G_m^T \nu_m$) and the other containing the fluxes needing to be calculated ($G_c^T \nu_c$). If we assume the system to be at steady state, the net rate of formation or consumption of the intermediates becomes 0, reducing Equation 4.3 to:

$$G_m^T \nu_m + G_c^T \nu_c = 0 \tag{4.5}$$

$$\nu_c = -(G_c^T)^{-1} G_m^T \nu_m \tag{4.6}$$

Under certain conditions, however, this method of flux calculation fails, as reviewed by Weichert (Wiechert 2001). In these cases, one or more carbon atoms of a particular metabolite are replaced by the ^{13}C isotope. By using analytical instruments, such as NMR and GC-MS, we can track the rate at which these heavy carbon atoms flow through the metabolic pathway, allowing us to determine the metabolic flux.

After performing metabolic flux analysis, we obtain a flux map containing the flux split ratios of all the metabolites (Manish, Venkatesh, and Banerjee 2007; Sarma et al. 2017). We are also able to identify where the control points of the concerned pathway lie, and if they are stubborn or amenable to manipulation. However, for a quantitative estimation of the extent to which these control points can be manipulated and the subsequent effects on the corresponding fluxes, we perform a metabolic control analysis.

4.1.4 Metabolic Control Analysis

Every reaction occurring in the cell is catalyzed by enzymes, and metabolic control analysis is a measure of how much influence an enzyme has over the rate of the corresponding reaction. This is best expressed by the flux control coefficient (C^J), which describes the change in the flux of a metabolite (J) by an infinitesimal change in the enzyme activity (E) (Kacser and Burns 1973; Heinrich and Rapoport 1974b, 1974a):

$$C^J = \frac{E}{J}\frac{dJ}{dE} \tag{4.7}$$

Enzyme activity is a system parameter, which we can change at will, while flux is a system variable, which is completely dependent on the parameter and that we cannot influence directly.

Enzyme activity is generally manipulated by certain effectors (inhibitors or activators) that bind to the enzyme, thus influencing its job as a catalyst. The response coefficient describes the influence exerted by such an effector (e_i) on the flux of the metabolite (J_k):

$$R_{X_i}^{J_k} = \frac{e_i}{J_k}\frac{dJ_k}{de_i} \tag{4.8}$$

Another coefficient—the elasticity coefficient—expresses the relationship between metabolite concentration (X_j) and rate of the reaction (ν_i), both of which are system variables:

$$\varepsilon_{X_j}^{i} = \frac{X_j}{\nu_i}\frac{\partial \nu_i}{\partial X_j} \tag{4.9}$$

Going back to the analogy of a metabolic pathway with a factory layout, we get to know what changes need to be made at which points in the factory layout and by how much, after performing metabolic control analysis. Hereafter, the biochemical engineer performs genetic engineering or other techniques to effectively change the factory processes.

To manually analyze the mathematical models of any metabolic pathway is a daunting task due to the huge number of variables and parameters involved. Various software packages have been developed to facilitate metabolic engineering by automating the analysis of these mathematical models, after which we can optimize

the cell by experimental means. These software packages are evolving continuously, and they have been discussed in the next section.

4.2 SOFTWARES USED IN METABOLIC ENGINEERING

Software packages and tools are an integral part of metabolic engineering for modeling metabolism and analysis of the mathematical models discussed above. Manual planning of how to design optimized cells is limited by the huge number of genes and corresponding metabolic reactions. In this section, we shall discuss in brief the various software packages and tools commonly used for metabolic engineering. These software packages typically simulate the metabolic pathways in a quantitative perspective by means of constraints-based flux analysis, which employs various optimization techniques to maximize/minimize one or more system variables based on alteration effects of one or more system parameters (objective function). For instance, it is possible to maximize biomass formation rates by choosing the right constraints reflecting the physicochemical conditions affecting the metabolism of the cells (Kim, Kim, and Lee 2008; Park, Kim, and Lee 2009). Many algorithms have been developed, which can be used to identify mutable targets based on flux distribution of the metabolic pathway, such as minimization of metabolic adjustment (MoMA), regulatory on/off regularization (ROOM), OptKnock, OptStrain, and many others (Segre, Vitkup, and Church 2002; Shlomi, Berkman, and Ruppin 2005; Boghigian et al. 2010). A few of these software packages have been discussed below.

4.2.1 MatLab

Developed for the calculation and manipulation of matrices, MatLab undoubtedly finds many applications in implementation and analysis of mathematical models of cellular metabolism.

Solving these mathematical models, as described in the previous section, is in essence solving optimization problems to evaluate the consequences of engineering the strain on its metabolic capabilities. This requires the optimization tool embedded in MatLab, known as "optimtool." Exploiting MatLab as a platform to run a variety of programs, many packages have been developed for in silico analysis of metabolic fluxes. One such package is the PFA toolbox (Morales et al. 2016), which runs by means of MatLab functions, and solves every step of a typical metabolic flux analysis problem, including the proposed "*interval* MFA" and "*possibilistic* MFA" (provide interval estimates rather than point wise solutions when multiple flux values are possible). PFA toolbox uses YALMIP as the optimizing tool, and also has an inbuilt graphic user interface for *possibilistic* perspectives of parameters and constraints. Cipher for evolutionary design (CiED) is another example, which is compiled via MatLab, and is used to evolve microbial strains in silico using an optimization function selecting beneficial mutations combined with a genome-scale metabolic model (Chemler et al. 2010). The most famous example of the same is, however, FiatFlux, which shall be discussed in the following section.

4.2.2 FiatFlux

FiatFlux (Zamboni, Fischer, and Sauer 2005) consists of two modules. In the first module, called the RATIO module, metabolic flux ratios are calculated by analyzing ^{13}C-labelling experiments obtained from mass spectrometry (MS). These split ratios of in vivo fluxes are represented as probabilistic equations. The software is preconfigured to derive metabolic flux ratios from [1-^{13}C] and [U-^{13}C] labeled glucose-labeling experiments conducted in yeast, *E. coli*, *Bacillus subtilis*, and others. In the second module, known as the NETTO module, the aforementioned equations, along with biomass requirements, are imposed as constraints in mass balance calculations using the experimentally measured extracellular fluxes, thus yielding the net carbon fluxes within the metabolic pathway. The major advantage of this software is that it is user-friendly, and anyone can reorient the software to meet their specific needs, such as the use of a different ^{13}C-labeled substrate or a different organism.

4.2.3 OptFlux

OptFlux (Rocha et al. 2010) is an open-source, user-friendly and modular software very commonly used for metabolic design and engineering. Implemented using the programming language Java, OptFlux is fully compatible with SBML and *Cell Designer*. Apart from containing provisions for phenotype simulation and introduction of experimentally measured fluxes, OptFlux also acts as a platform for implementing the strain optimization algorithms OptKnock and OptStrain algorithms. Both these algorithms employ a bi-level optimization strategy to select the superior mutant strain. They also use mixed integrated linear programming (MILP), which has the ability to optimize one objective (such as biomass production) within a second competing objective (such as metabolite production) (Boghigian et al. 2010). However, for OptKnock and OptStrain to determine the optimum number of mutations suitable for strain design, genomic space for knockout candidates needs to be exhaustively searched, which is computationally expensive. To overcome this limitation, the OptGene algorithm was developed, which combines evolutionary algorithms (EA) and simulated annealing (SA) (Boghigian et al. 2010).

4.2.4 Other Softwares

13CFLUX-2 (Weitzel et al. 2013) is used to plan and analyze ^{13}C-labeling experiments to determine in vivo metabolic fluxes. It uses a modified XML language, called FluxML. However, 13CFLUX-2 requires that several steps be carried out manually. ^{13}C-labeling-based metabolic flux analysis can be automated and standardized using Flux-P (Ebert et al. 2012), which is based on the Bio-jETI workflow framework. Isotopomer Network Compartmental Analysis (INCA) (Young 2014) is the first software able to perform ^{13}C metabolic flux analysis under both steady-state and nonstationary (transient labeling experiments) conditions, thus expanding the scope of metabolic engineering to photosynthetic organisms and mammalian cultures.

Although the software packages discussed above are very commonly used to plan in silico how to manipulate the cells, they are required to analyze data from huge datasets

(such as the pathway databases or protein databases), which becomes computationally expensive. Moreover, a lot of experimental work, which is used as a constraint in flux balance analysis, is involved before one can use these software packages. Performing these experiments for every strain separately is time-consuming, and calls for the use of highly sophisticated and expensive equipment. This problem of redundancy is best solved by the use of machine learning techniques, via which the machine is trained to predict results for repeated rounds of engineering cellular metabolism.

4.3 MACHINE LEARNING TECHNIQUES

Machine learning emphasizes the development of programs that accesses data and learns from experiences. The process of learning starts with the observation of data directly through instructions and experiences, followed by analysis of the data to check for possible patterns and finally perform decision-making. The popularly used machine learning algorithms are:

- *Supervised Machine Learning*: A data mining task where a function is inferred from labeled training data. This training data consists of a set of paired training examples, which has an input object or vector and the output object in the form of a supervisory signal. Based on the analysis of training data, the inferred function is generated helping to map new examples on the data set. The popular supervised machine learning algorithms are Logistic Regression, Decision Trees, Support Vector Machines (SVM), K—Nearest Neighbor (KNN), Naïve Bayes, Random Forest, Linear Regression, and Polynomial Regression algorithm.
- *Unsupervised Machine Learning*: In case of unsupervised learning, the objective is to find hidden information or structure from unlabeled training data. The popular unsupervised learning algorithms are K—Means Clustering, Hierarchical Clustering, and Hidden Markov model.
- *Semisupervised Machine Learning*: A variant of supervised learning which basically uses larger number of unlabeled data to train smaller number of labeled data. It is a combination of supervised and unsupervised machine learning technique.
- *Reinforced Machine Learning*: A machine learning technique inspired by behavioristic psychology focusing on the mechanisms of software agents taking actions in an environment in order to maximize the concept of cumulative award.

At the core of engineering metabolism lies the most important step of characterizing metabolism. Metabolomics is the field of study concerned with discovery of the complete set of metabolites in a cell through various analytical techniques such as GC-MS. This is accompanied by determining how these metabolites interact with each other, or, in other words, characterizing the metabolic pathways. To store the enormous amounts of data discovered through these methods, various databases have been created, such as KEGG (Kanehisa et al. 2017), EcoCyc (Karp 2002a), MetaCyc (Karp 2002b), and many more.

Research in the field of biology is an iterative cycle between mathematically modeling natural phenomenon, implementing the model via wet lab experimentations, and subsequently improving the model to better explain or manipulate the biological process. This cycle maps different levels of biological organization. For instance, modeling individual metabolic reactions or manipulating individual enzyme molecules ultimately has a consequence on the organism's metabolism at large (Kell 2006). Machine learning techniques and concepts aim to automate this iterative cycle for an enhanced knowledge discovery. Machine learning, in essence, involves teaching predictive models to the machine so that it can predict the outcome for any input. These predictive models are based on the observations of a system, thus allowing the experimental nature of biology to be a good candidate for implementing machine learning techniques. Such predictive models should be interpretable, easy to use, and flexible (Geurts, Irrthum, and Wehenkel 2009).

Given the annotated genome of a cell, it is possible to predict the metabolic pathways present in the cell. This has been carried out on *Helicobacter pylori*, by developing the PathoLogic algorithm, which uses MetaCyc or EcoCyc databases as a template for predicting the pathways (Paley and Karp 2002). This algorithm searches for pathways in the aforementioned databases containing at least one enzyme, which is common with the organism whose metabolism is being predicted. Once a number of pathways from the databases are chosen, the algorithm starts eliminating them one by one after every iteration using a set of manually described criteria. The last pathway to remain is predicted to be present in the concerned organism. However, as the size of the MetaCyc database increased, the algorithm started giving many false positive predictions as well. To correct this issue, in a later work (Dale, Popescu, and Karp 2010), a data-driven, transparent, and tunable pathway prediction algorithm was developed based on various machine learning techniques. A "gold-standard" dataset, containing various datasets from a wide variety of organisms, was constructed by taking information from many databases to train the machine learning algorithms. Various features were defined and computed for all the pathways in the gold standard dataset, which was subsequently divided into training and test sets. The training data were used for feature selection and parameter estimation for the predictor types of the data set, and the test data were used for the evaluation of the predictors. The PathoLogic algorithm takes as input the newly sequenced and annotated genome, and predicts the metabolic pathways using the best-evaluated predictor. A similar approach was followed when mining high-dimensional metabolic data for patterns to screen metabolic-based diseases using machine learning (Baumgartner, Böhm, and Baumgartner 2005). Based on a database of metabolic data curated from cases of phenylalanine hydroxylase deficiency, medium-chain acyl-CoA dehydrogenase deficiency, and 3-methylcrotonyl CoA carboxylase deficiency, various classification models were constructed and their parameters estimated. The validated model can be used for an automated prediction and diagnosis of metabolic disorders when certain markers of the patient's metabolism are given as input.

Machine learning has also been applied in the construction of metabolic networks for the calculation of parameters in stoichiometric and kinetic models. The approach has significant application in feature analysis of models relevant to bioreactors. SVF, RF, and several machine leaning algorithms have been used in the analysis of points

in NMR spectra and MS spectrograms. Some of the metabolomics software also uses machine-learning methods, which provide mechanisms for metabolomics analysis from data preprocessing and metabolic network analysis. The popularly used tools are mummychog, Cytoscape, Galaxy, FingerID, SIRIUS, Metaboanalyst, KNIME, Weka, Orange, and TensorFlow (Shannon et al. 2003; Li et al. 2013; Guitton et al. 2017).

Some of the commonly used machine learning techniques used in engineering of cellular metabolism, or for analysis of metabolomics data, have been discussed below.

4.3.1 Decision Tree Learning

The most commonly used nonparametric method, decision tree learning showcase excellent interpretability, efficiency, and accuracy (Geurts, Irrthum, and Wehenkel 2009). A decision tree is, in fact, a simple flowchart, where each node asks a "Yes/No" question. For each node, there are two branches representing either a "Yes" or a "No." The questions asked at each node can be described as criteria of some sort to decide whether the process proceeds to the next step or if it should be stopped at that step (Baumgartner, Böhm, and Baumgartner 2005; Dale, Popescu, and Karp 2010). The ranking of nodes is decided by the capacity of a particular feature to shortlist the number of predictions, until a particular solution is reached (Boccard, Veuthey, and Rudaz 2010). A decision tree has basically a flowchart-like structure where the internal nodes present the test for an attribute and the branches are basically the resultant outcomes of the tests conducted and the leaf nodes are the class labels. The topmost node is the root node. The popularly used decision tree learning algorithms are ID3 (Iterative Dichotomiser), C4.5, Classification and Regression Tree (CART) algorithm, CHAID and MARS (Breiman et al. 1984; Quinlan 1996). A detailed explanation of decision trees, how the results can be represented in tabular format, and their applications in biology has been provided by Geurts, Irrthum, and Wehenkel (2009). A cell can employ many metabolic pathways to synthesize a particular compound. Carbonell et al. (2011) proposed a novel retrosynthetic approach to determine which metabolic pathway to target to enhance production of the compound of interest using decision trees. The decision tree thus developed aimed to rank pathways on the basis of thermodynamic feasibility of the individual reactions of each pathway, performance and homogeneity of each enzyme involved, toxicity of the final product, and nominal flux of each pathway. Decision trees have also been used to predict the stability of a protein upon point mutations (change of a particular amino acid) with 82% accuracy (Huang, Gromiha, and Ho 2007).

Diabetes is a metabolic disorder, which has affected millions of people across the globe. The accurate prediction and detection of diabetes has always been a challenge. The study by AVN et al. (2017) mentioned the use of the Decision Tree C5.0 algorithm for the diagnosis and prediction of metabolic disorder-related diabetic diseases. The C5.0 algorithm in machine learning is used for the design of the decision tree with emphasis on information gain and entropy for the creation of the tree. Each stage is divided for absolute attributes, and the topmost attribute is predicted having maximum information gain.

Another study by Karimi-Alavijeh, Jalili, and Sadeghi (2016) has implemented the support vector machine (SVM), decision tree, and also the hybrid of the two

techniques, for the prediction of metabolic syndrome from Isfahan Cohort Study participants. Participants without the metabolic syndrome were selected according to ATPIII criteria and were considered for the study. The variables included in the study were gender, age, weight, and physical aspects such as body mass, waist circumference, and twenty other variables. The sensitivity, specificity, and accuracy criteria were considered for validation purposes. The results revealed the SVM method to be more accurate and efficient than the decision tree-based approach based on sensitivity and specificity criteria.

4.3.2 Association Rule Learning

Association rule learning is used to determine multifeature relationships between two disjoint datasets. If the two datasets are A and B, the relation between them is represented as $A \rightarrow B$, which is the basis of association rule learning. This association is usually a many-to-one relation. For example, every gene sequence annotated as belonging to a mitochondrion belongs to a eukaryotic cell, and this can be represented as "Gene: Mitochondrial → Origin: Eukaryota." Boudellioua et al. (2016) developed a system to envisage metabolic pathways in prokaryotes by obtaining data from UniProtKB/Swiss-Prot database, a protein sequence database. The association algorithm used was Apriori, which could determine relations between various attributes of UniProtKB, such as InterPro signatures and organism taxonomy. Association rule mining is also the basis of revealing microbial genotype-phenotype associations by using algorithms such as NETCAR (Tamura and D'haeseleer 2008) and CPAR (MacDonald and Beiko 2010). The former was used to mine out clusters of orthologous (COGs) of proteins related to six microbial phenotypes, via association rule learning, from a dataset of 11,969 COG profiles (corresponding to 155 prokaryotic organisms). COGs of proteins provide a link between the organism's genotype and phenotype profiles. The latter also works on a similar basis by using evolutionary ancestry of the organism as a pattern in phenotype prediction. In essence, CPAR generalizes over the maximum number of samples in a class, and then focuses on the samples not covered by the generalization. CPAR has been reported to provide a slightly higher accuracy coupled with a faster running time over NETCAR.

4.3.3 Clustering

Cluster analysis is a multivariate statistical method involving sorting data into groups/clusters, such that data of a certain cluster share a specific property not present in other clusters. Data obtained from metabolomics experiments can be grouped into clusters based on several features. The parameters usually chosen as the basis of segregation are intensities of the metabolites and experimental conditions. The clustering of metabolomics data can be broadly divided into two categories (Figure 4.2)—one-mode clustering and two-mode clustering (Hageman et al. 2008). In *one-mode clustering*, only a single parameter is changed at a time, and it is further subdivided into two methods based on the parameter being changed. In *metabolite-based clustering*, a biological system is subjected to a particular experimental condition, and the numerous metabolites of the cell are grouped into clusters based on their relative intensities under

FIGURE 4.2 Types of clustering methods for metabolomics data.

that experimental condition. It is assumed that metabolites of a particular cluster are obtained from the same biosynthetic pathway, or are subject to the same control loops. This method of clustering allows identification of biologically important metabolic markers, which can be used to identify species or even diseases (Meinicke et al. 2008). In *sample-based clustering*, on the other hand, the intensities of a single metabolite under different experimental conditions are used as the basis for grouping the metabolites. This method of clustering reveals information on how the metabolism of a cell can be correlated with its phenotypic features (Fiehn 2002). *Two-mode clustering* involves changing both the parameters simultaneously and clustering metabolites and experimental conditions together accordingly. The algorithms of hierarchical clustering, k-means clustering, and one-dimensional self-organizing maps (SOM) fall under the one-mode clustering approach (Beckonert et al. 2003; Meinicke et al. 2008), while the two-mode clustering relies on global optimization approaches, including simulated annealing (SA) and tabu search (TS) algorithms (Mechelen, Bock, and De Boeck 2004; Prelić et al. 2006). Hageman et al. (2008) developed a genetic algorithm to perform two-mode clustering of metabolomics data, thus determining which groups of metabolites respond to which experimental conditions. More details on genetic algorithms have been discussed in the following sections. Cluster analysis has been applied during the metabolic profiling of seven potato genotypes (Roessner 2001). This work revealed that although the phenotype was the same (increased accumulation of starch), the genotypes formed prominent clusters, implying that different metabolic conditions of a system can have the same phenotypic result (Trethewey 2004). However, cluster analysis has certain disadvantages, such as difficulty in interpretation (as the experimental conditions inevitably overlap), propagation of error (formed due to similar reasons), and poor reproducibility of results, as reviewed by Andreopoulos et al. (2009).

4.3.4 Bayesian Networks

Bayesian networks of two or more variables imply probabilistic interactions, or causal relationships between them, represented in the form of directed acyclic graphs. In terms of metabolic engineering, Bayesian networks may investigate relations between two metabolic reactions and/or their fluxes, as well. Kim et al. (2011) developed a framework to determine the effect of perturbations (genetic or environmental) on the metabolic network of the system. Intrinsic to this work was the development of metabolic modules, which are essentially clusters of metabolic pathways based on their fluxes. Hence, after calculating the metabolic fluxes by means of constraint-based analysis, the developed framework (referred to as FMB) performed Bayesian network analysis at the level of metabolic modules to reveal the effect of perturbations on the metabolism of the cell. Bayesian networks have also been used in the *ab initio* reconstruction of metabolic pathways based on gene expression data (Zhao et al. 2012). Qi et al. (2014) used Bayesian network analysis to infer protein–protein interactions underlying metabolic network reconstruction, keeping gene-product relationships (obtained from KEGG database) as a constraint. Such kind of knowledge-constrained based probabilistic inference reduces the solution space and increases the accuracy of prediction. In a similar work (Zhu et al. 2012), a Bayesian network analysis was used to reconstruct the metabolic network of a dividing yeast population by "stitching together" six types of data—metabolite concentration profiles, RNA expression patterns, variations in DNA, DNA-protein binding, protein–metabolite interaction, and protein–protein interaction. The network thus constructed has also been used to derive causal relationships of the metabolites with the genome, thus allowing the discovery of novel genes that could potentially regulate the cellular metabolism. In order to determine relevant biological markers for the identification of *Bacillus* spores, a combination of Bayesian networks and genetic algorithms was used to reduce the data dimensionality, that is, choose only a few of the data generated by mass spectroscopy experiments (Correa and Goodacre 2011).

4.3.5 Genetic Algorithms (GA)

Genetic algorithms are effective multivariate analysis approaches for analyzing metabolomics datasets (Zou and Tolstikov 2009). Inspired by the process of natural selection, which promises maximum survival of the fittest species under adverse conditions, GA is used to fish out certain critical parameters or the smallest subsets out of the datasets which can effectively describe the datasets (Yeo et al. 2015). GA starts with organizing the given dataset into smaller subsets, each of which is tested for its prediction ability. The subsets obtained in the next step of GA of higher classification accuracy are created by mechanisms inspired by reproduction, selection, mutation, crossover, and migration. This process is repeated for subsequent generations, until a subset with a predetermined accuracy level is reached (Trevino and Falciani 2006). In this way, the entire population of data is "evolved," or optimized, towards a global optimum. For example, GA were developed to select the best parameters suitable for preprocessing of LC-MS-based metabolomics data (Yeo et al. 2015). Boghigian et al. (2010) performed metabolic engineering on a strain of *E. coli* by means of

GAs. Suitable gene targets were identified by evolving the metabolic network with thermodynamically-favorable pathways. This was achieved by developing an optimization framework based on GAs and thermodynamically weighted elementary modes analysis (which, in essence, is metabolic flux analysis). Using this framework, they were able to optimize biomass and product formation simultaneously.

The term "metabolic engineering" was initially coined in the 1990s. At that time, metabolic engineering was a more theoretical branch of biochemical engineering. Since then, this field of research has diversified to a great extent, becoming more concrete rather than an abstract concept. Moreover, it has become a multidisciplinary area of research, which further facilitates its use in various biotechnological industries. However, metabolic engineering has still not matured as an independent field of research due to various reasons, as discussed in the next section.

4.4 FUTURE ASPECTS

Metabolic engineering is rapidly becoming an important parcel of upstream processing in various bioprocess industries. At present, however, research in the field of metabolic engineering is hampered by the huge amount of time taken to perform various experiments, and the need for a sophisticated and expensive experimental setup, and the attached costs. Machine learning methodologies are able to help in this regard, as they allow the engineer to predict outcome of strain optimization procedures, thus eliminating the need for redundant "trial-and-error" experiments for every strain being employed by the industry. Once an optimized strain is designed by in silico means, it can be validated in vitro by experimental means, and the same algorithm can be reused for optimizing another taxonomically similar strain, both of which share similar metabolic features. In this way, the machine is trained to virtually engineer the cellular metabolism for every varied input (wild-type strain). However, as of now, machine-learning techniques are mostly limited to analysis of metabolomics data. Hence, there is a long way to go before computer science can effectively play an integral part in biological sciences for the ease of research and production processes in multiple industries. This gap can be bridged by making more and researchers and industry stakeholders aware of the huge potential of metabolic engineering approaches in the reduction of costs, while enhancing productivity of bioprocess industries. Moreover, a common platform needs to be developed for effective communication between biochemical engineers and computer science engineers.

REFERENCES

Andreopoulos, B., A. An, X. Wang, and M. Schroeder. 2009. "A Roadmap of Clustering Algorithms: Finding a Match for a Biomedical Application." *Briefings in Bioinformatics* 10 (3): 297–314. doi:10.1093/bib/bbn058.

Avn, U., R. Gullapudi, M. Gonuguntla, and N. Rao MR. 2017. "Machine Learning Techniques For Predicting The Wellbeing In Diabetes." *International Journal of Pure and Applied Mathematics* 116 (6): 185–189.

Baumgartner, C., C. Böhm, and D. Baumgartner. 2005. "Modelling of Classification Rules on Metabolic Patterns Including Machine Learning and Expert Knowledge." *Journal of Biomedical Informatics* 38 (2): 89–98. doi:10.1016/j.jbi.2004.08.009.

Beckonert, O., M. E. Bollard, T. M.D. Ebbels, H. C. Keun, H. Antti, E. Holmes, J. C. Lindon, and J. K. Nicholson. 2003. "NMR-Based Metabonomic Toxicity Classification: Hierarchical Cluster Analysis and K-Nearest-Neighbour Approaches." *Analytica Chimica Acta* 490 (1–2): 3–15. doi:10.1016/S0003-2670(03)00060-6.

Boccard, J., J. L. Veuthey, and S. Rudaz. 2010. "Knowledge Discovery in Metabolomics: An Overview of MS Data Handling." *Journal of Separation Science* 33 (3): 290–304. doi:10.1002/jssc.200900609.

Boghigian, B. A., H. Shi, K. Lee, and B. A. Pfeifer. 2010. "Utilizing Elementary Mode Analysis, Pathway Thermodynamics, and a Genetic Algorithm for Metabolic Flux Determination and Optimal Metabolic Network Design." *BMC Systems Biology* 4: 49. doi:10.1186/1752-0509-4-49.

Boudellioua, I., R. Saidi, R. Hoehndorf, M. J. Martin, and V. Solovyev. 2016. "Prediction of Metabolic Pathway Involvement in Prokaryotic Uniprotkb Data by Association Rule Mining." *PLOS ONE* 11 (7): 1–16. doi:10.1371/journal.pone.0158896.

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and Regression Trees*. Wadsworth, Monterey, California.

Carbonell, P., A. G. Planson, D. Fichera, and J. L. Faulon. 2011. "A Retrosynthetic Biology Approach to Metabolic Pathway Design for Therapeutic Production." *BMC Systems Biology* 5: 122. doi:10.1186/1752-0509-5-122.

Chemler, J. A., Z. L. Fowler, K. P. McHugh, and M. A. G. Koffas. 2010. "Improving NADPH Availability for Natural Product Biosynthesis in Escherichia Coli by Metabolic Engineering." *Metabolic Engineering* 12 (2): 96–104. Elsevier. doi:10.1016/j.ymben.2009.07.003.

Correa, E. and R. Goodacre. 2011. "A Genetic Algorithm-Bayesian Network Approach for the Analysis of Metabolomics and Spectroscopic Data: Application to the Rapid Identification of Bacillus Spores and Classification of Bacillus Species." *BMC Bioinformatics* 12 (1): 33. BioMed Central Ltd. doi:10.1186/1471-2105-12-33.

Dale, J. M., L. Popescu, and P. D. Karp. 2010. "Machine Learning Methods for Metabolic Pathway Prediction." *BMC Bioinforma* 11: 15. doi:10.1186/1471-2105-11-15.

Ebert, B. E., A.-L. Lamprecht, B. Steffen, and L. M. Blank. 2012. "Flux-P: Automating Metabolic Flux Analysis." *Metabolites* 2 (4): 872–90. doi:10.3390/metabo2040872.

Fiehn, O. 2002. "Metabolomics - The Link between Genotypes and Phenotypes." *Plant Molecular Biology* 48 (1–2): 155–71. doi:10.1023/A:1013713905833.

Geurts, P., A. Irrthum, and L. Wehenkel. 2009. "Supervised Learning with Decision Tree-Based Methods in Computational and Systems Biology." *Molecular BioSystems* 5 (12): 1593. doi:10.1039/b907946g.

Gombert, A. K. and J. Nielsen. 2000. "Mathematical Modelling of Metabolism." *Current Opinion in Biotechnology* 11 (2): 180–86. doi:10.1016/S0958-1669(00)00079-3.

Guitton, Y., M. Tremblay-Franco, G. L. Corguillé, J. F. Martin, M. Pétéra, P. Roger-Mele, and A. Delabrière et al. 2017. "Create, Run, Share, Publish, and Reference Your LC–MS, FIA–MS, GC–MS, and NMR Data Analysis Workflows with the Workflow4Metabolomics 3.0 Galaxy Online Infrastructure for Metabolomics." *International Journal of Biochemistry and Cell Biology* 93: 89–101. Elsevier Ltd. doi:10.1016/j.biocel.2017.07.002.

Hageman, J. A., R. A. van den Berg, J. A. Westerhuis, M. J. van der Werf, and A. K. Smilde. 2008. "Genetic Algorithm Based Two-Mode Clustering of Metabolomics Data." *Metabolomics* 4 (2): 141–49. doi:10.1007/s11306-008-0105-7.

Heinrich, R. and T. A. Rapoport. 1974a. "A Linear Steady-State Treatment of Enzymatic Chains." *European Journal of Biochemistry* 42: 97–105.

Heinrich, R. and T. A. Rapoport. 1974b. "A Linear Steady-State Treatment of Enzymatic Chains. General Properties, Control and Effector Strength." *European Journal of Biochemistry* 42 (1): 89–95.

Huang, L.-T., M. Michael Gromiha, and S.-Y. Ho. 2007. "iPTREE-STAB: Interpretable Decision Tree Based Method for Predicting Protein Stability Changes upon Mutations." *Bioinformatics (Oxford, England)* 23 (10): 1292–93. doi:10.1093/bioinformatics/btm100.

Kacser, H. and J. Burns. 1973. "The Control of Flux." *Symposia of the Society for Experimental Biology* 27: 65–104.

Kanehisa, M., M. Furumichi, M. Tanabe, Y. Sato, and K. Morishima. 2017. "KEGG: New Perspectives on Genomes, Pathways, Diseases and Drugs." *Nucleic Acids Research* 45 (D1): D353–61. doi:10.1093/nar/gkw1092.

Karimi-Alavijeh, F., S. Jalili, and M. Sadeghi. 2016. "Predicting Metabolic Syndrome Using Decision Tree and Support Vector Machine Methods." *ARYA Atherosclerosis* 12 (3): 146–152.

Karp, P. D. 2002a. "The EcoCyc Database." *Nucleic Acids Research* 30 (1): 56–58. doi:10.1093/nar/30.1.56.

Karp, P. D. 2002b. "The MetaCyc Database." *Nucleic Acids Research* 30 (1): 59–61. doi:10.1093/nar/30.1.59.

Kell, D. B. 2006. "Metabolomics, Modelling and Machine Learning in Systems Biology - Towards an Understanding of the Languages of Cells: Delivered on 3 July 2005 at the 30th FEBS Congress and 9th IUBMB Conference in Budapest." *FEBS Journal* 273 (5): 873–94. doi:10.1111/j.1742-4658.2006.05136.x.

Kim, H. U., T. Y. Kim, and S. Y. Lee. 2008. "Metabolic Flux Analysis and Metabolic Engineering of Microorganisms." *Molecular BioSystems* 4 (2): 113–20. doi:10.1039/B712395G.

Kim, H., T. Kim, and S. Lee. 2011. "Framework for Network Modularization and Bayesian Network Analysis to Investigate the Perturbed Metabolic Network." *BMC Systems Biology* 5 (Suppl 2): S14. BioMed Central Ltd. doi:10.1186/1752-0509-5-S2-S14.

Li, S., Y. Park, S. Duraisingham, F. H. Strobel, N. Khan, Q. A. Soltow, D. P. Jones, and B. Pulendran. 2013. "Predicting Network Activity from High Throughput Metabolomics." *PLoS Computational Biology* 9 (7). doi:10.1371/journal.pcbi.1003123.

MacDonald, N. J. and R. G. Beiko. 2010. "Efficient Learning of Microbial Genotype-Phenotype Association Rules." *Bioinformatics* 26 (15): 1834–40. doi:10.1093/bioinformatics/btq305.

Manish, S., K. V. Venkatesh, and R. Banerjee. 2007. "Metabolic Flux Analysis of Biological Hydrogen Production by Escherichia Coli." *International Journal of Hydrogen Energy* 32 (16): 3820–30. doi:10.1016/j.ijhydene.2007.03.033.

Mechelen, I. V., H. H. Bock, and P. De Boeck. 2004. "Two-Mode Clustering Methods: A Structured Overview." *Statistical Methods in Medical Research* 13 (5): 363–94. doi:10.1191/0962280204sm373ra.

Meinicke, P., T. Lingner, A. Kaever, K. Feussner, C. Göbel, I. Feussner, P. Karlovsky, and B. Morgenstern. 2008. "Metabolite-Based Clustering and Visualization of Mass Spectrometry Data Using One-Dimensional Self-Organizing Maps." *Algorithms for Molecular Biology* 3 (1): 1–18. doi:10.1186/1748-7188-3-9.

Morales, Y., G. Bosque, J. Vehí, J. Picó, and F. Llaneras. 2016. "PFA Toolbox: A MATLAB Tool for Metabolic Flux Analysis." *BMC Systems Biology* 10 (1): 1–10. BMC Systems Biology. doi:10.1186/s12918-016-0284-1.

Paley, S. M. and P. D. Karp. 2002. "Evaluation of Computational Metabolic Pathways Predictions for Helicobacter Pylori." *Bioinformatics* 18 (5): 715–24.

Park, J. M., T. Y. Kim, and S. Y. Lee. 2009. "Constraints-Based Genome-Scale Metabolic Simulation for Systems Metabolic Engineering." *Biotechnology Advances* 27 (6): 979–88. Elsevier B.V. doi:10.1016/j.biotechadv.2009.05.019.

Prelić, A., S. Bleuler, P. Zimmermann, A. Wille, P. Bühlmann, W. Gruissem, L. Hennig, L. Thiele, and E. Zitzler. 2006. "A Systematic Comparison and Evaluation of Biclustering Methods for Gene Expression Data." *Bioinformatics* 22 (9): 1122–29. doi:10.1093/bioinformatics/btl060.

Qi, Q., J. Li, and J. Cheng. 2014. "Reconstruction of Metabolic Pathways by Combining Probabilistic Graphical Model-Based and Knowledge-Based Methods." *BMC Proceedings* 8 (Suppl 6): 1–10. doi:10.1186/1753-6561-8-S6-S5.

Quinlan, J. R. 1996. "Improved Use of Continuous Attributes in C4.5." *Journal of Artificial Intelligence Research* 4: 77–90. doi:10.1613/jair.279.

Rocha, I., P. Maia, P. Evangelista, P. Vilaça, S. Soares, J. P. Pinto, J. Nielsen, K. R. Patil, E. C. Ferreira, and M. Rocha. 2010. "OptFlux: An Open-Source Software Platform for in Silico Metabolic Engineering." *BMC Systems Biology* 4. doi:10.1186/1752-0509-4-45.

Roessner, U. 2001. "Metabolic Profiling Allows Comprehensive Phenotyping of Genetically or Environmentally Modified Plant Systems." *The Plant Cell Online* 13 (1): 11–29. doi:10.1105/tpc.13.1.11.

Sarma, S., A. Anand, V. K. Dubey, and V. S. Moholkar. 2017. "Metabolic Flux Network Analysis of Hydrogen Production from Crude Glycerol by Clostridium Pasteurianum." *Bioresource Technology* 242: 169–77. Elsevier Ltd. doi:10.1016/j.biortech.2017.03.168.

Segre, D., D. Vitkup, and G. M Church. 2002. "Analysis of Optimality in Natural and Perturbed Metabolic Networks." *Proceedings of the National Academy of Sciences USA* 99 (23), 15112–15117.

Shannon, P., A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. 2003. "Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks." *Genome Research* 13: 2498–2504. doi:10.1101/gr.1239303.metabolite.

Shlomi, T., O. Berkman, and E. Ruppin. 2005. "Regulatory On/off Minimization of Metabolic Flux Changes after Genetic Perturbations." *Proceedings of the National Academy of Sciences USA* 102 (21): 7695–7700.

Stephanopoulos, G. 1999. "Metabolic Fluxes and Metabolic Engineering." *Metabolic Engineering* 1 (1): 1–11. doi:10.1006/mben.1998.0101.

Stephanopoulos, G. N., A. A. Aristidou, and J. Nielsen. 1998. *Metabolic Engineering: Principles and Methodologies*. Metabolic Engineering. Vol. 54. California, USA doi:10.1016/B978-0-12-666260-3.50019-4.

Tamura, M. and P. D'haeseleer. 2008. "Microbial Genotype-Phenotype Mapping by Class Association Rule Mining." *Bioinformatics* 24 (13): 1523–29. doi:10.1093/bioinformatics/btn210.

Trethewey, R. N. 2004. "Metabolite Profiling as an Aid to Metabolic Engineering in Plants." *Current Opinion in Plant Biology* 7 (2): 196–201. doi:10.1016/j.pbi.2003.12.003.

Trevino, V. and F. Falciani. 2006. "GALGO: An R Package for Multivariate Variable Selection Using Genetic Algorithms." *Bioinformatics* 22 (9): 1154–56. doi:10.1093/bioinformatics/btl074.

Weitzel, M., K. Nöh, T. Dalman, S. Niedenführ, B. Stute, and W. Wiechert. 2013. "13CFLUX2 - High-Performance Software Suite for 13C-Metabolic Flux Analysis." *Bioinformatics* 29 (1): 143–45. doi:10.1093/bioinformatics/bts646.

Wiechert, W. 2001. "13C Metabolic Flux Analysis." *Metabolic Engineering* 3 (3): 195–206. doi:10.1006/mben.2001.0187.

Yeo, H. C., B. K.-s. Chung, W. Chong, J. X. Chin, K. S. Ang, M. Lakshmanan, Y. S. Ho, and D.-Y. Lee. 2015. "A Genetic Algorithm-Based Approach for Pre-Processing Metabolomics and Lipidomics LC – MS Data." *Metabolomics* 12: 5. Springer US. doi:10.1007/s11306-015-0884-6.

Young, J. D. 2014. "INCA: A Computational Platform for Isotopically Non-Stationary Metabolic Flux Analysis." *Bioinformatics* 30 (9): 1333–35. doi:10.1093/bioinformatics/btu015.

Zamboni, N., E. Fischer, and U. Sauer. 2005. "FiatFlux—a Software for Metabolic Flux Analysis from 13C-Glucose Experiments." *BMC Bioinformatics* 6: 209. doi:10.1186/1471-2105-6-209.

Zhao, Y., M. H. Chen, B. Pei, D. Rowe, D. G. Shin, W. Xie, F. Yu, and L. Kuo. 2012. "A Bayesian Approach to Pathway Analysis by Integrating Gene-Gene Functional Directions and Microarray Data." *Statistics in Biosciences* 4 (1): 105–31. doi:10.1007/s12561-011-9046-1.

Zhu, J., P. Sova, Q. Xu, K. M. Dombek, E. Y. Xu, H. Vu, Z. Tu, R. B. Brem, R. E. Bumgarner, and E. E. Schadt. 2012. "Stitching Together Multiple Data Dimensions Reveals Interacting Metabolomic and Transcriptomic Networks That Modulate Cell Regulation." *PLoS Biology* 10 (4): e1001301. doi:10.1371/journal.pbio.1001301.

Zou, W. and V. V. Tolstikov. 2009. "Pattern Recognition and Pathway Analysis with Genetic Algorithms in Mass Spectrometry Based Metabolomics." *Algorithms* 2 (2): 638–66. doi:10.3390/a2020638.

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

5 Big Data and Transcriptomics

Sudharsana Sundarrajan, Sajitha Lulu, and Mohanapriya Arumugam

CONTENTS

5.1 Background 73
5.2 Methods in Transcriptomics 74
5.3 Microarray 74
5.3.1 Data Analysis 76
5.3.1.1 Identification of Differentially Expressed Genes 76
5.3.1.2 Clustering Gene Expression Data 77
5.3.1.3 Classification 78
5.3.1.4 Biological Significance 79
5.3.2 Quality Control 80
5.3.2.1 Monitoring RNA and Array Quality 80
5.3.3 Tools and Databases 80
5.3.4 Applications in Health Care 81
5.4 Deep Sequencing (Ref-Seq) 83
5.4.1 Data Processing 83
5.4.1.1 Quality Assessment 83
5.4.1.2 Read Mapping 84
5.4.1.3 Read Quantification and Counting 84
5.4.1.4 Transcript Identification and Alignment 85
5.4.1.5 Transcript Quantification 85
5.4.2 Quality Control 85
5.4.2.1 Low-Level Metrics 86
5.4.2.2 High-Level Metrics 86
5.4.3 Tools and Databases 86
5.4.4 Applications in Health Care 87
5.5 Future Prospects 87
References 88

5.1 BACKGROUND

DNA encodes the information required to determine the properties and functions of the cells in living organisms. Using this blueprint, the cells dynamically access and translate specific instructions via gene expression by switching on and off a specific set of genes. The data encoded in the particular genes are transcribed into

RNA molecules, which are further translated into proteins or directly utilized to fine tune the gene expression. The set of RNAs transcribed at certain time and condition reveals the current state of the cell. Transcriptome analysis characterizes the transcriptional activity by focusing on the subset of target genes (and transcripts) or profile thousands of genes in parallel to provide a global picture of cellular function. Thus, gene expression analysis provides a snapshot of actively expressed genes and transcripts under various conditions and time.

The term transcriptome includes all kinds of complementary RNA synthesized during transcription of the genome at a point of time. The microRNA (miRNA) and messenger RNA (mRNA) are of greatest interest in transcriptome research. The expression levels of mRNA reflects the gene activity of the cell, while the difference in the miRNA expression levels, which is a major posttranslational regulator of the gene activity, are thought to be associated with several cellular functions. Global analysis of gene expression is a major advancement in biomedical research. The analysis of RNA expression has three fundamental applications: disease subclassification, identification of key genes, and elucidation of biological pathways.[1]

The current chapter discusses the current state-of-the-art technologies in transcriptome research. In addition, the pros and cons of the technologies will be discussed along with their potential application in the health care sector.

5.2 METHODS IN TRANSCRIPTOMICS

Studies on individual transcripts started in early 1970s. In 1980s, low-throughput sequencing methods, namely Sanger sequencing, were used to sequence random transcripts called expressed sequence tags (ESTs). With the advent of high-throughput methods, such as sequencing by synthesis, Sanger sequencing was taken over. Few other methods, which were laborious and captured small subsection of the transcriptome, such as northern blot and reverse transcriptase quantitative PCR (RT-qPCR) were also available before the evolution of current transcriptomics techniques. Serial analysis of gene expression (SAGE) was one of the early transcriptome techniques, which worked based on the Sanger sequencing method by concatenating random transcript fragments. Later on, during early 2000s, the contemporary microarray and RNA-Seq techniques were developed. In late 2000s, a variety of microarrays were produced, which were utilized to measure the expression of known genes of various model organisms. The advancement in scientific technologies allowed the measurement of more genes in a single array. The improvement in the fluorescence detection also increased the measurement accuracy and sensitivity of low abundance transcripts. RNA-Seq refers to the sequencing of cDNAs, and the number of counts of each transcript determines their abundance. In 2004, massively parallel signature sequencing (MPSS) utilized complex hybridization series to sequence 16–20 bp sequences. After 2008, RNA-Seq gained popularity and allowed the measurement of 10^9 transcript sequences.

5.3 MICROARRAY

The microarray technique involves hybridization of mRNA molecule to the DNA template of its origin. Thousands of spotted samples, known as probes (with known

FIGURE 5.1 Schematic of the experimental steps to investigate differential gene expression.

identity), are immobilized on a solid support (microscopic glass slides, silicon chips, or nylon membrane). The spots can be cDNA, DNA, or oligonucleotides, which are used to determine the complementary binding of the unknown sequences leading to analysis of numerous gene expressions. An orderly arrangement of the probes on the array is essential since the spots on the array are used for the identification of the genes.[2]

The experimental design includes the following four steps (Figure 5.1):

1. *Sample preparation and labeling*: The first step is the extraction of RNA from the tissues of interest using either column or using solvents such as phenol-chloroform.

   Labeling step depends on the technology being used. Affymetrix uses biotin-labeled complementary RNA for hybridizing, while in the past radioactive labels were used. Many laboratories now use fluorescent labels, with two dyes Cy3 (excited by a green laser) and Cy5 (excited by a red laser). In the majority of the experiments the two samples are hybridized to the arrays and each sample is labeled with one dye, which allows measuring the gene expression in both the samples simultaneously.
2. *Hybridization*: In this step the probes on the solid support and the labeled target (RNA) form heteroduplexes through Watson–Crick base pairing. Hybridization can be achieved either manually or using robotic technology. In the manual

hybridization procedure, the hybridization solution containing the target is injected onto the array kept under the cover slip in a hybridization chamber. The chamber is incubated between 45°C and 65°C for 12 to 24 hours. In the robotic procedure, hybridization is carried out robotically using a hybridization station. In robotic hybridization, variability between the hybridizations and operators were reduced along with controlled hybridization setup.

3. *Washing*: Hybridization is followed by a washing process. The aim of the washing step is that only DNA complementary to each will remain bound to the probe on the array. Washing can be done either using a low-salt wash or with a high temperature wash. The step reduces cross hybridization reactions and removes excessive hybridization solution. A majority of the automated hybridization stations include washing as an automated process.
4. *Image acquisition*: The final step is to get an image of the surface of the hybridized microarray. The slide containing the heteroduplexed DNA molecules is scanned by a scanner. The laser causes the hybrid bonds to glow. The camera records the images and the computer attached to it allows us to view the results and store the data.
5. *Image processing*: The image generated by the scanner is the raw image generated after the experiment. The image processing software converts the images into numerical information, which quantifies the gene expression. The feature extraction step involves identifying the positions of the spots. The pixels in each spot on the image are identified. Similarly, the pixels contributing to the background are also identified. The intensity of the feature, background, and quality control information are calculated. The feature extraction software computes the numerical measurements of gene expression from the image.

5.3.1 Data Analysis

Microarray studies can be divided into three classes based on their aims, which includes class comparison, class discovery, and class prediction. The groups involved in the study can represent different biological states, which may be disease states, treatment groups, or histological subtypes. The steps involved in the conversion of complex microarray data into a easily understandable biological data are summarized in Figure 5.2. The major steps include identification of significantly regulated genes, identification of global patterns of gene expression, and determination of the biological significance of the mined genes and group of genes. The experiments with adequate numbers of replicates will produce lists of differentially expressed genes, which are identified by appropriate statistical tests. Similarly unsupervised clustering methods can identify global patterns among the differentially expressed genes. The biological significance of the list of differentially regulated genes is analyzed individually using gene annotations.[3]

5.3.1.1 Identification of Differentially Expressed Genes

Differentially expressed genes are those whose expression levels differ significantly under different conditions. The fold change determines the difference between the

FIGURE 5.2 Steps involved in microarray data analysis.

groups and their biological significance. Various univariate statistical analyses such as t-test, two-sample t-tests, F-statistic, and modified t-test (SAM) are available to determine the relative expression of genes from normalized microarray data. For multiple classes analysis of variance (ANOVA) is used.[4] Various software packages, such as Bioconductor packages, implemented in R are available to identify the differentially expressed genes. A list of programs available for differential gene expression analysis is summarized in Table 5.1.

5.3.1.2 Clustering Gene Expression Data

Clustering is the process of grouping objects into a set of disjoint classes, namely clusters. The objects within a class are highly similar with each other and the objects in other classes are more dissimilar. Clustering is the first step of gene expression matrix analysis. One of the major characteristics of gene expression data is that it can be clustered to both genes and samples. The coexpressed genes can be grouped

TABLE 5.1
List of Programs Available for Differential Gene Expression Analysis

Program	Website
SAM	http://www-stat.stanford.edu/~tibs/SAM/
MeV	http://www.tm4.org/mev/
iArray	http://zhoulab.usc.edu/iArrayAnalyzer.htm
EDGE	http://www.genomine.org/edge/
Cyber-T	http://cybert.microarray.ics.uci.edu/

into clusters based on the expression patterns. In gene-based clustering, the genes are considered as objects and the samples are considered as features. In addition, the samples can be partitioned into homogenous groups. The groups correspond to particular macroscopic phenotypes, which may include clinical syndromes or disease types. On the other hand, sample-based clustering considers samples as objects and genes as features. Clustering algorithms such as K-means, Self-organizing map (SOM), hierarchical clustering, graph-based methods such as Cluster Identification via Connectivity Kernels (CLICK), Corrupted clique graph (CAST), and model-based clustering approaches are used to group genes and partition samples.[5] Other clustering algorithms are summarized in Table 5.2.

5.3.1.3 Classification

Supervised learning (viz. classification, discriminant analysis, and class prediction) considers predefined classes and the task is to determine the basis of the classification from the set of labeled objects. The information is used to build a classifier, which can be used to predict the class of unlabeled observations. Normalized gene expression data is used as the input for building the classification rule. Various algorithms such as k Nearest Neighbours (kNN), Artificial Neural network (ANN), random forest, Support vector machines (SVM), and weighted voting algorithms were used.[6] The machine-learning application tools used for classification are summarized in Table 5.3.

TABLE 5.2
List of Programs Available for Cluster Analysis

Program	Website
Cluster and Treeview	http://rana.lbl.gov/EisenSoftware.htm
dChip	http://biosun1.harvard.edu/complab/dchip/
MeV	http://www.tm4.org/mev/
MAGIC Tools	http://www.bio.davidson.edu/projects/magic/magic.html
CAGED	http://www.genomethods.org/caged

TABLE 5.3
List of Programs for Classification

Program	Website
Weka	http://www.cs.waikato.ac.nz/ml/weka/
SAS	http://www.sas.com/technologies/analytics/datamining/miner/
IBM/SPSS	http://www.spss.com/software/modeling/modeler-pro/
SVMlight	http://svmlight.joachims.org/
LIBSVM	http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/

5.3.1.4 Biological Significance

Once the differential genes are mined, the biological significance of the list of genes can be determined. Various databases and tools offer broad biological themes for the list of differential genes. This tools aids in the identification of particular genes of interest from a list of potential targets. A list of commonly used gene annotation resources is summarized in Table 5.4.

TABLE 5.4
List of Tools Available for Functional Annotation

Program	Website
oPOSSUM	http://www.cisreg.ca/oPOSSUM/
MATCH	http://www.gene-regulation.com/pub/programs.html#match
ConTra	http://bioit.dmbr.ugent.be/ConTra/index.php
Whole genome rVISTA	http://genome.lbl.gov/vista/index.shtml
TFSCAN	http://mobyle.pasteur.fr/cgi-bin/portal.py?form=tfscan
TFSEARCH	http://www.cbrc.jp/research/db/TFSEARCH.html
TransFind	http://transfind.sys-bio.net/
GenMAPP2	www.genmapp.org
ArrayXPath	http://www.snubi.org/software/ArrayXPath/
GO-cluster	http://www.mpibpc.mpg.de/go-cluster/
GO-view	http://db.math.macalester.edu/goproject
Onto-express	http://vortex.cs.wayne.edu/Projects.html
Gominer	http://discover.nci.nih.gov/gominer/
PathExpress	http://bioinfoserver.rsbs.anu.edu.au/utils/PathExpress/
Ingenuity pathway analysis	http://www.ingenuity.com/
Pathway studio	http://www.ariadnegenomics.com/products/pathway-studio/
KOBAS	http://kobas.cbi.pku.edu.cn
GSEA	http://www.broadinstitute.org/gsea/
DAVID	https://david.ncifcrf.gov/
Gene ontology consortium	http://geneontology.org
OBBO	http://www.obofoundry.org, http://obofoundry.github.io
Reactome	http://www.reactome.org/
KEGG pathway database	http://www.genome.jp/kegg/pathway.html

5.3.2 Quality Control

Reasons for quality variation of the experiments is due to the contaminations on the chip, nonuniform dispersal of fluids, or problems during experiment handling. The combined experimental and bioinformatics procedures require rigorous quality control measures to be monitored at each and every step to get accurate results. Collections of standard set of protocols for microarray experiments are available at Brown Lab, Stanford and The Institute of Genomics Research (TIGR).

5.3.2.1 Monitoring RNA and Array Quality

Before running the microarray experiments, the quality of the RNA extracted from the experimental samples should be checked. Agarose gel electrophoresis with ethidium bromide can be used to evaluate the quality of the samples.

Similarly, the quality of the array should be checked before proceeding with the experiment. The spots on the array are checked for the uniformity of the oligonucleotide products coated on them. The slides are scanned to verify the spiked products, which are later eliminated by a washing procedure before proceeding to the hybridization step.

1. *Signal saturation metric*: The fluorescent signal saturation is one of the major concerns affecting the spot intensity measurement. The microarray quantification software packages calculate the signal saturation. A feature is identified as saturated if 50% of the pixels in the spot are saturated. The best way is to eliminate the frequently saturated gene measurements.
2. *Signal intensity*: Signal intensity is the average raw signal intensity from both channels independently. The value is highly dependent on the labeling technique. Low signal intensity values denote problems during labeling, poor DNA quality, and inappropriate amounts of the sample being used.
3. *Signal intensity ratio*: The value is the ratio between the red and green channel intensities, which is calculated from the processed red and green signals. The value should be close to 1. The deviation from the value indicates problem with the normalization procedure.
4. *Signal-to-noise ratio*: The signal-to-noise ratio is calculated by dividing signal intensity by the background noise. The ratio denotes how well the spots can be detected above the background intensity. The signal-to-noise ranges between 100 and 30, and less than 30 indicates poor reliability.

Other quality control measures include investigating the negative controls, nonuniform features, and background noise.

5.3.3 Tools and Databases

Various tools and databases are available, which are used to store and analyze microarray data. Some of the tools and databases used for the analysis are summarized in the following Table 5.5.

TABLE 5.5
List of Databases and Tools Available for Microarray Data Analysis

Gene expression database	ArrayExpress, Gene Expression Omnibus (GEO), Genevestigator, L2L Microarray Database (L2L MDB), 4DXpress, Stanford Tissue Microarray Database, Prostate Expression Database, Oncomine
General programing tool	Biobase, tkWidgets, BASE, Chipster, EzArray, GeneX, MARS, CarmaWeb, TM4, ArrayTrack, dChio, EMMA2, FGDP, Gecko, GEPAT, MiMiR, EMAAS, SBEAMS, WebArray, 2HAPI, 4DXpress, ArrayPipe, ArrayPlex, ArrayQuest, Asterias, Bioconductor, Biosphere, CARPET, CIBEX, Celsius, ChIPOTle, CoCo, Expression Profiller, EzArray, GECKO, GEISHA, GEPAS, GEPAT, GeneCruiser, GenePattern, GenePublisher, GeneX, Genevestigator, Genopolis, J-Express, KMD, M-CHIPS, MAGMA, MIDAW, NTAP, PEPR, RACE, RAD, SBEAMS, TAMEE, Webarray, arrayMagic, dChip, lumi

5.3.4 Applications in Health Care

Microarrays are efficiently used in clinical diagnosis, which ranges from identification disease-relevant genes to diagnosis of the diseases using its biomarkers. The microarrays are employed for genotyping and determination of disease-causing genes, mutation analysis, screening of single nucleotide polymorphisms (SNP), detection of chromosomal abnormalities, and global determination of posttranslational modifications (Figure 5.2).

The global analysis of the gene expression aids in the identification of biomarkers involved in the disease. The identification of the biomarkers can be achieved by comparing the differential gene expression of the genes from healthy and diseased samples. By analyzing the genes associated with the different stages of the disease, or condition from the clinical samples, we can identify genes expressed in the particular stage of the disease.

Rus et al. used cDNA microarray enclosing samples from 21 different SLE patients and 12 controls to identify candidate molecular markers of the disease. The expression pattern of 375 potential genes was analyzed using the peripheral blood mononuclear cells (PBMC). 50 genes were observed to possess 2.5 fold differences in their expression level. The study found that the majority of the genes previously not associated with SLE belonged to various families such as the IL-1 cytokine family, TNF receptor family, IL-8, and its receptor family. The same group has investigated the gene expression profiling in PBMC from 12 and 14 patients with active and inactive disease, respectively. 14 upregulated and 15 downregulated genes were found to be distinct between the active and inactive state of the disease. Most of the genes belonged to the protease family, adhesion molecules, and neurotrophic factors.[7]

Adib et al. compared the differential gene expression profiles between 4 normal ovarian samples, and 6 pairs of primary and omental serous adenocarcinoma. 421 genes and 118 genes were found to have more than two fold and three fold overexpression respectively in primary ovarian cancer compared to normal tissue. The genes were found to be involved in epithelia and cell–cell contacts. The study also identified a distinct biomarker, MGB2, which is significantly overexpressed in primary

and metastatic ovarian cancer compared to normal ovarian tissue. The investigation also identified other potential biomarkers such as HPN, IFI-15K, KLK6, CP, SLP1, and HE4, which have previously proven to be associated with other types of cancer.[8]

Sundarrajan et al. analyzed the gene expression profiles of lesional and nonlesional samples of psoriatic transcriptome. The results indicated around 244 genes were upregulated and 183 genes were downregulated. The genes were found to be involved in intracellular cascades of the immune response, cytoskeleton and cell adhesion, regulation of cell proliferation, and apoptosis.[9] The study also identified a few potential biomarkers not previously studied in psoriasis. In another study, Sundarrajan et al. used supervised analysis utilizing a random forest (RF) algorithm to distinguish the psoriatic samples from the normal samples. The differential gene expression profiles were first clustered based on the coexpression profile using weighted gene coexpression network analysis (WGCNA) to identify the top 50 genes implicated in the disease, which could be used as a predictor for the next supervising analysis. The genes were found to be involved in functions such as cell cycle, cell proliferation, cytoskeleton rearrangement, keratinocyte proliferation, angiogenesis, and immune response, which are impaired in the disease condition. The RF classifier was trained using the expression profiles from two different microarray datasets. The classifier was tested against three different datasets where it was found to have a specificity value ranging between 0.96 and 1.[10]

Golub et al. studied 38 bone marrow samples from acute leukemic patients. The acute leukemia patients were grouped into acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL) categories. The investigators used supervised analysis for class prediction using 50 genes, which were differentially expressed between 11 and 27 AML and ALL samples, respectively. The predictor of 50 genes was utilized to test set of 34 new leukemic samples. 29 of 34 samples were correctly classified. The genes used for the prediction were involved in cell cycle, cell adhesion, transcription, and oncogenes, which were found to be involved in cancer pathogenesis. The second part of the same study used class discovery on the initial 38 leukemic samples to determine the efficiency of the global gene expression analysis in distinguishing AML and ALL. They used SOM to distinguish between the two groups. 24 of the 25 ALL samples clustered together in one group and 10 out of 13 AML samples were clustered in a second class. The results of the class discovery studies indicated that it is possible to discover the diagnostic classes of the cancer when morphological tests were unavailable whereas the biological and clinical information were available.[11]

Various applications of microarray technology have been utilized in the area of cardiomyopathy. Some studies involved comparisons of failing and nonfailing heart, before and after left ventricular assist device (LVAD) implantation, and dilated and hypertrophic cardiomyopathy. The microarray analysis of failing and nonfailing hearts provides therapeutic insights. The gene expression profile comparison identified that the genes involved in fatty acid metabolism were up-regulated and genes involved in glucose metabolism were down-regulated in failing hearts. These results provide an insight into drugs prescribed to shift fatty acid to glucose metabolism. In a supervised analysis of the gene expression of the dataset of the patients with heart failure, the investigators observed changes in the pattern of gene expression at different clinical

stages.[12–13] In the study conducted by Steenman et al, on failing and non failing hearts, a distinct cluster of patients emerged during unsupervised analysis, who require the highest medical urgency while awaiting cardiac transplantation.[14]

Microarray technology has been utilized to investigate the differential gene expression of pathogens. An oligonucleotide microarray targeting the 16S rRNA gene was developed to detect a panel of 40 predominant human intestinal bacterial pathogens in human fecal samples. A ViroChip system containing 1600 unique 70-mer long oligonucleotide probes, covering 140 viral genomes, was used to detect severe acute respiratory syndrome viruses. The oligonucleotide microarrays were developed to analyze and identify drug-resistant *M. tuberculosis* strains, and the results were comparable to those of the standard antimicrobial susceptibility tests.[15]

5.4 DEEP SEQUENCING (REF-SEQ)

RNA sequencing (RNA-Seq) is a recently developed transcriptome profiling technique utilizing deep-sequencing technologies. It allows entire transcriptomes to be surveyed at single-base resolution and concurrently profile the gene expression on the genome scale. RNA-Seq determines previously undetected changes occurring in a disease states, in response to therapeutics, under different environmental conditions and across a broad range of other studies. RNA-seq can detect both known and novel features in a single assay, gene fusions, transcript isoforms, single nucleotide variants, and others. The major strength of RNA-Seq lies in the identification of novel features of the transcriptome along with its ability to capture the whole transcriptome, which includes both known and unknown regions.[16]

RNA-Seq utilizes deep-sequencing technologies. The RNA of interest is isolated from the tissue of interest. The RNA is converted into a library of cDNA fragments, which are attached to the adaptors to one or both ends. Each molecule (with or without amplification) is sequenced to obtain short sequences from one end (single-end sequencing), or both ends (pair-end sequencing). Each read has a typical size between 30 and 400 bp, depending on the sequencing technology. The reads are aligned either to the reference transcripts or the reference genome, or assembled from a *de novo* genomic sequence (Figure 5.3). The results can be used to analyze both genomic and posttranslational mutations and mRNA expression levels of the genes. The sequencing platforms include Illumina IG, Applied Biosystems SOLiD, Roche 454 Life Science, or Helicos Biosciences tSMS. There are three main sequencing technologies including pyro-sequencing developed by 454 Life Sciences, reversible dye-terminator-based technology by Illumuna, and SOLiD technology by Applied Biosystems.[17]

5.4.1 Data Processing

5.4.1.1 Quality Assessment

Since RNA-Seq involves numerous processes, which range from sample preparation to sequencing, it is not a straightforward method to identify and quantify the RNA species from the reads sequenced. The quality assessment requires to filter data, remove the low-quality sequences, contaminators, adaptors, and overrepresented

FIGURE 5.3 Schematic of RNA-seq experimental steps.

sequences. Read quality decreases towards 3′ end of the reads and as it becomes too low, the bases should be removed to improve the quality and mappability.

5.4.1.2 Read Mapping

After preprocessing of the sequence, the following step is to map the short reads to the reference genome or assemble the reads onto the contigs and align them to the reference genome. If the reads accumulate at the 3′ end of the transcripts in Poly (A) selected samples, it indicates that the RNA quality is low. Many bioinformatics program such as ELAND, SOAP, Bowtie, MAQ, BWA, ZOOM, STAR, and others are suitable to map reads that are not located at the poly (A) tails and exon-intron splicing junctions. Tools such as BLAT, TopHat, MapSplice, and GEM align the reads located in the exon-intron boundaries to help to determine the alternative splicing patterns. The alignment of sequence reads to multiple locations of the genome may occur due to polymorphisms.

5.4.1.3 Read Quantification and Counting

Once the initial quality control steps are completed, the reads should be checked for GC content and gene length bias. With the availability of well-annotated reference transcriptome, we can analyze the genetic constitution of the sample of our study.

Read counting is an essential step in RNA-Seq workflow. The number of RNA-Seq reads generated from a transcript is directly proportional to the transcript's relative abundance in the sample. In case of alternative splicing, the reads mapping to this region will map to constitutive or shared exons, complicating the process of read counting.

5.4.1.4 Transcript Identification and Alignment

When the reference genome is accessible, the RNA-Seq analysis maps the reads to the reference genome or transcriptome and determines which transcripts are expressed. The alignment of reads to the reference genome can be done either mapping to the genome or to the transcriptome. Regardless of the mapping reference, the reads may align to either one position or to multiple positions (multireads). It is difficult to align short reads since they span across several splice junctions. Varieties of tools such as GRIT can annotate the isoforms correctly.

5.4.1.5 Transcript Quantification

The most common application of RNA-Seq is the estimation of gene and transcript expression. It depends on the number of reads that map to each of the transcript. Numerous statistical methods are used to quantify transcript abundance based on the read coverage. The RPKM (reads per kilo base per million mapped reads) is a widely used method to account for expression and normalized read counts with respect to overall mapped read number and gene length. Other than read coverage, sequence depth, gene length, and isoform abundance also determine the transcript abundance. Many algorithms such as RSEM, eXpress, Sailfish, kallisto, and others are developed to estimate the transcript-level expression. Once the read counts are estimated, data normalization is a crucial step in the data processing since it is important to determine accurate results of the gene expression and further analysis. Multiple parameters such as transcript size, sequencing depth and error rate, GC-content, and many more should be considered while choosing normalization methods. For example quantile normalization can improve the quality of the mRNA-Seq data. EDASeq, an R package using within and between normalization methods can decrease the GC-content.

Most algorithms utilize mathematical functions such as probability distributions (e.g., Poisson distribution) followed by Fisher's exact test. The list of differentially expressed genes can be used to gain biological insights into the experiment. Various advanced analyses such as gene ontology, gene sets, network interference, and knowledge databases can be used to understand the biological context of the differentially expressed genes.

5.4.2 Quality Control

The quality of the RNA-Seq can be assessed using both low-level metrics (including read quality, duplication rate, GC content, nucleotide composition, and so on) and high-level metrics (including mapping statistics, coverage uniformity, saturation of the sequence depth, ribosomal RNA contamination, reproducibility of the replicates, read distribution, and strand specificity).

5.4.2.1 Low-Level Metrics

The raw sequence-based metrics checks the experiments at low-level as prior sequence alignments are not required. The raw sequence quality is assessed based on the Phred quality score (Q). Phred score measures the base-calling reliability from Sanger sequencing chromatograms. It is defined as $Q = -10 \times \log10(P)$, where P is the probability of erroneous base calling. GC content gives the percentage of either guanine or cytosine bases in a DNA sequence. It is a simple way to measure the nucleotide composition. Read duplication is often determined by read length, transcript abundance, PCR amplification, and sequence depth.

5.4.2.2 High-Level Metrics

Mapping statistics is an easy intuitive way to assess the success of the RNA sequencing. This includes the number of reads aligned to the reference genome, number of reads aligned to the unique location in the reference genome, number of reads mapped to mitochondria, and number of splice mapped reads. Assessing rRNA contamination in the sequencing is simple, and is accomplished by aligning the reads to the reference genome and counting how many reads are mapped to ribosome genes, or by aligning reads directly to the ribosomal RNA sequences. The saturation test is done to determine whether the current sequence depth is ideal to satisfy the purpose of the experiment. When the sequence is unsaturated, the RPKM will be unstable and the presence of low abundant isoforms will remain unknown.

5.4.3 Tools and Databases

The next generation sequence analysis largely depends on the bioinformatics tools to support the analysis process. Some of the tools and databases, which are commonly utilized for the processing and analysis are summarized in Table 5.6.

TABLE 5.6
List of Tools and Packages Used for Data Processing

Function	Tools
Preprocessing and quality assessment	FastQC, HTQC, Trimmomaatic, BBMap, FLASH, RSeQc
Mapping	ELAND, SOAP, SOAP2, MAQ, Bowtie, BWA, ZOOM, STAR, BLAT, HTSeq, Easy RNASeq, GenomicRanges, Feature-Counts
Expression quantification	Alexa-seq, Cufflinks, RSEM
Differential expression	Cuffdiff, DESeq, DESeq2, EdgeR, PoissonSeq, Limma-voom, MISO
Alternative splicing	TopHat, MapSplice, SpliceMap, SplitSeek, GEMmapper, SpliceR, Splicing-Compass, GliMMPS, MATS, rMATS
Variants detection	GATK, ANNOVAR, SNPiR, SNiPlay3
Pathway analysis	GSEA, GSVA, SeqGSEA, GAGE, SPIA, TAPPA, DEAP, GSAASeqSP
Coexpression network	GSCA, DICER, WGCNA

5.4.4 Applications in Health Care

RNA-Seq technology can be applied for various prognostic, diagnostic, and therapeutic needs implicated in various disease conditions. The multifaceted applications of RNA-Seq include identification of gene fusions, differential expression of the disease causing transcripts, and detection of diversity of RNA species.

RNA-Seq is currently being utilized to assess the expression profiles of genes involved in a number of complex disease such as metabolic disorders, cardiomyopathy, multiple sclerosis, and so on. Leti et al. analyzed the involvement of microRNAs in the regulation of metabolic disorder in patients suffering with nonalcoholic fatty liver disease (NAFLD). Around 30 and 45 miRNAs were found to be upregulated and downregulated, respectively. The genes were implicated in various significant pathways such as apoptosis signaling and growth factor beta signaling.[18]

Punt et al. identified biomarkers expressed in both cancer and immune cells involved in cervical cancer. The authors utilized samples from squamous cell cervical cancer. The RNA-Seq and differential gene expression analysis revealed T-cell leukemia/lymphoma 1A (*TCL1A*) as a novel immune cell expressed biomarker for predicting the survival in patients with cervical cancer.[19]

Ren et al. investigated the complex landscape of genomic alternations in prostate cancer by analyzing the transcriptomes of 14 pairs of prostate cancer and normal tissue. The RNA-Seq analysis revealed the involvement of various gene fusions, alternative splicing, expression of long noncoding RNAs and genes, and somatic mutations in the disease condition.[20]

Amyotrophic lateral sclerosis (ALS) is associated with degeneration of the upper and lower motor neurons in the brainstem and spinal cord. Brohwan et al. determined the major contributing factor by comparing the RNA-sequencing reads from the postmortem spinal section tissue samples from ALS and healthy individuals. Many differentially expressed genes were identified and the tumor necrosis factor (TNF) was identified as an upstream regulator involved in the disease and inflammatory processes, and the TNF signaling cascade. The investigators identified TNFAIP2 as a hub gene, which encodes intracellular proteins of the tumor necrosis factor family.[21]

Li et al. utilized transcriptome analysis to gain insights about the psoriasis disease mechanism. The investigators used 92 psoriatic and 82 normal punch biopsy samples to sequence polyadenylated RNA-derived complementary DNAs. The results identified differentially expressed genes were mostly enriched in immune mediated cascades. The weighted gene coexpression network analysis (WGCNA) revealed multiple modules of coordinately expressed epidermal differentiation genes, overlapping significantly with genes regulated by the long noncoding RNA. They also noted that the genes expressed in the dermal regions were significantly downregulated in psoriatic biopsies.[22]

5.5 FUTURE PROSPECTS

The high-throughput technologies provide a valuable pathway to study the transcripts of organisms. The technologies provide more insight about the genes expressed in different disease condition, disease progression, RNA editing, fusion transcripts,

alternative splicing, and allele-specific expression. Transcriptomics research is moving forward towards single-cell transcriptomics studies, since the expression of the genes vary from one cell to other. Mining data from public repositories such as ENCODE, TCGA, GEO, and the Geuvadis project may provide new insight about gene regulation. The high volume of data requires high-end computational resources for transcriptome assembly and other downstream analysis. Hence, highly parallel data processing steps are required for better analysis and faster processing of the results.

REFERENCES

1. R.M. Fryer, J. Randall, T. Yoshida et al., Global analysis of gene expression: Methods, interpretation, and pitfalls, *Exp. Nephrol.* 10, 2002, pp. 64–74.
2. G. Barker and D. Stekel, Microarray bioinformatics, *Ann. Bot.* 93, 2004, pp. 615–616.
3. L.D. Miller, P.M. Long, L. Wong, et al., Optimal gene expression analysis by microarrays, *Cancer Cell* 2, 2002, pp. 353–361.
4. N.E. Olson, The microarray data analysis process: From raw data to biological significance, *NeuroRx J. Am. Soc. Exp. Neurother.* 3, 2006, pp. 373–383.
5. L. Kaufman and P.J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley series in probability and mathematical statistics, Wiley, Hoboken, N.J, 2005.
6. Sandrine, D. and F. Jane, Classification in microarray experiments. In *Statistical Analysis of Gene Expression Microarray Data*. CRC press. 2003; 93–159.
7. V. Rus, S.P. Atamas, V. Shustova et al., Expression of cytokine- and chemokine-related genes in peripheral blood mononuclear cells from Lupus Patients by cDNA Array, *Clin. Immunol.* 102, 2002, pp. 283–290.
8. T.R. Adib, S. Henderson, C. Perrett et al., Predicting biomarkers for ovarian cancer using gene-expression microarrays, *Br. J. Cancer* 90, 2004, pp. 686–692.
9. S. Sundarrajan, S. Lulu, and M. Arumugam, Insights into protein interaction networks reveal non-receptor kinases as significant druggable targets for psoriasis, *Gene* 566, 2015, pp. 138–147.
10. S. Sundarrajan and M. Arumugam, Weighted gene co-expression based biomarker discovery for psoriasis detection, *Gene* 593, 2016, pp. 225–234.
11. T.R. Golub, D.K. Slonim, P. Tamayo et al., Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring, *Science* 286, 1999, pp. 531–537.
12. M.M. Kittleson, K.M. Minhas, R.A. Irizarry et al., Gene expression analysis of ischemic and nonischemic cardiomyopathy: Shared and distinct genes in the development of heart failure, *Physiol. Genomics* 21, 2005, pp. 299–307.
13. F.-L. Tan, C.S. Moravec, J. Li et al., The gene expression fingerprint of human heart failure, *Proc. Natl. Acad. Sci. USA*. 99, 2002, pp. 11387–11392.
14. M. Steenman, G. Lamirault, N. Le Meur et al., Distinct molecular portraits of human failing hearts identified by dedicated cDNA microarrays, *Eur. J. Heart Fail.* 7, 2005, pp. 157–165.
15. M.B. Miller and Y.-W. Tang, Basic concepts of microarrays and potential applications in clinical microbiology, *Clin. Microbiol. Rev.* 22, 2009, pp. 611–633.
16. U. Nagalakshmi, Z. Wang, K. Waern et al., The transcriptional landscape of the yeast genome defined by RNA sequencing, *Science* 320, 2008, pp. 1344–1349.
17. Z. Wang, M. Gerstein, and M. Snyder, RNA-Seq: A revolutionary tool for transcriptomics, *Nat. Rev. Genet.* 10, 2009, pp. 57–63.

18. F. Leti, I. Malenica, M. Doshi et al., High-throughput sequencing reveals altered expression of hepatic miRNAs in non-alcoholic fatty liver disease-related fibrosis, *Transl. Res. J. Lab. Clin. Med.* 166, 2015, pp. 304–314.
19. S. Punt, W.E. Corver, S.A.J. Van der Zeeuw et al., Whole-transcriptome analysis of flow-sorted cervical cancer samples reveals that B cell expressed TCL1A is correlated with improved survival, *Oncotarget* 6, 2015, pp. 38681–38694.
20. S. Ren, Z. Peng, J.-H. Mao et al., RNA-seq analysis of prostate cancer in the Chinese population identifies recurrent gene fusions, cancer-associated long noncoding RNAs and aberrant alternative splicings, *Cell Res.* 22, 2012, pp. 806–821.
21. D.G. Brohawn, L.C. O'Brien, and J.P. Bennett, RNAseq analyses identify tumor necrosis factor-mediated inflammation as a major abnormality in ALS spinal cord, *PloS One* 11, 2016, pp. e0160520.
22. B. Li, L.C. Tsoi, W.R. Swindell et al., Transcriptome analysis of psoriasis in a large case-control sample: RNA-seq provides insights into disease mechanisms, *J. Invest. Dermatol.* 134, 2014, pp. 1828–1838.

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

6 Comparative Study of Predictive Models in Microbial-Induced Corrosion

Nitu Joseph and Debayan Mandal

CONTENTS

6.1 Introduction 92
6.2 Background 92
6.3 Broadly Classified Phases 93
6.3.1 Substrate Organism Interaction 93
6.3.1.1 Sulfate Reducing Bacteria (SRB) 93
6.3.1.2 Metal Reducing Bacteria (MRB) 93
6.3.1.3 Metal Depositing Bacteria (MDB) 93
6.3.1.4 Acid Producing Bacteria (APB) 94
6.3.1.5 Mycetes (Fungi) 94
6.3.2 Biofilm Formation 94
6.3.3 Biofilm Induced Corrosion 94
6.3.4 Nature of Corrosion 96
6.3.4.1 Pitting Corrosion 96
6.3.4.2 Flask Shaped Cavitation 96
6.3.4.3 Uniform Layer Corrosion 96
6.3.4.4 Etch Line Corrosion 96
6.3.5 Kinetics Prediction Models 97
6.3.5.1 Michaelis–Menten Model 98
6.3.6 Formulating Biofilm Formation 99
6.3.6.1 Adhomian Decomposition 99
6.3.7 Decomposition Rate Evaluation 101
6.3.7.1 Laplace Transformation of Plank–Nerst Equation 101
6.3.7.2 Butler–Volmer Equation Modification 104
6.3.8 Relevant Data Handling Techniques 107
6.3.8.1 Classification and Regression Trees (CART) 107
6.3.8.2 K-Nearest Neighbor Method (KNN) 108
6.3.8.3 Clustering 108
6.3.8.4 Artificial Neural Networks 109
6.3.8.5 Support Vector Machines (SVN) 109

6.3.9 Optimal Combination 109
6.4 Conclusion 110
References 110

6.1 INTRODUCTION

Iron is such a metal, whose origin itself is shrouded in mysteries. While some studies suggest that the earliest iron artifacts have their origin from meteoric iron deposits, others emphasize the existence of iron smelting processes as early as 5000 BCE. The oldest artifacts have been discovered from the cradles of ancient civilizations, like Mesopotamia and Egypt. But the iron wares were not as widely used as their silver and gold counterparts, owing to the single major problem of "corrosion." With the onset of Iron Age around 1200 BCE–1300 BCE, as the metal became more freely available and abundant, man began to venture into the inner workings of this troublesome menace called corrosion, along with ways to try and prevent it.

The phenomenon of anaerobic iron corrosion has mind-boggled scientists for centuries. The popular notion of metal corrosion occurring due to the interaction of the metal with air and moisture could not explain the mechanisms driving the corrosion of buried pipelines, pasteurization tanks, along with a variety of structures of mechanical, infrastructural, as well as cultural significance. It was not until the early 1900s scientists started looking at the gravity of the corrosion problem (Olmstead and Hamlin 1900). Scientists had always viewed the phenomenon with skepticism, as in most cases if an abiotic mechanism could be related to a particular case of corrosion, it was explained to have been caused by microbes. It was not until the latter half of the twentieth century that the frame of focus was shifted on to the persistent presence of microbial colonies, which were causing severe monetary losses.

6.2 BACKGROUND

The progressive deterioration of a substrate after being subjected to redox reaction, facilitated by biofilms formed by the action of colonizing bacteria, is known as MIC, alternatively known as biocorrosion (Xu et al., 2016). The conventional electrochemical corrosion involves the release of metal cations as a resultant of electron transfer from a zero-valent metal to an external electron acceptor. However, biocorrosion is synergistic in nature, involving the interactions between the metal substrate, abiotic corrosion products, microbial cells, and the EPS secreted by them.

With present-day machineries and infrastructure being heavily dependent on metals and related products, MIC poses an important challenge. Across all industrial sectors, 20%–30% of corrosion in metal structures could be attributed to MIC (Koch et al., 2010), amounting to 2%–3% of the GDP of developing nations (Koch et al., 2001). Its effect is more prominent in the energy industry, as the corrosion in the oil and gas pipelines cause a loss in the range of hundreds of billion dollars (Zhiyuan and Zongdi 2017).

This chapter attempts to analyze the complex links between bacteria and the corrosion process they induce. Various mathematical modeling methods used for studying MIC have been analyzed, and a set of feasible combinations are presented for better understanding.

6.3 BROADLY CLASSIFIED PHASES

6.3.1 Substrate Organism Interaction

From an electrochemical perspective, corrosion can be viewed as an interfacial process. The multiple parameters of the physicochemical environment are crucial in determining reaction kinetics. Parameters like pH, salt content, redox potential, etc. can be altered by the presence of a microbial colony at the interface (Wells and Melchers 2011).

This alters the otherwise purely abiotic nature of the redox reaction involving iron and suitable oxidants. In environments with little or no oxygen and a pH > 6, the normal chances of abiotic corrosion are abysmally low (Wells and Melchers 2011). But it was observed these areas exhibit a rather high corrosion rate (Gu 2014).

The basic requisite for bacterial proliferation is water. The presence of water along with an oxidized electron donor, a reduced electron acceptor, an energy, and a carbon source facilitates the accelerated growth of microbial consortia. A comprehensive broad-spectrum analysis on the genomics of bacterial colonies has revealed the existence of a multitude of anaerobic bacterial species instrumental in the corrosion process (Marangoni et al., 2013). Obligate anaerobic bacteria involved in sulfate respiration are one of the primary organisms responsible for the MIC. They are popularly known as sulfate reducing bacteria (SRB), metal reducing bacteria (MRB), metal depositing bacteria (MDB), and acid producing bacteria (APB) (Beech and Gaylarde 1999).

6.3.1.1 Sulfate Reducing Bacteria (SRB)

The SRB family includes bacteria like *Desulfovibrio desulficans, Desulfomaculum ruminis, Desulfonema limicola,* and facultative anaerobic bacteria like *Desulfuromonas acetoxidans* (Little et al., 2000). Biogenic sulfur compounds thus formed also pose serious health risks. From the early 1930s, investigations had begun on the pitting corrosion caused in cast iron, as SRB consortia were traced as the causative reason (Beech et al., 2000).

6.3.1.2 Metal Reducing Bacteria (MRB)

Naturally, iron objects form a protective passive layer of oxide on their surfaces. The bacterial consortium involving *Alteromonas putrefaciens, Shewanella putrefaciens,* and *Pseudomonas* genera are key causative species (Maluckov 2012). Organisms belonging to this category are predominantly involved in metabolic activities resulting in the reduction of iron and manganese metals (Beech et al., 2000).

6.3.1.3 Metal Depositing Bacteria (MDB)

These bacterial species are involved in the process of biologically metabolizing various metal oxides, commonly iron and manganese. The divalent ferrous ions (Fe^{2+}) present in the bulk-dissolved state are converted into a metabolized precipitate chiefly composed of trivalent ferric ions (Fe^{3+}). Bacterial species like *Gallionella*, *Leptothrix,* and *Geobacter sp* are involved in this process (Beech et al., 2000).

6.3.1.4 Acid Producing Bacteria (APB)

Inorganic acids like sulfuric acid, sulfurous acid, nitric acid, and carbonic acids are produced as metabolic wastes by some bacteria (Beech et al., 2000). Bacterial species like *Thiobacillus sp*, *Thiothrix,* and *Beggiatoa spp* are involved in the production of sulfurous acid while those like nitrogen oxidizing bacteria and ammonia oxidizing bacteria are involved in the production of nitric acid resulting corrosion of concrete and natural stone. Consequently, with the fluctuation in the alkalinity, there arises an increase in lead contamination and copper solubility.

6.3.1.5 Mycetes (Fungi)

The most widely occurring form of corrosion induced by the fungi is that of aluminum. There have been extensive studies on the corrosion of aircraft fuel tanks due to jet fuel contamination. The biocontaminant was identified as a consortium of *Hormoconis resinae, Aspergillus spp., Fusaruium spp.,* and *Penicillium spp.* The *Hormoconis resinae* biometabolized the hydrocarbons into organic acids. These metabolites form chelation compounds with copper, iron, aluminium, and zinc, thus forming weak spots in the fabric of the aircraft metal lining (Beech et al., 2000).

6.3.2 Biofilm Formation

Matrix layers are formed when bacterial species colonize a particular substrate and start the secretion of Extra cellular Polymeric Substances (EPS) (Beech and Sunner 2004), leading to the onset of corrosion (Alabbas et al. 2013). The EPS, along with the microbial colonies, constitute the biofilm commonly called "slime layer." The EPS layer are commonly composed of minor concentrations of lipids and nucleic acids engulfed in ploysaccharides and protiens. Formed as a monolayer in the initial stages, these have a potential of increasing their thickness up to several centimeters and tend to be highly hydrated (~97% water). One of the primary characteristics of biofilm is the exponential rate of bacterial cell growth exhibited; with the cell count in the colonies reaching as high as Log_{10} (6.0) cells/cm^2 (Bouman et al., 1982) thereby establishing synergistic communities commonly known as "Microconsortia." Another prominent characteristic is the free motility of the bacteria within the EPS layer, resulting in pronounced pH gradient across the matrix (Figure 6.1) (Gu 2012).

6.3.3 Biofilm Induced Corrosion

The primary mechanism of MIC involves the release of the dehydrogenase enzyme, which results in the formation of cathodic hydrogen as a byproduct at the metal surface. This further aids the growth of the SRB consortia. The bacterial reduction of sulfate is carried out by the hydrogenase positive (H+) SRB. This results in a cathodic depolarization process, as the hydrogen are scavenged away by the bacteria, thus leading to anodic corrosion (Figure 6.2).

This process culminates in the bacterial release of hydrogen sulfide, which in turn forms metallic sulfide (ferrous sulfide) as an end product (Videla 2010). Subsequently, a galvanic cell is formed between the metallic substrate and its sulfide promoting further corrosion. Here the metal sulfide acts as a cathode. Therefore, this implies

FIGURE 6.1 MIC in progress under the biofilm.

that the rate of corrosion is directly proportional to the concentration of metal sulfide present in the vicinity. The metal sulfide and hydroxide are products of the corrosion process. The rate of anodic reaction is accelerated by the adsorbed hydrogen-driven cathodic reaction (Loosdrecht et al., 2002).

The precipitated metal sulfide has a depolarizing effect, thus resulting in production of hydrogen. The thiosulfate and polythionate metabolites also aggravate the corrosion problem (Figure 6.3).

The process of biofilm formation involves the following steps:

a. Planktonic colonization of the vicinity of an exposed weak point.
b. Development of exopolymer chains anchoring the planktonic SRB to the reaction site (Sessile state) (Beech et al., 1994).
c. Formation of extensive network of exopolymers and consequent thickening of the biofilm matrix.
d. Initiation of the corrosion process under anoxic conditions of the biofilm.

FIGURE 6.2 Cathodic depolarization process.

FIGURE 6.3 Stages of biofilm formation.

6.3.4 Nature of Corrosion

6.3.4.1 Pitting Corrosion

This form of corrosion is the resultant of passive breakdown of a metal caused by the metabolites of SRB forming multiple pits of varying sizes on the metal surface, as shown in the Figure 6.4 (Amadi et al. 2007). For example, stainless steel pipeline corrosion (Liang et al., 2016).

6.3.4.2 Flask Shaped Cavitation

This form of corrosion occurs with the SRB entering the hairline cracks formed as a result of the stress rupture of the substrate (Figure 6.5).

6.3.4.3 Uniform Layer Corrosion

This form of corrosion is caused by acid-producing bacteria. The metabolites of the biofilm cause the total erosion of the top layer of the substrate (Figure 6.6) (Wells et al., 2012).

6.3.4.4 Etch Line Corrosion

This form of corrosion is caused by fungal action. The fungal hyphae spread in irregular patterns and leave behind trails of corrosive metabolites corroding away the substrate in an etch line fashion (Figure 6.7).

FIGURE 6.4 Pitting corrosion.

FIGURE 6.5 Flask-shaped cavitation.

FIGURE 6.6 Uniform layer corrosion.

FIGURE 6.7 Etch line corrosion.

6.3.5 Kinetics Prediction Models

As biofilm formation in case of MIC phenomenon is essentially kicked off by the release of enzymes, as explained earlier, enzyme kinetics models are used. Enzyme kinetics encompasses the understanding on how the enzyme functions, formation of the enzyme substrate complex, and the factors effecting enzyme substrate complex. One can classify these mechanisms into:

a. Single-substrate mechanisms.
b. Multiple-substrate mechanisms.

In this case, the enzyme follows a single-substrate mechanism. Therefore, Michaelis–Menten kinetics are applied.

6.3.5.1 Michaelis–Menten Model

The Michaelis–Menten model attempts to describe ideal enzyme kinetics. An enzymatic reaction in equilibrium is given by:

$$E + S \underset{k_r}{\overset{k_f}{\rightleftarrows}} ES \xrightarrow{k_{cat}} E + P$$

where E is the Enzyme, S is the Substrate, ES is the enzyme-substrate complex, P is the final product, k_f is the forward rate constant, k_r is the reversed (backward) rate constant and k_{cat} is the catalytic rate constant. The k_{cat} is also called turnover rate and is defined as the maximum number of molecules an enzyme can produce per second. It is represented by:

$$k_{cat} = \frac{V_{max}}{E_T}$$

The enzyme-substrate complex is a reversible reaction. The rate laws of each reactant from this equation would be:

$$\frac{dE}{dt} = -k_f \cdot E \cdot S + k_r \cdot ES + k_{cat} \cdot ES$$

$$\frac{dS}{dt} = -k_f \cdot E \cdot S + k_r \cdot ES$$

$$\frac{dES}{dt} = k_f \cdot E.S - k_r \cdot ES - k_{cat} \cdot ES$$

$$\frac{dP}{dt} = k_{cat} \cdot ES$$

while solving this equation, the steady state assumption is relied upon, which in an equation form is:

$$E + ES = E_o$$

This assumption states the concentration of the free or bound enzymes is constant. After solving the rate of product formation (P), we obtain:

$$v = \frac{dP}{dt} = \frac{(V_{max}) \cdot S}{k_m + S}$$

This is the Michaelis–Menten kinetics equation. Here k_m represents the substrate concentration when the reaction rate is half of its maximum value. It is given by the following equation:

$$k_m = \frac{k_r + k_{cat}}{k_f}$$

In biofilm reactions, this same equation is used to derive the following formula:

$$D\frac{d^2C}{dz^2} = q\frac{C}{k+C}X_f$$

with boundary conditions of: At $z = 0$, $dC/dz = 0$, and at $z = 1$, $C = C_1$.

The annotations are explained in the Adhomian decomposition method, in the following section.

6.3.6 Formulating Biofilm Formation

Biofilm structure formation is governed by various physical and biological factors. Common factors, which come into interplay here, are growth yield, mass transfer, conversion rates (substrate), detachment, etc. These are further micromanaged by specific strain dependent factors. They come in two forms—porous, heterogenous structures, and compact ones.

Mass transfer is affected by pores only at a local level. So, at lower flow rates and detachment forces, the porous and heterogeneous structures are formed. While at higher flow rates, the sheer increase of shear stress lead to the formation of compact biofilms. Through all these factors, one seeks to assert substrate flux conditions by use of various analytical expressions.

6.3.6.1 Adhomian Decomposition

Muthukaruppan et al. (2013) has used this method for analyzing biofilm for a square law of microbial death rate. Taking substrate consumption, as defined by Michaelis Menten Kinetics, the nonlinear equations thus obtained with boundary values can be both solved by the Adomian decomposition method and other numerical methods, which are prevalently used in programming software. The main advantage of Adomian decomposition—and why it is preferred—is because it procures a fast converging series with repeated iterations. So much so that even the first iteration itself, as shown in various case studies, has a high degree of accuracy.

The nonlinear ODE obtained is as follows:

$$D\frac{d^2C_f}{dz^2} = \frac{r^2M}{d}\left(\frac{C_f}{k+C_f}\right)^2$$

where k is the Michaelis–Menten constant, C_f is the biofilm's substrate concentration, D is biofilm's diffusion coefficient, M being biomass yield per substrate unit, d is Microbial death constant, z is taken as a coordinate value.

Making it dimensionless by including the following parameters, the equation can be solved using Adomian decomposition:

$$C = \frac{C_f}{k};\ x = \frac{z}{l};\ t = \frac{Mr^2l^2}{dkD};\ C_l = \frac{C_1}{k}$$

Where l is biofilm thickness, t is the dimensionless version of biofilm thickness, C_1 is the substrate concentration outside the biofilm. The two equations we obtain on substrate concentration, C, and concentration flux, $\varnothing$, are:

$$\frac{d^2C}{dx^2} = t\left(\frac{C}{1+C}\right)^2$$

$$\varnothing(x) = \frac{1}{\sqrt{t}}\left|\frac{dC}{dx}\right|_{x=1}$$

Now, Adomian decomposition method essentially depends on the nonlinear differential equation, suppose;

$$F(x,\ y(x)) = 0$$

Breaking it into its linear (let L) and nonlinear (let N) parts, one gets:

$$L(y(x)) + N(y(x)) = 0$$

L is taken as an invertible operator and solving for $L(y)$ gives us:

$$L(y) = -N(y)$$

As L is invertible, taking inverse on both sides, we obtain:

$$y = -L(N(y)) + \varphi(x)$$

where $\varphi(x)$ is the constant of integration. This y can now be represented in the form of an infinite series:

$$y = \sum_{n=0}^{\infty}(y_n)$$

Following this, the nonlinear term (N) can be written in terms of Adomian polynomials (A_n):

$$N(y) = \sum_{n=0}^{\infty}(A_n)$$

where $(A_n(x)) = \left|\frac{1}{n!}\frac{d^n}{d\lambda^n}N\sum_{n=0}^{\infty}(\lambda^n y_n)\right|\lambda = 0.$

Substituting these values y can be solved from the previous equation after some truncations in the infinite series.

Coming back to the substrate concentration equation, when solved, the resulting equation is:

$$C(x) = C_l + \frac{t}{2}\left(\frac{C_l}{1+C_l}\right)^2 (x^2-1) + \frac{t^2 C_l^3}{(1+C_l)^5}\left(\frac{x^4}{12} - \frac{x^2}{2} + \frac{5}{12}\right)$$
$$+ \frac{t^2 C_l^2}{4(1+C_l)^6}\left(\frac{x^6}{30} - \frac{x^4}{6} + \frac{x^2}{2}\right) - \frac{t^2 C_l^5}{(1+C_l)^7}\left(\frac{x^6}{30} - \frac{x^4}{6} + \frac{x^2}{2}\right)$$
$$+ \frac{2t^2 C_l^2}{(1+C_l)^7}\left(\frac{x^6}{360} - \frac{x^4}{24} + \frac{5x^2}{24}\right) + \frac{3t^2 C_l^6}{4(1+C_l)^8}\left(\frac{x^6}{30} - \frac{x^4}{6} + \frac{x^2}{2}\right)$$
$$- \frac{2t^2 C_l^5}{(1+C_l)^8}\left(\frac{x^6}{360} - \frac{x^4}{24} + \frac{5x^2}{24}\right) + \frac{t^2 C_l^4}{360(1+C_l)^8}(-155 + 66C_l)$$

The concentration flux equation yields the following solution:

$$\varnothing = \frac{1}{\sqrt{t_l}}\left[t\left(\frac{C_l}{1+C_l}\right)^2 - \frac{2}{3}\frac{t^2 C_l^3}{(1+C_l)^5} + \frac{2}{15}\frac{t^2 C_l^2}{(1+C_l)^6} - \frac{8}{15}\frac{t^2 C_l^5}{(1+C_l)^7}\right.$$
$$\left. + \frac{8}{15}\frac{t^2 C_l^2}{(1+C_l)^7} + \frac{2}{5}\frac{t^2 C_l^6}{(1+C_l)^8} - \frac{8}{15}\frac{t^2 C_l^5}{(1+C_l)^8}\right]$$

These equations give the needed value of concentration flux factor and thereby help in finding the property of biofilm formed through its microbial death rate.

6.3.7 Decomposition Rate Evaluation

The weight loss of a metal due to decomposition, or more accurately in this case—corrosion, is dependent on various factors pertaining to immediate or absolute environments. As the corrosion rate is determined the amount of loss can be quantified. This weight loss is seen to be increasing with increased periods of exposure; more in saltwater environments as opposed to their freshwater counterparts.

Corrosion of metals is heavily influenceby redox reactions, environmental physicochemical properties, oxygen diffusion rate, and microbial growth in addition to the previously mentioned factors. The analytical models for this are solely developed with the purpose of predicting the corrosion rate of a metal exposed to a certain environment.

6.3.7.1 Laplace Transformation of Plank–Nerst Equation

Ukpaka et al. (2011) has developed an analytical model for corrosion rate with respect to oxygen diffusion transport in a water environment. To achieve this, at first the

model is taken to represent the steel's corrosion in the presence of water, shaped by oxygen diffusion, while neglecting the convective motion of oxygen. Therefore, the Plank–Nernst law is taken as:

$$\overrightarrow{\varnothing_k} = -D_r^o \vec{\nabla} C_k + \left(\frac{-z \cdot D_r^o \cdot F}{RT} C_k \right) \vec{\nabla} E$$

where, $\overrightarrow{\varnothing_k}$ is the mass flux, D_r^o is the oxygen diffusivity constant in the rust layer, C_k is the species concentration, F is Faraday constant, R is Ideal Gas constant, T is absolute temperature, and E is the electrolyte's electric potential.

Taking into consideration the migration and diffusion, the mass transport equation becomes

$$\frac{\partial C_k}{\partial t} + \vec{\nabla} \overrightarrow{\varnothing_k} = S_k$$

where, S_k is an oxygen production/consumption source term.

Assuming oxygen consumption occurs only at the metallic surface, due to biochemical reactions; substituting the second equation into the first we get the migration diffusion equation:

$$\frac{\partial C_k}{\partial t} + \frac{\partial}{\partial x} \left(\frac{-z \cdot D_r^o \cdot F}{RT} \frac{\partial E}{\partial x} C - D_r^o \frac{\partial C}{\partial x} \right) = 0$$

Taking only the diffusion equation into account, neglecting the migration part;

$$E = \frac{\partial C}{\partial t} - \frac{\partial}{\partial_X} [\partial C / \partial x] = 0$$

Assuming D_r^o to be dimensionless, and $D_r^o = W(x) \cdot D_*^o$; where D_*^o is independent of position.

So, the previous equation can be written as

$$\frac{\partial C}{\partial t} - D_*^o \frac{\partial}{\partial_X} \left[W(x) \frac{\partial_C}{\partial_X} \right] = 0$$

$$\underset{\text{or}}{\Rightarrow} \frac{\partial_C}{\partial t} - D_*^o \cdot W(x) \frac{\partial^2 C}{\partial_{X^2}} = 0$$

Thus, a second order differential is obtained. So, Laplace transformation and separation of variables can be applied to solve it

$$\frac{\partial_C}{\partial t} = D_*^o \cdot W(x) \frac{\partial^2 C}{\partial_{X^2}}$$

So, $C = TX$; where C is the oxygen concentration, T is the time coefficient, and X is the position coordinate. Differentiating it twice with respect to t and x, two *equations can be obtained*:

$$\frac{\partial_C}{\partial t} = T^1 X$$

$$\frac{\partial^2 C}{\partial_x^2} = TX^{II}$$

Substituting them in the primary equation that was obtained:

$$T^1 X = D_*^o \cdot W(x) TX^{\|}$$

$$\text{or} \quad \frac{T^1 X}{TX} = D_*^o \cdot W(x) \frac{TX^{II}}{TX} = \lambda^2$$

$$\text{or} \quad \frac{T^1}{T} = \lambda^2 \quad \text{and} \quad D_*^o \cdot W(x) \frac{X^{11}}{X} = \lambda^2$$

The first equation becomes a first order differential equation if we resolve further;

$$T^1 - \lambda^2 T = 0$$

Solving it using Laplace, the resulting equation is:

$$T_{(t)} = t_o e^{\lambda^{2f}}$$

The second equation obtained can be written as;

$$D_*^o \cdot W(x) X^{11} = \lambda^2 x$$

Solving through Laplace transform and application of partial law fraction, gives:

$$X_{(t)} = \frac{X_o}{\lambda}(e^{\lambda} - e^{-\lambda})$$

Among the two equations at hand, the second one is used to obtain the biocorrosion reaction's functional parameters. As for the first equation, it can be used to modify the general formula of corrosion rate, which is;

$$c_R = \frac{534W}{DAt}$$

to:

$$C_R = \frac{534W}{DAt_o e^{\lambda^{2f}}}$$

This corrosion rate is now tailored to the factor of oxygen diffusion transport in water, making it easier to apply.

Previously, $C = TX$ was taken. Substituting the resolved equations now

$$C_{(t)} = t_o e^{\lambda^{2f}} \left[\frac{X_o}{\lambda} (e^{\lambda_{--}\lambda}) \right]$$

Putting the constant values yield

$$C_{(t)} = 0.679 \frac{X_o}{\lambda} t_o e^{\lambda^{2f}}$$

This equation gives us the oxygen diffusion transport rate at any given time and space.

6.3.7.2 Butler–Volmer Equation Modification

The corrosion current has a lukewarm relationship with the potential and pH. Usual iron dissolution comprises active, transition, passive, and pitting stages. The biofilm spread is heavily influenced by the amount of dissolved oxygen through differential aeration. Localized corrosion shifts to a broader area just by an increase in the amount of dissolved oxygen. In the presence of saline water, however, if chloride ions are present, a biofilm would slow down the corrosion rate.

Picioreanu and van Loosdrecht (2002) have formulated a mathematical model based on the factor of differential aeration and saline presence; while conjuring out a relation between corrosion rate and corrosion potential. A major positive aspect of this model is that the cathodic and anodic regions need not be defined. However, as there are a lot of passive undercurrents affecting the metal pertaining to the surroundings, sometimes due to these the potential cannot be directly correlated with rate; therefore, the model should be employed with caution.

As mentioned before, the anodic current is affected by its four phases.

$$j_1 = \left[(1-\sigma) j_{1,a} + \sigma j_{1,p}\right](1-\sigma_p) + \left[(1-\sigma_{pit}) j_{1,apit} + \sigma_{pit} j_{1,p}\right] \times \sigma_p \exp\left[\frac{-\alpha_1 F}{RT}(\Phi_M - \Phi - E_{pit})\right]$$

where j_1 is the anodic current density, σ is the shift of regions, Φ is the solution potential, Φ_M is the metal potential, α_1 is the charge transfer coefficient ranging from 0 to 1, F is Faraday's constant, R is ideal gas constant, and T is the absolute temperature.

The Butler–Volmer equation is used in depicting the current density of the active region:

$$j_{1,a} = j_{1,0}\left\{\exp\left[\frac{\alpha_1 F}{RT}(\Phi_M - \Phi - E_{e,1})\right] - \exp\left[\frac{\alpha_1 F}{RT}(\Phi_M - \Phi - E_{e,1})\right]\right\}$$

where $j_{1,0}$ is current exchange density, and the given fraction can be calculated from the Tafel slope. Some of the values can be calculated by the following formulas:

- At local conditions, the equilibrium potential is: $E_{e,1} = E^0_{e,1} + \frac{0.059}{2}\log(C_2)$
- $\sigma = \exp\frac{\left[2\frac{\alpha_1 F}{RT}(\Phi_M - \Phi - E_p)\right]}{1+\exp\left[-2\frac{\alpha_1 F}{RT}(\Phi_M - \Phi - E_p)\right]}$; as proposed by Hines.
- Flade potential is given by: $E_p = E^0_p - 0.059pH$
- $\sigma_p = \exp\frac{\left[4\frac{\alpha_1 F}{RT}(\Phi_M - \Phi - E_{pit})\right]}{1+\exp\left|-4\frac{\alpha_1 F}{RT}(\Phi_M - \Phi - E_{pit})\right|}$.
- $\sigma pit = \exp\frac{\left[2\frac{\alpha_1 F}{RT}(E_{pit} - E_p)\right]}{1+\exp\left|-2\frac{\alpha_1 F}{RT}(E_{pit} - E_p)\right|}$

According to these equations current density above pitting potential experiences a sudden surge:

$$j_{1,apit} = j_{1,0}\left\{\exp\left|\frac{\alpha_1 F}{RT}(E_{pit} - E_{e,1})\right| - \exp\left[\frac{-\alpha_1 F}{RT}(E_{pit} - E_{e,1})\right]\right\}$$

However, the pitting potential linearly decreases

$$E_{pit} = -0.29 + 0.125p$$

Apart, from the anodic reactions, the cathodic reactions too can be given by Butler–Volmer Equations:

$$j_2 = j_{2,0}\left\{\exp\left|\frac{\alpha_2 F}{RT}(\Phi_M - \Phi - E_{e,2})\right| - \exp\left[\frac{-\alpha_2 F}{RT}(\Phi_M - \Phi - E_{e,2})\right]\right\}$$

The $j_{2,0}$ usually depends on molar concentrations and is calculated as

$$i_{2,0}(C_1, C_4) = i_{2,0}^{\infty}(C_1)^{\gamma_1}(C_4)^{\gamma_4}\,[1S]$$

In this case, the equilibrium potential (at 298 K) is given by

$$E_{e,2} = E_{e,2}^0 + \frac{0.059}{4}\log\left(\frac{C_1}{C_1^*}\right) - 0.059\log(C_4)$$

The Tafel slope is similarly used for the fraction part. Apart from these, oxygen intake can be calculated by Monod reactions (Meena et al., 2014). The ferrous ion oxidations are calculated depending on ferrous ion concentrations, pH value and dissolved oxygen. The ferric ion precipitation and ionic product of water, being homogenous equations, which can be taken at equilibrium, can be represented by respective equilibrium constants.

While solving this model, the assumptions taken are:

a. System is taken at steady state. This is pertaining to the fact that chemical reactions occur at a relatively faster rate than biofilm phase changes.
b. Transport terms for chemicals are defined by molecular diffusion and migration in the mass flux equations.
c. Nernst–Einstein equations are used to have a relation between ionic mobility and diffusion coefficient, assuming the solution to be infinitely diluted.
d. System was taken to be isothermal in nature wherein the physical parameters are constant.

So the volume element can be now given by

$$D_i\nabla^2 C_i + D_i\frac{z_i F}{RT}\nabla(C_i\nabla\Phi) + \sum_{j=1}^{1} s_{ij} r_j = 0$$

where D_i is the coefficient of diffusion, S_{ij} gives the coefficient of stoichiometry.

While solving this some boundary conditions are set: electroneutrality condition is maintained, potential value on the outside boundary is taken as 0, boundary conditions of each reacting chemicals must have their rates conform to the flux of intake, and the cathodic current must counterbalance the anodic one.

After this stage, simple arithmetic methods to calculate boundary value problems are incorporated. At a 3D spatial distribution, the final relation between concentrations, total ionic currents (j), and potentials of solution, as well as metal can be obtained as

$$j = -\frac{F^2}{RT}\nabla\Phi\sum_{i=2}^{7} z_i^2 D_i C_i - F\sum_{i=2}^{7} z_i D_i \nabla C_i$$

where i represents the number of chemical species involved, which in this case is 7. Therefore, this equation obtained can be used in several case studies to formulate a corrosion rate if the electrode and solution potential is obtained.

6.3.8 Relevant Data Handling Techniques

Data obtained during experimental procedures may be broadly classified into the following types on the basis of their source:

a. *Structured data*: These are the data sets which comply with the predefined type, format, and structural criteria. For example, the analytical and experimental data obtained under controlled conditions.
b. *Semistructured data*: These are the data sets that are self-describing in nature and possess minimal structural features. For example, sensor generated data on biofilm behavior.
c. *Unstructured data*: These are the data sets lacking an inherent structure. For example, qualitative parametric outputs, random variables, etc.

To ensure a hassle-free handling of the obtained data it is quintessential to divide the data into three groups. The divisions are as mentioned hereafter:

a. *Training data*: A model is built using individual tests or model selection techniques. Optimal parameters are identified using a trained data-analyzing model.
b. *Test data*: The model built with the training data is evaluated followed by testing using hypothesis data.
c. *Confirmation data*: The model built with the training data is evaluated followed by confirmation of finding by test data.

The prevalent data handling techniques used in analysis of laboratory data are as follows:

6.3.8.1 Classification and Regression Trees (CART)

It is a model which predicts the target output value using multiple input values. The algorithmic structure involves a series of conditional and absolute questions and answers, which will link the input data points to the next level of question. The resultant of this algorithm will be a tree structure terminating in points called nodes. The principal components of this algorithm include data splitting rules, stopping rules, and prediction criteria at the end of each terminal node. This method renders an overall accuracy of 97.35% (Nisbet et al., 2009).

Characteristic features of this method are

i. *Nonparametric nature*: The algorithm does not heavily rely on data belonging to certain distribution types.

ii. The impact of outliers (any sampling unit that does not belong to the population of interest) in the input variable is minimum.
iii. Incorporates both testing with the data sets as well as cross checking to determine the goodness of fit.
iv. Use of same variable is allowed in different parts of the tree.
v. This method can be used in conjunction with other methods.

6.3.8.2 K-Nearest Neighbor Method (KNN)

This algorithm classifies a new data point on the basis of attributes and training samples. For a given query point K, a number of closest neighborhood points are identified and are assigned to the same class. In other words, it uses a majority vote in the classification of K objects. This method is solely based on memory of the algorithm and does not involve any model to fit. Hence, any ties can be broken at random. The average accuracy of this method is 71.28% (Danades et al., 2016).

Characteristic features of this method are

i. All the data points feature in the n dimensional Euclidean space.
ii. Classification is withheld till a new instance arrives.
iii. Comparison of feature vectors of each point forms the basis of classification.
iv. Real or discrete value can be assumed by the target function.

6.3.8.3 Clustering

The method of identifying similar groups of data in a data set is called clustering. This method attempts to fit a data set with a particular model assuming the data are the resultant of a mixture of underlying probability distributions. The method is broadly classified into hard clustering where each data point belongs completely to a cluster or not, and soft clustering where each data point is assigned with a probability to be grouped into a cluster. Two major techniques involved under this approach are:

i. *Expectation maximization*: In this approach a parametric probability distribution represents each particular cluster. This is followed by parameter estimation by using an iterative refinement algorithm. This algorithm comprises of an expectation step and a maximization step. The former step involves probability assignment to each data point while the latter step involves reestimation of model parameters.
ii. *Conceptual clustering*: In this approach a classification scheme is produced for a set of unlabeled data points. A popular method of COBWEB is used for the purpose of incremental conceptual learning. It renders a hierarchic clustering as a classification tree. Each node represents a cluster and its corresponding probabilistic description.

Characteristic features of this method are:

i. Most methods biased towards finding clusters with features such as size, shape, or dispersion.
ii. Methods with least square, as criteria tend to find equal number of data points in each cluster.
iii. Methods based on nonparametric density estimation tend to be least biased.

The efficiency of this method is highly variable on the basis of the basic criteria used for clustering.

6.3.8.4 Artificial Neural Networks

The abstract mathematical method built on the foundations of the modern neuroscience employed for solving complex mathematical data problems is known as neural networks. The most widely used form of this method is that of BPANN. It is a multilayer feed forward method with error back propagation-trained algorithm. The average efficiency of this method is 84.46% (Safi and White 2017).

Characteristic features of this method are

i. Massively parallel distributive structure
ii. Nonlinearity
iii. Input/output mapping
iv. High degree of adaptability and fault tolerance

6.3.8.5 Support Vector Machines (SVN)

In this method, each data item is plotted as a point in an n-dimensional space. Value of each of the n features is assigned as the coordinate value. The classification solution is arrived at by identifying a hyper plane clearly differentiating the two classes. The average efficiency of this method is found to be around 70% (Danades et al., 2016).

Characteristic features of this method are:

i. Effective in high dimensional spaces.
ii. Cases with number of dimensions exceeding the number of samples can be solved well.
iii. Performance decreases with the increase in the size of the data sets.

6.3.9 Optimal Combination

As this area of research has been taken into the limelight recently, it is at the nascent stage where not all of its processes can be analytically determined. Pre-existing biological models have been predominantly used and then modified based on initial, intermediary, and final conditions as boundary values and tailored equations are obtained. One menacing factor about this whole process is most of these analytical equations are based on one or two factors; but in reality, this type of corrosion is a very complex process encompassing many undercurrents of physical, biological, and environmental process. However, since the laboratory experimental data for many a case study have shown that these equations give reasonable accuracy in the resulting values, they can be used.

An optimal combination would be a universal equation encompassing all the possible factors. However, as we increase the number of factors, human-made errors during the data intake also increase leading to an unfavorable result. So, it is advisable to choose equations that correlate the obtained data point to the desired factor. As far as kinetics, due to this being a single substrate mechanism, the Michaelis–Menten model is preferred. In the phase of biofilm formation, Min'kov et al. (2006) have modified and presented an analytical relation between substrate flux concentration

and volumetric flow rate. In the same phase, Muthukaruppan et al. (2013) has presented a relationship of substrate flux concentration to biofilm thickness, substrate concentration on the outside of the biofilm. As the second issue has high accuracy due to the application of the Adhomian method, it might be preferred but it will still depend on available data sets. For the corrosion rate, however, the Plank–Nernst equation solved using the Laplace transform seems more reliable, as the other available option of Butler–Volmer does not always hold true. However, if the specific combination of metal is known to have followed a correlation of potential to rate, the Butler–Volmer equation would yield better results than the generalized version. So, choosing of equation would be mostly circumstantial.

6.4 CONCLUSION

As nature would have it, microbes always find an exception to every generalized theory known to man. So is the case with theorizing the process of MIC. An all-inclusive approach perfectly dealing with all the variables has yet to be found. However, in this chapter an attempt has been made to put together simple techniques of biological experimentation models coupled with data handling methods to formulate optimal combinations. The Kinetic modeling of the biocorrosion process can be effectively accomplished by the Michaelis–Menten model, while that of corrosion rate by Plank–Nernst equation solved using Laplace transform. Usually, MATLAB is used for solving them, as it has many in-built functions for boundary value problems. For example, in the Adhomian Decomposition method in Matlab, for two-point boundary value problems consisting of ordinary differential equations (ODEs)—a built-in function called bvp4c is popularly used. Similarly, the symbolic package of Matlab is used to calculate the analytical derivatives of the Butler–Volmer model. As the Newton–Rhapson system was followed, the linear system in it had been subjected to the BiCG (biconjugated gradient method) with Jacobi preconditioning. This method is frequently used for large systems of linear equations. Any large system of data points are smoothly iterated through predefined programs—significantly reducing calculation times. Artificial neural network was found to be the optimally suited handling method for data inputs of the experimentation processes and subsequent modelings, owing to its high adaptability to variations and superior efficiency. Moreover, if the species is of a rare kind and the studies need to be focused on it's definitive characteristics, an individual based model, like BacSim (Kreft et al., 2001) is more preferable than analyzing them in bricks. A combination of the aforementioned methods is found to be capable of providing a qualitative and quantitative explanation to the process of corrosion induced by microbial infestation.

REFERENCES

Alabbas, F. M., Williamson, C., Bhola, S. M., Spear, J. R., Olson, D. L., Mishra, B., & Kakpovbia, A. E. 2013. Microbial corrosion in linepipe steel under the influence of a sulfate-reducing consortium isolated from an oil field. *Journal of Materials Engineering and Performance*, 22(11), 3517–3529. https://doi.org/10.1007/s11665-013-0627-7

Amadi, S. A., Ukpaka, C. P., & Neeka, J. B. 2007. Mechanisms of the microbial corrosion of aluminum alloys. *Journal of Industrial Pollution Control*, 23(2), 197–208. https://doi.org/10.5006/1.3580810

Beech, I. B., & Gaylarde, C. C. 1999. Recent advances in the study of biocorrosion: An overview. *Revista de Microbiologia*, 30(3), 177–190. https://doi.org/10.1590/S0001-37141999000300001

Beech, I. B., & Sunner, J. 2004. Biocorrosion: Towards understanding interactions between biofilms and metals. *Current Opinion in Biotechnology*, 15(3), 181–186. https://doi.org/10.1016/j.copbio.2004.05.001

Beech, I., Bergel, A., Mollica, A., Flemming, H., Scotto, V., & Sand, W. 2000. Simple methods for the investigation of the role of biofilms in corrosion. *Brite-Euram III Themat*. Netw. N° ERB BRRT-CT98-5084; 5084.

Beech I. B., Cheung C. W. S., Chan C. S. P., Hill M. A., Franco R., & Lino A. R. 1994. Study of parameters implicated in the biodeterioration of mild steel in the presence of different species of sulfate-reducing bacteria. *Int. Biodeterior. Biodegradation*, 34:289–303. 10.1016/0964-8305(94)90089-2

Bouman, S., Lund, D., Driessen, F. M., & Schmidt, D. G. 1982. Growth of thermoresistant streptococci and deposition of milk constituents on plates of heat exchangers during long operating times. *J. Food Prot*, 45, 806–12, 815.

Danades, A., Pratama, D., Anggraini, D., & Anggriani, D. (2016). Comparison of accuracy level K-Nearest Neighbor algorithm and support vector machine algorithm in classification water quality status. 137–141. 10.1109/FIT.2016.7857553.

Gu, T. 2012. New understandings of biocorrosion mechanisms and their classifications. *Journal of Microbial & Biochemical Technology*, 4(4). https://doi.org/10.4172/1948-5948.1000e107

Gu, T. 2014. Theoretical modeling of the possibility of acid producing bacteria causing fast pitting Biocorrosion. *Journal of Microbial & Biochemical Technology*, 6(2), 68–74. https://doi.org/10.4172/1948-5948.1000124

Koch G. H., Brongers M. P. H., Thompson N. G., Virmani Y. P., & Payer J. H. 2001 *Corrosion Cost and Preventive Strategies in the United States*. FHWA-RD-01-156.CC Technologies Laboratories, NACE International, Dublin, OH.

Koch, G. H., Thompson, N. G., Brongers, M. P. H., Virmani, P. Y., & Payer, J. H. 2010. Cost of corrosion, study unveiled. *Nace International*, (NACE), 65(1), 10.

Kreft, J. U., Picioreanu, C., Wimpenny, J. W. T., & Van Loosdrecht, M. C. M. 2001. Individual-based modelling of biofilms. *Microbiology*, 147(11), 2897–2912. https://doi.org/10.1099/00221287-147-11-2897

Liang, R., Aktas, D. F., Aydin, E., Bonifay, V., Sunner, J., & Suflita, J. M. 2016. Anaerobic biodegradation of alternative fuels and associated biocorrosion of carbon steel in marine environments. *Environmental Science and Technology*, 50(9), 4844–4853. https://doi.org/10.1021/acs.est.5b06388

Little, B. J., Richard I, R., & Pope, R. K. (2000). The relationship between corrosion and the bilogical sulfur cycle. *Corrosion NACE International*, 56(394), 194–196.

Maluckov, B. S. (2012). Corrosion of Steels Induced by Microorganism. *Association of Metallurgical Engineers of Serbia (AMES)*, 223–231.

Marangoni, P. R. D., Robl, D., Berton, M. A. C., Garcia, C. M., Bozza, A., Porsani, M. V., … Pimentel, I. C. (2013). Occurrence of Sulfate Reducing Bacteria (SRB) associated with biocorrosion on metallic surfaces in a hydroelectric power station in ibirama (SC)-Brazil. *Brazilian Archives of Biology and Technology*, 56(5), 801–809. https://doi.org/10.1590/S1516-89132013000500011

Meena, V., Indira, K., Kumar, S., & Rajendran, L. 2014. A new mathematical model for effectiveness factors in biofilm under toxic conditions. *Alexandria Engineering Journal*, 53(4), 917–928. https://doi.org/10.1016/j.aej.2014.09.003

Min'kov, L. L., Pyl'nik, S. V., & Dueck, L. H. 2006. Steady-state problem of substrate consumption in a biofilm for a square law of microbial death rate. *Theoretical Foundations of Chemical Engineering*, 40(5), 496.

Muthukaruppan, S., Eswari, A., & Rajendran, L. (2013). Mathematical modelling of a biofilm: The adomian decomposition method. *Biomedical & Life Sciences*, 5(4), 456–462. https://doi.org/10.4236/ns.2013.54059

Nisbet, R., Elder, J. F., & Miner, G. 2009. *Handbook of Statistical Analysis and Data Mining Applications*. Amsterdam: Elsevier.

Olmstead, W. M., & Hamlin, H. 1900 Converting portions of the Los Angeles outfall sewer into a septic tank. *Engineering News* 44, 317–318.

Picioreanu, C., & van Loosdrecht, M. C. M. 2002. A mathematical model for initiation of microbiologically influenced corrosion by differential aeration. *Journal of The Electrochemical Society*, 149(6), B211. https://doi.org/10.1149/1.1470657

Safi, S., & White, A. 2017 Short and long-term forecasting using artificial neural networks for stock prices in Palestine: A comparative study. *Electronic Journal of Applied Statistical Analysis*, North America.

Ukpaka, C. P., Amadi, S. A., Ahuchogu, I. G., & Odharo, J. 2011. Modeling the rate of biocorrosion and the effects of redox-reactions of metals in water environment. *Journal of Engineering and Technology Research*, 3(13), 371–380. https://doi.org/10.5897/JETR11.058

van Loosdrecht, M. C. M., Heijnen, J. J., Eberl, H., Kreft, J., & Picioreanu, C. 2002. Mathematical modelling of biofilm structures. *Antonie van Leeuwenhoek*, 81, 245–256.

Videla, H. A. 2010. Biocorrosion and biofouling of metals and alloys of industrial usage. Present state of the art at the beginning of the new millennium. *Revista de Metalurgia de Madrid*, 39(Extra), 256–264. https://doi.org/10.3989/revmetalm.2003.v39.iExtra.1128

Wells, P. A. (Tony), & Melchers, R. E. 2011. Microbial corrosion of sewer pipe in Australia - Initial field results. *18th International Corrosion Congress 2011*, 1–12.

Wells, T., Melchers, R., Joseph, A., Bond, P., Vitanage, D., Bustamante, H., … Evans, T. 2012. A collaborative investigation of the microbial corrosion of concrete sewer pipe in Australia. *OzWater-12 Australia's National Water Conference and Exhibition*, May, 8–10.

Xu, D., Li, Y., & Gu, T. 2016. Mechanistic modeling of biocorrosion caused by biofilms of sulfate reducing bacteria and acid producing bacteria. *Bioelectrochemistry*, 110, 52–58. https://doi.org/10.1016/j.bioelechem.2016.03.003

Zhiyuan, L., & Zongdi, S. 2017. The carbon trading price and trading volume forecast in Shanghai City by BP neural network, *International Journal of Economics and Management Engineering*, 11(3), 623–629.

7 Application of Data Mining Techniques in Autoimmune Diseases Research and Treatment

Sweta Bhattacharya and Sombuddha Sengupta

CONTENTS

7.1 Introduction 114
7.2 Machine Learning and Knowledge Discovery Tasks 116
7.2.1 Supervised Learning 116
7.2.2 Classification 117
7.2.3 Regression 117
7.2.4 Unsupervised Learning 117
7.2.5 Association Rule Mining 118
7.2.6 Clustering 118
7.2.7 Reinforced Learning 118
7.3 Autoimmune Diseases: An Overview 118
7.4 Role of Data Mining in Autoimmune Diseases 120
7.4.1 Diabetes Mellitus 120
7.4.1.1 Application of Data Mining in Diabetes Mellitus 120
7.4.2 Lupus Disease 122
7.4.2.1 Role of Data Mining in Lupus Disease 122
7.4.3 Myasthenia Gravis 123
7.4.3.1 Application of Data Mining in Myasthenia Gravis 124
7.4.4 Grave's Disease 124
7.4.4.1 Application of Data Mining in Grave's Disease 125
7.4.5 Inflammatory Bowel Syndrome 125
7.4.5.1 Application of Data Mining in Inflammatory Bowel Syndrome 126
7.4.6 Multiple Sclerosis 127
7.4.6.1 Application of Data Mining in Multiple Sclerosis 127
7.5 Conclusion 128
References 128

7.1 INTRODUCTION

The human body is indeed an interesting yet complex unit, formed through million years of evolution, transcending the unicellular to arrive at the complex multicellular. For proper functioning, the body comprises several inherent systems needing to work in harmony to facilitate homeostasis and normal behavior. If a system, or several systems, go awry from their normal function it can lead to severe detrimental or life threatening complications. One of the key systems is the immune system. To explore the immune system or its nuances is beyond the scope of this chapter. Hence, we shall be concentrating on a portion of this vast topic.

Autoimmunity is a situation in which the immune system goes off-course, resulting in an inappropriate response to self elements or self antigens.[1] Earlier, it was proposed that the self reactive lymphocytes were screened and eliminated from the body, preventing an autoimmune response. However, later studies have shown that not all T-cells and B-cells that have the potential to target self antigens are successfully removed. This leads to a situation where the "army is fighting against the king." The damage induced to cells and organs may be due to the antibodies or the T cells of the body. Some common autoimmune diseases have been listed in Table 7.1.

Data mining simply means the process of extracting data, or a mining process to squeeze out valuable information from a collection of existing data. Other terms commonly used in place of data mining, although having a slight different meaning,

TABLE 7.1
Some Common Autoimmune Diseases in Humans

Disease	Self Antigen	Immune Response	Reference
	Organ Specific Diseases		
Hashimoto's Thyroiditis	Thyroid cells	T_{DTH} cells and auto- antibodies	Caturegli et al. 2014[3]
Pernicious Anemia	Gastric parietal cells, intrinsic factors	Auto-antibodies	Bizzaro et al. 2014[4]
Myocardial Infarction	Heart	Auto-antibodies	Li et al. 2013[5]
Myasthenia Gravis	Acetylcholine receptors	Auto-antibodies	Lindstrom et al. 1998[6]
Grave's disease	Thyroid stimulating hormone receptor	Auto-antibody	Weetman et al. 2000[7]
Goodpasture's syndrome	Lung and renal basement membrane	Auto-antibodies	Greco et al. 2015[8]
	Systemic Autoimmune Disease		
Systemic lupus erythematosus (SLE)	DNA, nuclear protein, RBC	Auto-antibodies, complexes	Tan et al. 1982[9]
Multiple sclerosis	Brain	T_{H1} and T_C cells, auto-antibodies	Lublin et al. 1996[10]
Rheumatoid arthritis	Connective tissues	Immune complex, auto-antibody	McInnes et al. 2017[11]
Ankylosing sponkylitis	Vertebrae	Immune complex	Chen et al. 2016[12]

are: data dredging, data archaeology, data analysis, data extraction, etc. Nowadays, people synonymously use this term with another popular term called knowledge discovery from data (KDD).[2] This process of knowledge extraction of useful data from the bulk data has several steps, as listed below:

- *Data cleaning*: To remove noise or inconsistency or misleads in the data collected or stored.
- *Data integration*: A step where multiple sources of data are combined.
- *Data selection*: A step where which analysis is to be done is determined, and the data pertaining to that cause is screened.
- *Data transformation*: Data is consolidated to make it suitable for mining purposes. This is done by summary or aggregation operations.
- *Data mining*: Intelligent methods are used to extract data patterns.
- *Pattern evaluation*: Done to identify truly fascinating patterns.
- *Knowledge presentation*: Various visualization procedures are employed to display the mined knowledge to the interface user.

The steps of data cleaning, data integration, data selection, and data transformation come under the category of preprocessing of the data collected.[13] Data mining is a single and important step in the entire process, as it helps in the retrieval of important patterns to make the vast data collection potentially useful to the interface user. A flowchart of the process of knowledge retrieval has been shown in Figure 7.1.

The incidence of autoimmune diseases is great and vast in number. Through the years, the data pertaining to these diseases is increasing at an exponential rate. Data mining offers a suitable and easy method of dealing with this avalanche of data, and hence, pattern recognition and deciphering becomes easier.[2] The later sections of this chapter explain in great detail the crux of the concept of data mining, along with its application, in order to diagnose or treat some of the commonly occurring autoimmune diseases in the world.

FIGURE 7.1 Knowledge retrieval process.

7.2 MACHINE LEARNING AND KNOWLEDGE DISCOVERY TASKS

Machine learning is an application of artificial intelligence, which enables systems to learn information automatically and improve based on training experience, without being programmed explicitly. The basic objective is to develop computer programs able to access data and learn by themselves, and help in predictions based on examples provided during training. The machine learning technique helps to create systems that can adapt and learn on their own—based on the experience of analyzing data patterns.[14]

Knowledge discovery, on the other hand, is a concept including methods and techniques to extract useful information and understanding from data. The knowledge discovery process comprises multiple stages to be followed sequentially. The steps of knowledge discovery are selection, preprocess, transformation, data mining, interpretation, and evaluation, as shown in Figure 7.2. The crucial part of the knowledge discovery process is the application of machine learning algorithms when analyzing data for mining important information to understand correlations and dependencies among variables.[15]

Machine learning is broadly categorized in to three types, which are supervised learning, unsupervised learning, and reinforced learning.

7.2.1 Supervised Learning

In supervised learning, the system learns a function called "target function" from a labeled training data set. The training data set is analyzed and an inferred function is produced, which is used to predict the value of output variables or dependent variables. The input values of the function consist of a set of attributes and features. The subset of the input variables, for which the output is already known, is considered as training data used to make predictions for the values of unknown output variables.

FIGURE 7.2 Steps involved in knowledge discovery.

The supervised learning method is also categorized into two types—classification and regression, on the basis of learning techniques. The popular techniques used in supervised learning are decision trees, instance based learning, k-nearest neighbors (k-NN), genetic algorithm, artificial intelligence, and support vector machines. The details of these techniques are briefly discussed in the following section.[16]

- *Decision Tree*: A decision tree is basically a tree structure having root, branches, and leaves. The tests conducted on an attribute is denoted by the internal nodes, and the branches represent the outcome of the tests. The leaf nodes represent the classes, and the highest node is considered as the root note with the utmost significance.[17]
- *Instance Based Learning* (IBL): It is a learning algorithm where new problem instances are compared with instances found in training data sets to adapt the model to formerly unseen data.[18]
- *k—Nearest Neighbors* (k-NN): It is a type of IBL used for classification of new objects based on attributes and training sample data. The *k* number of objects closest to a given query points are identified and classified in this technique.[13]
- *Genetic Algorithm*: Is a search-based algorithm founded on the theory of natural selection and genetics. In genetic algorithm there are multiple options or choices of solutions for a particular problem. These solutions are recombined and mutated to get the optimal solution, which would be extremely time-consuming using traditional approaches.[19]
- *Artificial Intelligence (AI)*: As said by the father of AI—John McCarthy, it is "The science and engineering of making intelligent machines, especially intelligent computer programs." It is a technique to program a computer or a computational robot to think logically and intelligently, similar to the working of a human brain. As human beings act based on experience gained, AI also uses knowledge gained (through training of data sets) to solve problems, perform predictions, and generate outputs from respective inputs.[20]

7.2.2 Classification

The concept of classification is similar to categorization, wherein a machine-learning technique is used on known data to categorize new data into distinct classes. As an example of class, we can consider blood groups or classification of mails as spam.[13]

7.2.3 Regression

Here the objective is to predict continuous or discrete numerical values and identify the linear relationships existing among attributes in the dataset. The popularly used regression algorithm types are "basic," "linear," and "polynomial."[13]

7.2.4 Unsupervised Learning

In unsupervised learning, the system tries to discover information from unlabeled data sets and find associations between variables. The training data consists of only

instances without any data labels. "Association rule mining" and "clustering" are forms of unsupervised learning algorithms.

7.2.5 Association Rule Mining

This technique aims to discover interesting relationship of correlations ,which exist among variables in data sets. The most popular association rule mining algorithm "apriori," is an exploratory data analysis technique used in a wide range of research in biological science and bioinformatics.[21]

7.2.6 Clustering

Clustering is a technique to form groups, or clusters, of data based on common attributes or characteristics. The method analyses the input data set and—based on the nature of the data—forms clusters or groups depending on informative patterns existing in the data set.

7.2.7 Reinforced Learning

It is a family of machine learning techniques wherein the system learns through direct interaction with the environment in order to maximize the notion of cumulative award. The system is completely unaware of the behavior of the environment, and hence, applies trial and error methods to find plausible solutions to a problem.[22]

7.3 AUTOIMMUNE DISEASES: AN OVERVIEW

The immune system has developed over the course of evolution to protect the body from invading pathogens. However, the immune system can lead to pathogenesis in two cases: in the case of immunodeficiency diseases and autoimmune diseases.[23] Immunodeficiency diseases results from the failure of one part, or more than one part, of the immune system to protect the body. An autoimmune disease, on the other hand, is a case where the body's immune system fails to distinguish between itself and foreign antigens. Autoimmune diseases are rare and have been seen to affect 3%–5% of the world population, with thyroid diseases and diabetes among the most common of these conditions.[23] Statistical records have shown there are more than 100 different types of autoimmune diseases, some being organ-specific while others are systemic in nature.

Immune tolerance has been seen to be an innate characteristic, which was reported by Macfarlane Burnet in 1978.[24] Tolerance denotes the state where the immune system is accustomed to the innate antigens present in the body, and hence does not attack the cells or organs from the same body. The concept of autoimmunity was still not fully accepted, even though the concept was reported as early as the twentieth century by Paul Ehrlich who coined the term "horror autotoxicus."[25] Autoimmune diseases can be either organ-specific or systemic in nature. In organ-specific disease, the immune response is induced in response to a single antigen on a particular organ.[1] The cells of that organ are damaged and overproduction or overstimulation of the antibodies

may even block further the functioning of the organ. In case of systemic autoimmune diseases, the immune response is towards a wide range of antigens. A large number and types of tissue can be damaged, and the disease is often marked by the hyper stimulation of the T and B cells.[23]

Central tolerance in the thymus and the bone marrow help shape the immune system of the body. This way, the self reacting antibodies are removed in the thymic medulla so that the immune system does not target the self antigens. Even in the presence of the strict surveillance of the central and peripheral tolerance, some of the self reactive antibodies escape leading to the manifestation of autoimmune disease.[26]

The occurrence of autoimmune disease has been seen to be greater in females than males; however, the geoepidemiology becomes even more complex with the consideration of factors like age, ethnicity, gender, etc. The exact reason for the occurrence of autoimmune disease is still under much debate, however, a few main points need to be mentioned at this point (Wang 2015).[23]

Genetics can be a plausible cause for the onset of autoimmune diseases. Diseases like Addison's disease or autoimmune polyendocrinopathy syndrome type I, etc. are example of this. Mutation in the autoimmune regulator gene has been seen, which affects the negative selection in the thymus and bone marrow. Mutation in the tumor necrosis factor (TNF) receptor superfamily member 6 (TNFRSF6), TNFRSF6 bound membrane ligand, and caspase 10 can also lead to apoptosis of lymphocytes, which can lead to the accumulation of a large number of mononuclear cells in the lymphoid tissue, resulting in failure to delete the autoreactive cells.[26] Linkage variants of human leukocyte antigens (HLA), which is the gene product of the major histocompatibility complex (MHC), have also been held responsible for the onset of autoimmune disorders. Genome wide association studies (GWAS) have documented several of these variants.[27] Lastly, epigenetic and stochastic factors have also been known for causing autoimmune disorders. Environmental factors, like infectious agents and molecular mimicry of the pathogens, also hold important roles in these diseases.

The list for the causes of autoimmunity is exhaustive. Substances like glucose and cellular apoptosis have also been designated as causal agents in recent studies. For reference, the main causes have been given in Figure 7.3. Early diagnosis is possible nowadays with the advent of new technology and better diagnosis procedures. One such tool, as mentioned before is data mining. The following sections now elaborate

FIGURE 7.3 Causes of autoimmunity.

on the role of data mining in autoimmune disease research, and further show how biology and information technology may complement each other.

7.4 ROLE OF DATA MINING IN AUTOIMMUNE DISEASES

7.4.1 Diabetes Mellitus

The increase in the sedentary lifestyle due the advancement of technology has given diabetes the name, "disease of modern society." Lack of exercise, obesity, and consumption of foods rich in trans fat and saturated fats are some of the key reasons why an individual might be affected with diabetes. According to the data published by the CDC in 2017, diabetes affects 30.3 million individuals of which 7.2 million remain undiagnosed.[28] Biologically, diabetes is a disease marked by a spike in the blood glucose level of an organism.[29] This has been attributed to two reasons: (1) the body cells become unresponsive to insulin, and (2) the body does not produce insulin. Based on these characterizations we can classify them as Type I and Type II.[30] Overtime, an increase in blood sugar level can have serious complications such as imparting damage to eyes, kidney, and nerves, provoking heart attacks, strokes, etc. There are many tests to diagnose diabetes and some are listed below.[28,31]

- *Fasting plasma test*: The blood glucose level at a point in time is checked after the individual has fasted for more than 8 hours.
- *A1C test*: This test gives the average amount of blood glucose over a period of 3 months. Other names of this test include, HbA1C, glycated hemoglobin, and glycosylated hemoglobin test.
- *Random plasma glucose test*: Done for the diagnosis of the disease when the individual has not fasted. Blood for this analysis can be drawn at any time.

7.4.1.1 Application of Data Mining in Diabetes Mellitus

The basic objective behind using data mining is to extract valuable information from data sets using classification, association, and clustering techniques in order to aid in the prediction of diseases. Generally, multiple diagnostic tests are conducted on patients, which are often time consuming and painful. The number of tests could be reduced with the use of data mining techniques by analyzing general, physiological, and other relevant factors to produce accurate detection results. Diabetes mellitus is one such disease where data mining techniques have been used significantly across different age groups of patients to ensure early disease prediction. The study by Aiswarya et al. (2015) reported on the use of decision trees and naïve Bayes algorithms in the diagnosis of the disease at an early stage, providing an opportunity for quicker treatment of patients.[32] The naïve Bayes algorithm is based on Bayes' theorem, which assumes the existence of independence among the contribution or predictor variables. The naïve Bayes algorithm is often chosen for analyzing data sets due to its simplistic technique and applicability on large data sets. It is also evident that the detection of diabetes mellitus is often based on physical tests, which are prone to errors leading to falsified results. The indecisions related to these tests could be

rectified with the use of data mining algorithms. The k-NN and ANFIS algorithms have proved to be more accurate in classification of the disease in comparison to other existing approaches.[33] In such cases, the k-NN algorithm helps in classifying the objects and produces predictions based on training data sets in the attribute space. The k means an algorithm uses a partitioning method depending on the input parameters for clustering of the dataset. The ANFIS algorithm uses a combination of the adaptive neural network and fuzzy inference system for prediction of the disease from a classified dataset.

Diabetes mellitus is often detected among patients with irregular lifestyles. The study by Pardha Repalli (2011) highlighted the likelihood of people being affected by the disease based on their lifestyle-related activities. The various factors contributing towards the disease have also been analyzed using decision tree algorithms. The results revealed that age is a significant factor in patients aged >65 years, who are more prone towards the disease in comparison to people aged <20 years.[34] Breault (2001) used naïve Bayes data mining algorithms in predicting the diabetic status among patients, and results generated had a 65% to 85.5% accuracy in the prediction of the disease considering eight variables.[35] Data mining techniques have also been used to predict the chances of heart disease among diabetic patients. In his study, Parthiban (2011) used a naïve Bayes data mining classification algorithm on a data set of patients of Indian origin, while also using the WEKA tool for implementation purposes. The study helped to generate an optimized prediction model using a minimal training data set while considering four major attributes namely age, gender, blood pressure, and blood sugar level of the patients. The results of the study helped to classify the age groups, ranges of body weight, and blood pressure and sugar levels of patients suffering from diabetes.[36]

In cases where medical diagnosis is not evident to detect diabetes mellitus among patients, clustering and classification techniques have proven to provide more accurate results. The use of machine learning techniques have helped in this regard using various tools such as WEKA, Tangra, and MATLAB. Also, these tools help to use various classification techniques on the same data set in minutes, yielding more accurate predictions than the conventional time-consuming methods used in the health care sector. The study by Rakesh (2013) used hybrid data mining techniques combining ANFIS and PCA algorithm, neural network and PCA Algorithm on the data set. The results provide an interesting insight in analyzing the contributions of the various factors towards diabetes mellitus disease, and the predictions have shown more accuracy than the traditional algorithmic approaches.[37]

Diabetes mellitus has become one of the major health issues in today's world, and the major challenge is to prevent the disease rather than cure it. Also, since the disease already affects large numbers of patients there exists an enormous data set in the form of medical and health records. Computer aided diagnostic (CAD) systems could be used successfully for knowledge discovery on these patient records, followed by application of machine learning and data mining algorithms for the detection and prediction of the disease.[38]

Diabetes is an everlasting disease, which has affected numerous lives making it a societal and health care challenge among the present generation. The confusion in detection of this disease is often guided by a lack of awareness and poor detection mechanisms. Data mining approaches help in this regard by investigating large data

sets and discovering knowledge with the aim of disease prediction. The patterns found during this process also help in the detection of various other diseases related to diabetes mellitus, thus providing the ability for better treatment planning and risk management.

7.4.2 Lupus Disease

Lupus is a debilitating disease, where the body targets a wide variety of its own cells and organs. It can attack the joints, skin, kidney, heart, brain, lung, etc. Lupus is more common in females than males and statistics have shown that African American women are more prone to it than other ethnicities such as Caucasian or Asian. Hispanic women have been documented to present a more severe form of this disease.[39] Several kinds of lupus exist and their categorizations are given as follows:[39]

- *Systemic lupus erythematosus (SLE)*: The most common form of lupus; it can affect many parts of the body. It can manifest in a severe or a mild form.
- *Discoid lupus*: These give rise to red rashes that do not fade or go away.
- *Subacute cutaneous lupus*: Results in sores when the body has been exposed to sunlight.
- *Drug induced lupus*: Certain medicines can cause an immune response inside the body. These symptoms usually go away after medication intake stops.
- *Neonatal lupus*: It is an extremely rare case of lupus and affects newborns due to antibodies present in the mother.

The common symptoms include pain or swelling in joints, fevers with no known causes, mouth ulcers, swelling in the face and around the eyes, red rashes (butterfly rashes), hair loss, etc. Also, these symptoms overlap with other diseases making detection rather difficult. The symptoms have also been seen to "come and go." These advents and departures are said to be "flares," and common medical practice aims to make these flares less painful.[40] Till now there is no cure for lupus, however, changes in lifestyle and medicines can help to control this disease. The commonly available diagnostic test is the antinuclear antibody testing (ANA).

7.4.2.1 Role of Data Mining in Lupus Disease

Lupus disease has been commonly detected among patients in American, African, Caucasian, and Hispanic countries. The patients suffering from the disease are also prone to cardiovascular and osteoporosis diseases. The disease gets mostly detected at a later stage, due to lack of awareness, leading to high risk of infection and depression. Hence, data mining techniques could be used in the detection of this disease at an early stage to provide the chance of treatment among patients. The study by Gomathi (2015) uses the decision tree approach for the prediction of lupus among patients based on eleven criteria pertinent to the disease, as derived by the American College of Rheumatology. The criteria considered in the study are malar rash, photosensitivity, discoid rash, nonerosive arthritis, oral ulcers, pericarditis, neurological disorders, hematologic disorders, positive antinuclear antibody, and immunologic disorder. Being a technique without any requirement of parameter setting, the decision tree

has helped to achieve results which are easier to read and comprehend. The Rapid Minor tool was used in the study where 15 patient data sets were used for creating the decision tree model. The data sets consisted of the clinical profiles of the patients and the generated decision tree revealed age as the most significant attribute, followed by gender. Caucasians were more affected by the disease than Hispanics, Americans, and other groups.[41]

A similar study was conducted by John (2013) highlighting the pattern of Lupus disease among patients based on age, gender, ethnicity and social class factor.[42]

Classification and regression tree (CART) algorithms have also been used in the analysis and prediction of lupus disease. The CART algorithm uses a sequence of questions whose answers build subsequent questions, thus forming a tree structure. The algorithm computation terminates when all questions are answered and no questions are left. The computation of specificity, sensitivity, and accuracy in the study of lupus disease has justified the use of CART algorithms to generate more accurate results in the prediction of the disease.[41] Another study on lupus disease used a data set consisting of seven factors pulled from the case sheets of patients suffering from the disease. The factors considered in the study were age, gender, sample type of serum and plasma, ACR criteria, disease activity (mild, moderate, and severe), and tests conducted. The performance of the algorithm used was validated against the results generated using decision tree, support vector machine, and ID3 algorithm. ID3 is a decision-making algorithm invented by Ross Quinlan, which helps to create a decision tree from an input data set.

7.4.3 Myasthenia Gravis

This is a chronic autoimmune disease demarcated by extreme weakness and the degeneration of skeletal muscles.[43] The name is derived from the Latin "grave muscle weaknesses." This disease has no known cure; however, treatment can lead to control of symptoms, thus improving the quality of life for the person afflicted. Nerve cells run through the length of the body and the transmission of impulses through them makes the day-to-day function of our body relatively simpler. In Myasthenia gravis, there is an error in nerve signal transduction via the neurons due to the generation of antibodies against the acetylcholine receptors.[44] The generated antibodies bind to the receptor ligand interaction sites and prevent the interaction of the receptors with acetylcholine. This hinders the transmission of the impulse and the polarization cannot cross the synapse, resulting in an interruption in the transmission network. Reports have shown that people with myasthenia gravis can develop thymomas, which may result in cancer.[43]

The common symptoms seen are drooping shoulders, difficulties in swallowing, weaknesses in arms and legs, and shortness of breath and blurred vision due to weaknesses in muscles controlling eye movements.[44] Years of dealing with the disease has brought out a number of diagnostic methods such as:

- *Edrophonium test*: This test involves injection of edrophonium chloride, which relieves muscle pain by blocking the breakdown of acetyl choline and increasing the neurotransmitters concentration at the neuromuscular junction.

- *Blood test*: Done to find out the presence of antibodies against the acetylcholine receptors. Anti MuSK is another antibody checked for in suspected cases of myasthenia gravis.
- *Diagnostic imaging*: Done using CT-Scans or magnetic resonance imaging to look for thymomas.
- *Electrodiagnostics*: This involves repeated stimulation of nerves with small pulses to tire the muscles. If the person suffers from myasthenia gravis, he or she will not respond to this treatment as the nerve transmission is hindered.

7.4.3.1 Application of Data Mining in Myasthenia Gravis

Myasthenia gravis affects nearly thirty million patients a year. Computational modeling techniques using 3D print technology have helped immensely in accurate diagnosis of the disease. There are many diseases with similar symptoms. As an example, tumors, strokes, nerve and muscle diseases have common symptoms of distorted speech and difficulty in swallowing similar to myasthenia gravis. Hence, often diagnostic tests fail to detect the disease early enough often because of physicians' erroneous judgments. Tumors and strokes are detected through MRI scans of the brain, muscular problems are detected through blood tests, and myasthenia gravis is detected through EMGs or nerve conduction tests. In order to eradicate confusions related to such expensive tests, studies suggest the use of machine learning, neural networks, and data mining algorithms for better predictive accuracy of the disease.

The graph theoretical method is an approach to help for data visualization and solution planning of the disease. Medical practitioners have used such an approach to map different symptoms with the hypothesis pertaining to the disease and perform multiple levels of investigations in parallel. The Bayesian network algorithm is one such graph theoretical algorithm where the probabilities of occurrence of the disease are mapped with the symptoms to provide more accurate and early detection of the disease.[45]

7.4.4 Grave's Disease

This particular autoimmune disease affects the thyroid leading to a symptom called hyperthyroidism. In this case, there is over stimulation of the thyroid gland leading to complications via the overproduction of thyroxine. People who are afflicted with complications like pernicious anemia, Addison's disease, celiac disease, and vitiligo are prone to develop Grave's disease. Here, the body produces antibodies against the hormone-producing cells of the body.[46] These antibodies mimic the thyroid-stimulating hormone, which is produced by the pituitary gland in the brain. Greater amounts of thyroxine are secreted due to overactivation of the gland cells in the thyroid, which leads to hyperthyroidism.

The common symptoms for hyperthyroidism are, shaking hands, a fast and irregular heartbeat, goiters, insomnias, muscle weaknesses, weight loss, and even sterility or erectile dysfunction. Females have been observed to get the disease more frequently than men, and even individuals aged below 40 years tend to get the disease.[47,48] It can lead to severe complications during pregnancy leading to preterm

births, fetal thyroid dysfunctions, or even maternal heart failures and preeclampsias. It tends to be even more fatal when it deregulates the heartbeat rhythm, causing congestive heart failure.[46]

There are diagnostic tests able to determine whether a person is inflicted with Grave's disease. Some of these are listed below:

- *Thyroid functioning test*: A sample of the blood is drawn from the patient and the levels of thyroxine or TSH are checked. A high level of thyroid hormone plus a low level of TSH signifies the presence of Grave's disease.
- *Radioactive Iodine uptake (RAIU)*: This test tells us how much iodine the thyroid takes up. A high uptake of iodine suggests Grave's disease.
- *Antibody testing*: Blood samples drawn from the patient are checked for antibodies against specific receptors of cells producing thyroxine. If such antibodies are found they can suggest the presence of Grave's disease.

7.4.4.1 Application of Data Mining in Grave's Disease

Various studies have been conducted for the prediction of Graves' disease among patients. One of the methods used is the application of neural networks to predict the disease among patients treated with antithyroid medications. A study by Orunesu (2004) treated patients with methimazole for 18 months and found that in the majority of the patients there was a relapse after 2 years of therapy. The factors responsible for the relapse were difficult to predict. Artificial neural network approaches were used to predict the chances of relapse. Initially, 27 variables were considered as part of the therapy, which were then reduced to only 7 variables, and their combinations were finally considered to be more effective in predicting the factors responsible for the relapse. The optimized seven variables contributing significantly were heart rate, presence of thyroid bruits, psychological aspects, serum TGAb and fT4 levels at presentation, thyroid USG results, and cigarette smoking.[49,50]

Multilayer neural networks and learning vector quantization techniques have also been used in thyroid disease detection by classifying thyroid data. The interpretations of thyroid data using algorithms such as naïve Bayes, decision trees, back propagation neural networks (BPNNs), and support vector machines along with medical examinations have enabled a more accurate diagnosis and detection of the disease in comparison to traditional test-based approaches.[49–51]

7.4.5 Inflammatory Bowel Syndrome

In this particular autoimmune disease, the intestine gets inflated due to the action of the immune system on the intrinsic antigens associated with this region.[52] This condition is surrounded by a lot of controversy as to whether it really is an automimmune condition, or simply the action of the human immune system on pathogens, which might induce an inflammation as a side effect. However, both autoimmune and immune-mediated phenomena have been linked to this condition.[53] It has two main forms of manifestations: Crohn's disease and ulcerative colitis.[54] If we consider it as an autoimmune condition, then the body produces antibodies against the intestinal epithelial cells in both cases; however, antibodies against human tropomyosin fraction

five has been seen in the case of ulcerative colitis. Furthermore, in the case of Crohn's disease, antibodies against *Saccharomyces cerevisiae* have been found and against perinuclear antineutrophil cytoplasmic antibodies have been seen to be common in the case of ulcerative colitis.[52]

Symptoms of this disease include cramps, diarrhea, weight loss, iron deficiency anemia, severe urgency to have bowel movements, etc. Other complications associated with this condition include a perforated or ruptured bowel, fistulae and perianal disease, toxic megacolon, and narrowing of the intestines.[54] Common forms of examinations for diagnosis include:

- *Complete blood count*: An increase in the number of white blood cells indicate inflammatory bowel syndrome.
- *Barium X-ray*: The patient is asked to intake the chalky white solution so that the upper part of the intestines can be scanned using X-rays. This method, however, is seldom used due to potential health complications.
- *Sigmoidoscopy*: A flexible tube with a camera attached is used to scan the last one-third region of the patient's large intestine. The anus and the intestinal walls are carefully scanned for ulcers, bleeding, or inflammations arising from the condition.
- *Capsule endoscopy*: The patient is made to swallow a small capsule with an inbuilt camera. The receiver system is worn as a belt by the patient. As the camera takes pictures of the gut scanning for symptoms of the disease, the pictures are sent to the receiver for subsequent download and analysis.

7.4.5.1 Application of Data Mining in Inflammatory Bowel Syndrome

Crohn's disease and ulcerative colitis (UC) are types of inflammatory bowel diseases, which are caused by abnormal responses of the body's immune system. In case of Crohn's disease, the gastrointestinal tract gets affected, which requires immediate medical attention. Studies have suggested that the diet plays a major role in this disease, and hence a study at Duke University created a web-based application to track food intake among patients. The participants would log their regular food intake and their fitness factors pertaining to the disease. The objective was to identify if any correlation existed among both of these factors. A correlation analysis was performed using advanced data mining approaches to understand the dependencies among the factors potentially contributing towards the disease. Due to the availability of EHRs and large data sets, the predictive analysis of factors impacting occurrence of such diseases have become extremely important to improve the quality of patients' lives.[55] A study by Eric Johnson (2012) highlighted the use of logistic regression using Bismarck on patients' data sets to analyze risk factors related to inflammatory bowel disease. The correlative factors were identified using a spatial data analysis technique and presented to researchers using data visualization tools for better insight. The Bismarck framework has proved to be successful in developing predictive models for the detection of the disease. Data mining approaches were successfully used to reduce irrelevant data in the huge and heterogeneous multidimensional database related to the disease, thus making predictions more accurate, specific, and relevant.[56,57]

7.4.6 Multiple Sclerosis

In this disease, the inherent immune system attacks the body's central nervous system (CNS), damaging the myelin sheath underlining nerve cells and oligodendrocytes. The main damage is done by the T cells, which enter the CNS through the blood vessels and produce inflammatory and degenerative changes.[58,59] The exact reason as to why these T cells become active against the CNS is still unknown and intense research is still in progress. This ailment comprises four main types: relapsing remitting multiple sclerosis (RRMS), secondary progressive multiple sclerosis (SPMS), primary progressive multiple sclerosis (PPMS), and progressing relapsing multiple sclerosis (PRMS).[60] This is indeed a complex disease as there seems to be no solid reason as to why this disease appears. Due to its mysterious origin, there are no specific diagnostic tests allocated for this disease. The main tests employed to detect this are MRI, lumber puncture, evoked potential testing, etc.[61] The common symptoms include paralysis, numbness, erectile dysfunction, pregnancy problems, slurred speech, muscle spasticity, etc.

The common treatment methods available include IV steroids, interferon injection, dimethyl fumarate, and copaxone.

7.4.6.1 Application of Data Mining in Multiple Sclerosis

Multiple sclerosis is an extremely unpredictable disease, which causes it to be very difficult to predict accurately. There are various clinical and biological markers related to neuroimaging and neurological aspects of the disease, nevertheless diagnostic accuracy in prediction remains a challenge. Various data mining approaches have proven to be extremely helpful in such complex predictions. Neural and Bayesian networks, decision trees, and linear regression models have been successfully used in such cases to compute accurate predictive outputs from complex data sets. Predictions are possible using both supervised and unsupervised learning methods. In supervised learning, predictions are made from known input/output samples, whereas in unsupervised learning the objective is to discover knowledge input data. Classification and regression are the two types of supervised learning methods. Clustering, summarization, and perception are types of unsupervised learning methods. Classification techniques have been used in the diagnosis of multiple sclerosis. The initial approaches considered in studies have eliminated irrelevant attributes from the data set with the help of feature selection such as the wrapper method. The input variables considered in the study were clinical, magnetic resonance imaging, and neurophysiological variables, which were further ranked using the wrapper algorithm. This led to the most significant input variables having an impact on the dependent variables of the disease using the Expanded Disability Status Scale. The accuracy of the prediction is calculated using a cross validation method, which revealed multilayer perceptron algorithm as the most accurate method in prediction of the disease. Neural networks have also been used to classify the neurophysiological pattern of the disease, namely the brainstem auditory evoked abilities, brain stem trigeminal evoked potential, visual evoked capabilities, and event-related capabilities. The classification of pathological gait patterns using machine-learning methods have also helped in the diagnosis of multiple sclerosis

with the optical coherence tomography method. Advanced computational methods have been used on gene expression data related to multiple sclerosis. Feature set and design classifiers have been used to predict multiple sclerosis in the control set of patients participating in the diagnosis of the disease followed by linear classifiers. The application of classifiers has helped to highlight biomarker genes, dysregulated genes, and discriminatory genes pertinent to the disease. The accuracy of the predictions have been further validated using the leave-one-out cross validation method, which validates the applicability of data mining approaches in the prediction of multiple sclerosis.[62–64]

7.5 CONCLUSION

The various case studies thus reinforce the fact that applications of data mining approaches have become extremely essential and popular for the detection, prevention, and evaluation of treatment effectiveness in autoimmune diseases. Although clinical and physiological tests are available for diagnosis, often these become overwhelming for patients while incurring huge financial expenses. The use of machine-learning approaches in data mining has helped to analyze patterns related to the clinical and physiological symptoms of patients. It has also allowed computing an accurate diagnosis of the disease, thus minimizing the level of confusion and errors. Also, risk analysis and prediction of the disease, based on demographic and social backgrounds, has become possible through mining of information and knowledge discovery. There exists an enormous asset of medical records data, which needs to be utilized and explored more appropriately and optimally. The discovery of knowledge from these health records, using hybrid approaches, should be practiced more extensively for the prediction and analysis of diseases, therefore saving millions of human lives across the world.

REFERENCES

1. Kindt, T. J., R. A. Goldsby, B. A. Osborne, and J. Kuby. *Kuby immunology*. New York: W.H. Freeman, 2007.
2. Han, J., J. Pei, and M. Kamber. *Data mining: Concepts and Techniques*. Waltham, Massachusetts: Elsevier, 2011.
3. Caturegli, P., A. De Remigis, and N. R. Rose. "Hashimoto thyroiditis: Clinical and diagnostic criteria." *Autoimmunity reviews* 13, no. 4, 2014: 391–397.
4. Bizzaro, N., and A. Antico. "Diagnosis and classification of pernicious anemia." *Autoimmunity Reviews* 13, no. 4, 2014: 565–568.
5. Li, J., S. Göser, F. Leuschner, H. Christian Volz, S. Buss, M. Andrassy, R. Öttl, G. Pfitzer, H. A. Katus, and Ziya Kaya. "Mucosal tolerance induction in autoimmune myocarditis and myocardial infarction." *International Journal of Cardiology* 162, no. 3, 2013: 245–252.
6. Lindstrom, J. M., M. E. Seybold, V. A. Lennon, S. Whittingham, and D. D. Duane. "Antibody to acetylcholine receptor in myasthenia gravis Prevalence, clinical correlates, and diagnostic value." *Neurology* 51, no. 4, 1998: 933–933.
7. Weetman, A. P. "Graves' disease." *New England Journal of Medicine* 343, no. 17, 2000: 1236–1248.

8. Greco, A., M. I. Rizzo, A. D. Virgilio, A. Gallo, M. Fusconi, G. Pagliuca, S. Martellucci, R. Turchetta, L. Longo, and M. D. Vincentiis. "Goodpasture's syndrome: A clinical update." *Autoimmunity Reviews* 14, no. 3, 2015: 246–253.
9. Tan, E. M., A. S. Cohen, J. F. Fries, A. T. Masi, D. J. Mcshane, N. F. Rothfield, J. G. Schaller, N. Talal, and R. J. Winchester. "The 1982 revised criteria for the classification of systemic lupus erythematosus." *Arthritis & Rheumatology* 25, no. 11, 1982: 1271–1277.
10. Lublin, F. D., and S. C. Reingold. "Defining the clinical course of multiple sclerosis results of an international survey." *Neurology* 46, no. 4, 1996: 907–911.
11. Agca, R., S. C. Heslinga, S. Rollefstad, M. Heslinga, I. B. McInnes, M. J. L. Peters, T. K. Kvien et al. "EULAR recommendations for cardiovascular disease risk management in patients with rheumatoid arthritis and other forms of inflammatory joint disorders: 2015/2016 update." *Annals of the Rheumatic Diseases* 76, no. 1, 2017: 17–28.
12. Chen, C. H., H. A. Chen, H. T. Liao, C. H. Liu, C. Y. Tsai, and C. T. Chou. "Suppressors of cytokine signalling in ankylosing spondylitis and their associations with disease severity, acute-phase reactants and serum cytokines." *Clinical and Experimental Rheumatology* 34, no. 1, 2016: 100–105.
13. Gnanapriya, S., R. Suganya, G. Sumithra Devi, and M. Suresh Kumar. "Data mining concepts and techniques." *Data Mining and Knowledge Engineering* 2, no. 9, 2010: 256–263.
14. Wilson, R. A., and F. Keil. "The MIT encyclopaedia of the cognitive sciences Bradford." (1999).
15. Mitchell, M. "*An Introduction to Genetic Algorithms*", Cambridge, Massachusetts: MIT Press, ISBN: 0-262-13316-4, 1996.
16. Hastie, T., R. Tibshirani, and J. Friedman. "Overview of supervised learning." In *The Elements of Statistical Learning*, pp. 9–41. Springer, New York, 2009.
17. Kohavi, R., and J. Ross Quinlan. "Data mining tasks and methods: Classification: decision-tree discovery." In *Handbook of data mining and knowledge discovery*, pp. 267–276. Oxford University Press, Inc., 2002.
18. Aha, D. W., D. Kibler, and M. K. Albert. "Instance-based learning algorithms." *Machine Learning* 6, no. 1, 1991: 37–66.
19. Man, K.-F., K.-S. Tang, and S. Kwong. "Genetic algorithms: Concepts and applications [in engineering design]." *IEEE transactions on Industrial Electronics* 43, no. 5, 1996: 519–534.
20. Forbus, K. D. "Qualitative process theory." *Artificial Intelligence* 24, no. 1, 1984: 85–168.
21. Agrawal, R. "Fast algorithms for mining association rules in large databases." In *Proc. 20th International Conference on Very Large Data Bases*, Sept, 1994, pp. 478–499. 1994.
22. Alpaydin, E. *Introduction to Machine Learning*. Cambridge, Massachusetts London, England: MIT press, 2014.
23. Wang, L., F.-S. Wang, and M. E. Gershwin. "Human autoimmune diseases: A comprehensive update." *Journal of Internal Medicine* 278, no. 4, 2015: 369–395.
24. Wang, L., F.-S. Wang, C. Chang, and M. Eric Gershwin. "Breach of tolerance: Primary biliary cirrhosis." In *Seminars in Liver Disease*, 34, no. 3, pp. 297–317. Thieme Medical Publishers, 2014.
25. Salinas, G. F., F. Braza, S. Brouard, P.-P. Tak, and D. Baeten. "The role of B lymphocytes in the progression from autoimmunity to autoimmune disease." *Clinical Immunology* 146, no. 1, 2013: 34–45.
26. Willrich, M. A. V., D. L. Murray, and M. R. Snyder. "Tumor necrosis factor inhibitors: Clinical utility in autoimmune diseases." *Translational Research* 165, no. 2, 2015: 270–282.

27. Cui, Y., Y. Sheng, and X. Zhang. "Genetic susceptibility to SLE: Recent progress from GWAS." *Journal of Autoimmunity* 41, 2013: 25–33.
28. National Diabetes Statistic Report, 2017, Available at: https://www.cdc.gov/features/diabetes-statistic-report/index.html cited on 15-12-2017
29. Kawasaki, E. "Type 1 diabetes and autoimmunity." *Clinical Pediatric Endocrinology* 23, no. 4, 2014: 99–105.
30. Kahaly, G. J., and M. P. Hansen. "Type 1 diabetes associated autoimmunity." *Autoimmunity Reviews* 15, no. 7, 2016: 644–648.
31. Atkinson, M. A., and G. S. Eisenbarth. "Type 1 diabetes: New perspectives on disease pathogenesis and treatment." *The Lancet* 358, no. 9277, 2001: 221–229.
32. Iyer, A., S. Jeyalatha, and R. Sumbaly. "Diagnosis of diabetes using classification mining techniques." *International Journal of Data Mining & Knowledge Management Process*, 5, no. 1, 2015: 1–14.
33. Vijayan, V., and A. Ravikumar. "Study of data mining algorithms for prediction and diagnosis of diabetes mellitus." *International Journal of Computer Applications* 95, no. 17, 2014: 12–16.
34. Repalli, P. "Prediction on diabetes using data mining approach." Stillwater: Oklahoma State University 2011.
35. Breault, J. L. "Data mining diabetics databases: Are rough sets a useful addition." In *Proceedings of the 33rd Symposium on Interface, Computing Science and Statistics*, *Orange, CA, USA*, pp. 13–16. 2001.
36. Parthiban, G., A. Rajesh, and S. K. Srivatsa. "Diagnosis of heart disease for diabetic patients using naive bayes method." *International Journal of Computer Applications* 24, no. 3, 2011: 7–11.
37. Motka, R., V. Parmarl, B. Kumar, and A. R. Verma. "Diabetes mellitus forecast using different data mining techniques." In *Computer and Communication Technology (ICCCT), 2013 4th International Conference on*, pp. 99–103. IEEE, 2013.
38. Hosseinpour, N., S. Setayeshi, K. Ansari-asl, and M. Mosleh. "Diabetes Diagnosis by Using Computational Intelligence Algorithms." *International Journal of Advanced Research in Computer Science and Software Engineering* 2, no. 12, 2012: 71–77.
39. Disease, L. Available at: https://www.ninds.nih.gov/Disorders/All-Disorders/Neurological-Sequelae-Lupus-Information-Page, cited on Dec 15, 2017.
40. Tan, E. M., A. S. Cohen, J. F. Fries, A. T. Masi, D. J. Mcshane, N. F. Rothfield, J. G. Schaller, N. Talal, and R. J. Winchester. "The 1982 revised criteria for the classification of systemic lupus erythematosus." *Arthritis & Rheumatology* 25, no. 11, 1982: 1271–1277.
41. Gomathi, S., and V. Narayani. "Applying decision tree algorithm to predict lupus using Rapid Miner." *International Journal of Applied Engineering Research* 10, no. 9, 2015: 2015.
42. Reynolds, J. A., and I. N. Bruce, "*Overview of the Management of Systemic Lupus Erythematosus*," Arthritis Research UK Topical Reviews, Chesterfield Derbyshire: Spring 2013.
43. Myasthenia Gravis Fact Sheet, Available: https://www.ninds.nih.gov/Disorders/Patient-Caregiver-Education/Fact-Sheets/Myasthenia-Gravis-Fact-Sheet, cited on: Dec 15, 2017.
44. Lindstrom, J. M., M. E. Seybold, V. A. Lennon, S. Whittingham, and D. D. Duane. "Antibody to acetylcholine receptor in myasthenia gravis Prevalence, clinical correlates, and diagnostic value." *Neurology* 51, no. 4, 1998: 933–933.
45. Liu, F., Z. Hou, L. Li, and R. Luo. "Constructing data mining model of five viscera correlation theory of Myasthenia Gravis based on rough set and association rules." In *Bioinformatics and Biomedicine Workshops (BIBMW), 2011 IEEE International Conference on*, pp. 778–783. IEEE, 2011.

46. https://www.niddk.nih.gov/health-information/endocrine-diseases/graves-disease (Accessed on Dec 16, 2017).
47. Smith, B., and R. Hall. "Thyroid-stimulating immunoglobulins in graves'disease." *The Lancet* 304, no. 7878, 1974: 427–430.
48. Weetman, A. P. "Graves' disease." *New England Journal of Medicine* 343, no. 17, 2000: 1236–1248.
49. Ozyilmaz, L., and T. Yildirim. "Diagnosis of thyroid disease using artificial neural network methods." In *Neural Information Processing, 2002. ICONIP'02. Proceedings of the 9th International Conference on*, 4, pp. 2033–2036. IEEE, 2002.
50. Orunesu, E., M. Bagnasco, C. Salmaso, V. Altrinetti, D. Bernasconi, P. Del Monte, G. Pesce, M. Marugo, and G. S. Mela. "Use of an artificial neural network to predict graves' disease outcome within 2 years of drug withdrawal." *European Journal of Clinical Investigation* 34, no. 3, 2004: 210–217.
51. Ioniţă, I., and L. Ioniţă. "Prediction of thyroid disease using data mining techniques." *BRAIN. Broad Research in Artificial Intelligence and Neuroscience* 7, no. 3, 2016: 115–124.
52. Wen, Z., and C. Fiocchi. "Inflammatory bowel disease: Autoimmune or immune-mediated pathogenesis?" *Journal of Immunology Research* 11, no. 3–4, 2004: 195–204.
53. Snook, J. "Are the inflammatory bowel diseases autoimmune disorders?" *Gut* 31, no. 9, 1990: 961.
54. Inflammatory Bowel Disease (IBD), Available on: https://www.ncbi.nlm.nih.gov/pubmedhealth/PMHT0022800/, Cited on Dec 16, 2017.
55. Cooper, J. G., and G. P. Purcell. "Data mining for correlations between diet and Crohn's disease activity." In *AMIA Annual Symposium Proceedings*, 2006, p. 897. American Medical Informatics Association, 2006.
56. Johnson, E. *Health care Data Mining Using In-database Analytics to Predict Diagnosis of Inflammatory Bowel Disease*. Master of Science Thesis: University of Washington, 2012.
57. Abouzahra, M., K. Sartipi, D. Armstrong, and J. Tan. "Integrating Data from EHRs to Enhance Clinical Decision-making: The Inflammatory Bowel Disease Case." In *Computer-Based Medical Systems (CBMS), 2014 IEEE 27th International Symposium on*, pp. 531–532. IEEE, 2014.
58. IFNB Multiple Sclerosis Study Group. "Interferon beta-1b is effective in relapsing-remitting multiple sclerosis I. Clinical results of a multicenter, randomized, double-blind, placebo-controlled trial." *Neurology* 43, no. 4, 1993: 655–655.
59. Multiple Sclerosis, Available at: https://medlineplus.gov/multiplesclerosis.html cited: 15-12-2017.
60. Lublin, F. D., and S. C. Reingold. "Defining the clinical course of multiple sclerosis results of an international survey." *Neurology* 46, no. 4, 1996: 907–911.
61. Poser, C. M., D. W. Paty, L. Scheinberg, W. Ian McDonald, F. A. Davis, G. C. Ebers, K. P. Johnson, W. A. Sibley, D. H. Silberberg, and W. W. Tourtellotte. "New diagnostic criteria for multiple sclerosis: Guidelines for research protocols." *Annals of Neurology* 13, no. 3, 1983: 227–231.
62. Bejarano, B., V. Segura, and P. Villoslada. "Data mining in multiple sclerosis: computational classifiers. Introduction and methods (Part I)."
63. Bejarano, B., V. Segura, and P. Villoslada. "Data mining in multiple sclerosis: computational classifiers. Introduction and methods (Part II)."
64. Guo, P., Q. Zhang, Z. Zhu, Z. Huang, and K. Li. "Mining gene expression data of multiple sclerosis." *PloS One* 9, no. 6, 2014: e100052.

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

8 Data Mining Techniques in Imaging of Embryogenesis

Diptesh Mahajan and Gaurav K. Verma

CONTENTS

8.1 Introduction 133
8.2 Imaging Techniques 134
8.2.1 Microscopy Techniques 134
8.2.2 Ultrasound Imaging 136
8.2.3 Computed Tomography/Micro Computed Tomography 136
8.2.4 Magnetic Resonance Imaging (MRI)/Micro-MRI/Functional-MRI/Magnetic Resonance Spectroscopy 137
8.3 Languages 138
8.3.1 CellML 138
8.3.2 SBML 139
8.3.3 Comparing CellML and SBML 140
8.4 Ongoing Projects 140
8.4.1 Bio-Emergences—Paris Saclay Institute of Neuroscience 140
8.4.2 CompuCell 3D-Indiana University 140
8.5 Future Prospects 141
References 141

8.1 INTRODUCTION

Over the last two decades, there has been an explosive production of data. This boom in datasets has observed a growth along two dimensions simultaneously—the number of fields and the number of cases (Feyyad 1996). The voluminous amount of structured and unstructured data generated has given rise to a new term—"big data." This term is not only defined by its volume, but its quality, diversity, and reliability (Sarkar 2016). In the field of biological sciences, the emergence of high throughput technology has led to the generation of huge molecular datasets at various levels of biological systems leading to the development of the "omics" era in life sciences (Li and Chen 2014). Researchers turn more often now towards big data to explore information at diverse scales, as it gives them a quantitative approach over the traditional qualitative approaches. The technological advancements in image analysis techniques coupled with the rapid price decline in high throughput instruments have

helped even the small-scale laboratories to generate big data. These developments have broadened the scope of fields like developmental biology, cancer biology, drug design, and many other subsequent fields.

In this chapter, we primarily focus on the science of embryology. Embryology deals with the study of prenatal developmental stages and congenital disorders occuring during embryonic development; it is also known as tetralogy. The classical embryology approach can detect the patterns of abnormal morphologic development but fails to recognize their cause (Carlson 2002). Contemporary molecular and imaging techniques have made it feasible to track the dynamic cellular or subcellular activities in the early stages of the embryo. This has helped us in identifying the reasons behind any inappropriate turn taken by the cell, which lead to birth defects. The advancements of optical design of microscopes have evolved from confocal microscopy and two-photon microscopy to light sheet microscopy, which can periodically generate cell-datasets of the developing embryo, one cell at a time. This exponential escalation of datasets has generated data well beyond petabytes in size (Marx 2013). With such huge datasets available, reliance on traditional analysis forms a bottleneck, hence leading to the development of the interdisciplinary field of bioinformatics. This field uses tools employing various data mining techniques to highlight new biological insights or hidden trends from the available data (Luscombe, Greenbaum, and Gerstein 2001). Data mining tools like SBML and CellML are used to elucidate the interaction of genes, proteins, and metabolites from the already existing data. This chapter critically reviews the techniques utilized to generate "big data" in embryology, and the languages used to mine the useful data with reference to some ongoing projects and platforms.

8.2 IMAGING TECHNIQUES

Contemporary imaging techniques provide an insight of the multidimensional and multiparameter activities at the cellular level. These usually capture the physical parameters including cell population, tissue property, and surface area (Eils and Athale 2003), as well as biological parameters, which include cell division, motility, and morphology changes (Pantazis and Supatto 2014).

Currently there are four techniques most widely used to track in vivo embryonic development.

8.2.1 Microscopy Techniques

i. *Confocal Fluorescence Microscopy*: The advent of fluorescence imaging has led to the emergence of 3D imaging of live embryo. This was a major leap over the conventional 2D static analysis of fixed embryos. Confocal microscopy uses point illumination, which is used to excite the biological fluorophores. Analysis of the emission spectrum of these fluorophores helps us in identifying the cell structure and its subsequent functions (Khairy and Keller 2011). This technique has been most widely used to study live morphogenesis (Pawley and Masters 2008) and to track cell lineage (Bao et al. 2006). Among vertebrates, the fluorescently labeled cells and

matrix motion can be tracked during the early embryonic development. Fish embryos provide a better understanding of embryogenesis as they are relatively smaller, and can be nurtured in nearly ambient environments and remain transparent throughout the early stages of cardiogenesis (Megason and Fraser 2007). Among vertebrates of higher order, the primitive streak formation, vasculogenesis, and gastrulation can be visualized prominently through confocal microscopy. Avian embryos allow for long-term optical access, as their germ disks are sleek and flat and are thus often used to study the early stages of embryogenesis in higher vertebrates (Yalcin et al. 2010). However, the process of embryonic development is highly sensitive to photodamage, and their development occurs at a scale beyond the range of confocal microscopy. The recent developments in multiphoton and light sheet microscopy have rectified these limitations to some extent.

ii. *Multiphoton Microscopy*: The extent to which the images can be analyzed is restricted by the optical properties of the embryonic tissues. These properties are highly dependent on the species and vary in time and space during development (Supatto et al. 2009). Unlike single photon illumination in confocal microscopy, multiphoton microscopy uses two photons, which substantially suppresses the background noise. It also uses nonlinear contrast mechanisms wherein the imaging is carried out without fluorescence probes, hence cells which cannot be photobleached can also be tracked. Multiphoton microscopy has been extensively used to reconstruct cell lineages and to study cell division patterns at a deeper tissue level, predominantly within drosophila (Fowlkes et al. 2008) and zebrafish (Hsieh et al. 2014) embryos. The recent advances include improved laser sources with increased excitation wavelength, which in-turn increases optical penetration and reduces photodamage (Andresen et al. 2009). However, the effectiveness of fluorescence and multiphoton microscopy techniques is limited to the earlier stages of embryonic development. A more sophisticated approach is required to analyze the later stages of embryogenesis using deeper tissue penetration.

iii. *Light-Sheet Microscopy*: Recording the images at multidimensional and multiparameter scales often compromises the speed performance. The aforementioned approaches use point illumination, hence can record images only one pixel at a time with pixel rates less than 1 MHz. To capture the rapid ongoing processes like the beating heart development, cilia motility, and fluid flow dynamics often a higher pixel rate is required. High-speed imaging also gives us a more detailed analysis of slower processes including cell proliferation, migration, and shape changes. Light-sheet microscopy is a promising solution for fast imaging of the live embryos. A distinctive advantage of this technique is it can carry out fast imaging with a wider angle of view and minimal photodamage. The whole-embryo imaging of the first 24 hours of the zebrafish has been beautifully captured by this technique, making it possible to track thousands of cells in parallel with high pixel rates and resolution (Keller et al. 2008). However, this technique also has its shortcomings; the high speed imaging results in uneven image

quality and limited imaging depth. These shortcomings result in relatively inaccurate intensity-based measurements and compromise deep tissue imaging.

The limitations of these microscopy techniques have given rise to alternate techniques like Bessel beam (Planchon et al. 2011) or multidirectional illumination (Huisken and Stainier 2007), which aim to improve either the depth of penetration or homogeneity of the fluorescence excitation and increase the spatial resolution throughout the field of view.

8.2.2 Ultrasound Imaging

This noninvasive technique sends short pulses of high frequency (ultrasonic) sound waves, which get reflected and/or transmitted when they encounter a tissue. Clinically, this technique is used for prenatal diagnosis of the embryo, which includes the development of follicles in the ovary and development of the fetus, the gestational sac, fetal parameters, and the placenta. It uses a probe containing one or more acoustic transducers. When the sound waves emitted hit a material with nonidentical acoustic impedance, a fraction of the wave gets reflected, which the probe detects as an echo. The time taken by the echo to travel to the probe is detected, allowing the depth of the tissue to be calculated. These calculated distances of the tissue from the probe are used to generate a series of 2D images in three different planes. These images are then superimposed by the computer to construct a 3D image (Jensen 2007). This technique provides real-time information and can provide visualization of the later fetal stages, which helps to detect abnormal morphology (Yu et al. 2008) and hemodynamic changes (Oosterbaan et al. 2009). Abnormal morphology may include early diagnosis of a trophoblastic disease, interruption of bladder line, genetic abnormality, and other subsequent anomalies. The hemodynamic changes are measured by Doppler ultrasound, which detects the change in blood flow and blood pressure by rebounding ultrasounds off circulating red blood cells. The rapid evolution of ultrasound technology has made it possible for us to trace information from organ to molecular level and this is primarily due to the development of microbubbles. These bubbles act as intravascular tracers and can be target specific if attached with suitable ligands (Thomasson et al. 2004). The past few years have seen the increased use of nanoparticles for extravascular tracing, as they can easily migrate to regions of vascular injury, or to regions where vascular permeability is aberrantly high. The images generated by nanoparticle tracers have a lower signal-to-noise ratio compared to microbubble tracers, and therefore have an improved image quality (Kherlopian et al. 2008).

8.2.3 Computed Tomography/Micro Computed Tomography

The basic principle behind this technique is the differential absorption of X-rays by different components like fat, bone, air, and water. Thus, depending on the tissue present, the amount of X-rays absorbed will differ. This difference in absorption is detected by the computer, which generates cross-sectional images or "slices" of the

part exposed. These slices are called tomographic images and are more informative than conventional X-rays. The tomographic slices generated by the computer are then stacked digitally to generate 3D images in order to easily identify and locate the tissue structures along with any possible abnormality present, such as an intrinsic tumor (Kalender 2006). The soft tissue contrast is relatively low for tumors, but this can be enhanced with iodinated contrast agents, which can detect tumors more easily.

For the study of animal specimens, micro-CT is used and emits higher energy X-rays than CT. The increase in energy helps in improving the resolution of the tomographic slices, but the increase in radiation also leads to adverse health effects. Micro-CT systems have low cost of maintenance and a brief scan time when compared to the other imaging techniques (Kim et al. 2011). Although micro-CT imaging has the highest possible theoretical resolution, the soft tissues poorly attenuate X-rays; therefore, most of the embryonic development imaging is performed on fixed tissues stained with molecules having high atomic weight to enhance X-ray attenuation (Metscher 2009). The lethal radiations of micro-CT often lead to abnormalities in developing embryos, making it difficult for scientists to study embryonic developments afterwards. Recent developments use gold nanoparticles as contrast agents, which increase the residence time and reduces cytotoxic effects (Murphy et al. 2008).

Both CT/micro-CT have high spatial resolutions, which can allow for visualization of the fine anatomical details; however, the radiation dose is a concern and cannot be neglected. Thus, they have limited applications in human studies.

8.2.4 Magnetic Resonance Imaging (MRI)/Micro-MRI/Functional-MRI/Magnetic Resonance Spectroscopy

This scan is performed by placing the subject under the influence of a strong magnetic field and radio waves, which generate images that cannot be visualized with ultrasound or CT-scans. The magnetic field aligns the hydrogen protons present in water molecules, which have a designated spin. A radio frequency (RF) pulse is emitted simultaneously, which the hydrogen proton absorbs causing its spin to flip through a process known as precession. After the RF pulse is turned off, the protons return to their native spin and realign with the applied magnetic field. While returning to their native state, the protons emit radio signals detected and processed by the scanners. The varying spin density among different tissues helps in generating an excellent tissue contrast (Hornak 2008).

Micro-MRI also follows a similar principle but employs a much stronger magnetic field. An increase in the magnetic field provides images with a better contrast, but may lead to peripheral nerve stimulation of the subject.

Functional-MRI is used to study the brain activity in response to a predetermined stimulus. The stimulus initiates a response from certain areas of the brain, leading to an increased metabolism in that part. The increased metabolism in turn initiates an increase in blood flow. Thus, the concentration of oxygenated blood increases in that region resulting in an imbalance in oxygenated and deoxygenated blood. This imbalance leads to a change in image contrast, which can be detected by the fMRI (Parrish et al. 2000).

Magnetic resonance spectroscopy is an emerging imaging technique combining the ability of nuclear magnetic resonance (NMR)—to detect biochemical moieties—and the ability of MRI to isolate individual 3D pixels. This is mainly carried out by emitting phase modulated RF pulses, which eliminates the possibility of contaminating nearby voxels. A combination of NMR with MRI helps in generating anatomical and biochemical information in parallel (Arias-Mendoza and Brown 2004).

The key advantage of MRI over other imaging techniques is its exceptional tissue contrast due to high signal-to-noise ratio and lack of ionizing radiation.

8.3 LANGUAGES

The imaging techniques discussed above generate enormous volumes of data. It is manually impossible to draw conclusions from such huge sample sizes. Thus, we need the help of computational analysis to interpret the required biological data. Computational technologies are essential to automate the processing, quantification, and analysis of high-information content biomedical images (Raza 2012). The first step is to store these datasets in structured formats, in a way that simplifies sharing and accessibility. The next step includes integrating and interpreting these stored datasets at genomic, proteomic, metabolomic, cellular, or whole organism level (Zaki et al. 2003).

To summarize, a system biologist faces a number of problems without a domain-specific language (Hucka et al. 2004):

i. In order to work with data obtained from multiple simulation/analysis tools, they need to re-encode the models received from different sources. This makes the process both error prone and time consuming.
ii. The development of new software tools has rendered the old systems obsolete. This has resulted in the loss of a number of usable models.
iii. Models developed by authors are usually developed under different modeling environments. The definitions to obtain these models are often too complex and cannot be retrieved easily and reused.

Thus, the development of computer-oriented, domain-specific languages, have become a necessity to study the dynamic interaction of genes, proteins, or metabolites at different levels under different modeling environments.

Currently two languages, SBML and CellML, are most widely used to interpret these biochemical interactions. Both these languages are based on XML. The reason prompting system biologists to choose XML as a basis is its user-friendly nature. XML is both human and machine-readable and this feature has encouraged its use in the field of bioinformatics (Achard et al. 2001).

8.3.1 CellML

The components defined in CellML (Figure 8.1) are linked together by connections to generate a model (Cuellar et al. 2015). The groups indicate the existence of a logical/physical group and metadata provides the basis for the model.

FIGURE 8.1 Components of a CellML model.

8.3.2 SBML

SBML allows models to represent arbitrary complexity. All the components constituting the model are described using a specific type of data object, which categorizes the relevant information (Hucka et al. 2015). The latest level of a SBML model definition consists of lists of components, as described in Figure 8.2.

FIGURE 8.2 Components of a SBML model.

8.3.3 Comparing CellML and SBML

Both these languages have similar goals, but digress with respect to their core abilities. CellML was primarily designed to create models, and hence can illustrate the mathematics of the cellular models in a more powerful and general way than SBML. SBML, on the other hand, was primarily designed to exchange datasets associated with pathways and reaction models, and hence can represent the models of biochemical reaction networks more profoundly than CellML. Although differing in core competencies, the development teams of both languages have a close working relationship, which ensures a synchronous development between languages.

8.4 ONGOING PROJECTS

8.4.1 Bio-Emergences—Paris Saclay Institute of Neuroscience

This project aims at pioneering novel techniques and strategies to improve the multistep approach of quantitatively understanding the morphogenesis of organisms, both under normal and pathological conditions. The architecture of the workflow can be broadly categorized into three steps

i. *Multiscale-multimodal observation*: This step involves in vivo/in toto imaging by techniques such as illumination and multiphoton microscopy.
ii. *Quantification*: Also termed phenomenological reconstruction, this step uses an algorithmic method to extract the relevant quantitative information.
iii. *Multilevel theoretical modeling*: The quantitative parameters obtained from the previous step are used to set parameters of the theoretical model.

This work also features a custom-made software "Mov-IT," which validates, proofreads, and analyses the reconstructed model. It helps in reconstructing accurate cell lineages of developing organisms. The in silico experimental embryology has been successfully demonstrated on zebrafish, ascidian, and sea urchin embryos. (Faure et al. 2016).

8.4.2 CompuCell 3D-Indiana University

Compucell 3D is a flexible modeling environment aimed at constructing in silico simulations of multiscale, multicellular biological processes, which include cancer cell proliferation, embryogenesis, tissue engineering, and other subsequent fields. Primarily headed towards modeling of various morphological patterns, this project integrates various mathematical models developed into a syntax based on XML. This has helped developers to build models with ease without recoding them from scratch. The models developed can capture cell growth, division, apoptosis, and physical parameters like surface area and volume. Different partial differential equation models are employed based on the behavior of the cells.

One key feature of this platform is the graphical user interface CompuCell player, which helps in 3D visualization and supports the feature of switching to 2D cross sections in each dimension along with ability to alternate the fields. To improve

performance, the player can be run in silent mode to generate a greater number of images between two steps.

In summary, this problem-solving environment has the following features (Swat et al. 2012)

i. Can run simulations in 3D.
ii. Designed for high performance and high memory consumption to analyze sizeable numbers of cells in parallel.
iii. Flexible and extensible allowing incorporation of new cell behaviors at a higher level/model.

8.5 FUTURE PROSPECTS

The reason behind a substantial number of cases showing abnormal embryonic development still remains unknown. With such huge data volumes increasing, the science of embryogenesis needs to shift its focus from in vivo to in silico experiments. The cell developmental models of various cells can now be reconstructed using computer simulations with the aid of applied mathematics. Thus, critical data mining approaches followed by in silico resynthesis of embryos will enable scientists to accurately unravel the factors causing subtle changes in cell behavior. This will eventually help them uncover the causes behind abnormal development (Knudsen et al. 2017).

REFERENCES

Achard, F., G. Vaysseix, and E. Barillot. 2001. "XML, Bioinformatics and Data Integration." *Bioinformatics* 17 (2): 115–25. doi:10.1093/bioinformatics/17.2.115.

Andresen, V., S. Alexander, W. M. Heupel, M. Hirschberg, R. M. Hoffman, and P. Friedl. 2009. "Infrared Multiphoton Microscopy: Subcellular-Resolved Deep Tissue Imaging." *Curr Opin Biotechnol* 20 (1): 54–62. doi:10.1016/j.copbio.2009.02.008.

Arias-Mendoza, F., and T. R. Brown. 2004. "In Vivo Measurement of Phosphorous Markers of Disease." *Disease Markers* 19 (2–3): 49–68.

Bao, Z., J. I. Murray, T. Boyle, S. L. Ooi, M. J. Sandel, and R. H. Waterston. 2006. "Automated Cell Lineage Tracing in Caenorhabditis Elegans." *Proceedings of the National Academy of Sciences* 103 (8): 2707–12. doi:10.1073/pnas.0511111103.

Carlson, B. M. 2002. "Embryology in the Medical Curriculum." *Anatomical Record* 269 (2): 89–98. doi:10.1002/ar.10075.

Cuellar, A., W. Hedley, M. Nelson, C. Lloyd, M. Halstead, D. Bullivant, D. Nickerson, P. Hunter, and P. Nielsen. 2015. "The CellML 1.1 Specification." *Journal of Integrative Bioinformatics* 12 (2): 259. doi:10.2390/biecoll-jib-2015-259.

Eils, R., and C. Athale. 2003. "Computational Imaging in Cell Biology." *The Journal of Cell Biology* 161 (3): 477–81. doi:10.1083/jcb.200302097.

Faure, E., T. Savy, B. Rizzi, C. Melani, O. Stašová, D. Fabrèges, R. Špir et al. 2016. "A Workflow to Process 3D+time Microscopy Images of Developing Organisms and Reconstruct Their Cell Lineage." *Nature Communications* 7. doi:10.1038/ncomms9674.

Feyyad, U. M. 1996. "Data Mining and Knowledge Discovery: Making Sense out of Data." *Ieee Expert Intelligent Systems And Their Applications* 11 (5): 20–25. doi:10.1109/64.539013.

Fowlkes, C. C., C. L. Luengo Hendriks, S. V. E. Keränen, G. H. Weber, O. Rübel, M. Y. Huang, S. Chatoor et al. 2008. "A Quantitative Spatiotemporal Atlas of Gene Expression in the Drosophila Blastoderm." *Cell* 133 (2): 364–74. doi:10.1016/j.cell.2008.01.053.

Hornak, J. P. 2008. "The Basics of MRI." *Biomedical Engineering* 24 (2003): 2–6.

Hsieh, C.-S., C.-Y. Ko, S.-Y. Chen, T.-M. Liu, J.-S. Wu, C.-H. Hu, and C.-K. Sun. 2014. "In Vivo Long-Term Continuous Observation of Gene Expression in Zebrafish Embryo Nerve Systems by Using Harmonic Generation Microscopy and Morphant Technology." *Journal of Biomedical Optics* 13 (6): 64041. doi:10.1117/1.3050423.

Hucka, M., A. Finney, B. J. Bornstein, S. M. Keating, B. E. Shapiro, J. Matthews, B. L. Kovitz et al. 2004. "Evolving a Lingua Franca and Associated Software Infrastructure for Computational Systems Biology: The Systems Biology Markup Language (SBML) Project." *Systems Biology* 1 (1): 41–53. doi:10.1049/sb.

Hucka, M., F. T. Bergmann, S. Hoops, S. M. Keating, S. Sahle, J. C. Schaff, L. P. Smith, and D. J. Wilkinson. 2015. "The Systems Biology Markup Language (SBML): Language Specification for Level 3 Version 1 Core." *Journal of Integrative Bioinformatics* 12 (2): 266. doi:10.2390/biecoll-jib-2015-266.

Huisken, J., and D. Y. R. Stainier. 2007. "Even Fluorescence Excitation by Multidirectional Selective Plane Illumination Microscopy (mSPIM)." *Optics Letters* 32 (17): 2608. doi:10.1364/OL.32.002608.

Jensen, J. A. 2007. "Medical Ultrasound Imaging." *Progress in Biophysics and Molecular Biology*. doi:10.1016/j.pbiomolbio.2006.07.025.

Kalender, W. A. 2006. "X-Ray Computed Tomography." *Physics in Medicine and Biology*. doi:10.1088/0031-9155/51/13/R03.

Keller, P. J., A. D. Schmidt, J. Wittbrodt, and E. H. K. Stelzer. 2008. "Reconstruction of Zebrafish Early Embryonic Development by Scanned Light Sheet Microscopy." *Science* 322 (5904): 1065–69. doi:10.1126/science.1162493.

Khairy, K., and P. J. Keller. 2011. "Reconstructing Embryonic Development." *Genesis*. doi:10.1002/dvg.20698.

Kherlopian, A. R., T. Song, Q. Duan, M. A. Neimark, M. J. Po, J. K. Gohagan, and A. F. Laine. 2008. "A Review of Imaging Techniques for Systems Biology." *BMC Systems Biology*. doi:10.1186/1752-0509-2-74.

Kim, J. S., J. Min, A. K. Recknagel, M. Riccio, and J. T. Butcher. 2011. "Quantitative Three-Dimensional Analysis of Embryonic Chick Morphogenesis Via Microcomputed Tomography." *Anatomical Record* 294 (1): 1–10. doi:10.1002/ar.21276.

Knudsen, T. B., B. Klieforth, and W. Slikker Jr. 2017. "Programming Microphysiological Systems for Children's Health Protection." *Experimental Biology and Medicine* 242, no. 16 (2017): 1586–92. doi:10.1177/1535370217717697.

Li, Y., and L. Chen. 2014. "Big Biological Data: Challenges and Opportunities." *Genomics, Proteomics & Bioinformatics* 12 (5). Beijing Institute of Genomics, Chinese Academy of Sciences and Genetics Society of China: 187–89. doi:10.1016/j.gpb.2014.10.001.

Luscombe, N. M., D. Greenbaum, and M. Gerstein. 2001. "Review What Is BioinformaticsAn." *Gene Expression* 40 (4): 83–100. doi:10.1053/j.ro.2009.03.010.

Marx, V. 2013. "The Big Challenges of Big Data." *Nature* 498 (7453): 255–60. doi:10.1038/498255a.

Megason, S. G., and S. E. Fraser. 2007. "Imaging in Systems Biology." *Cell*. doi:10.1016/j.cell.2007.08.031.

Metscher, B. D. 2009. "Micro CT for Comparative Morphology: Simple Staining Methods Allow High-Contrast 3D Imaging of Diverse Non-Mineralized Animal Tissues." *BMC Physiology* 9 (1). doi:10.1186/1472-6793-9-11.

Murphy, C. J., A. M. Gole, J. W. Stone, P. N. Sisco, A. M. Alkilany, E. C. Goldsmith, and S. C. Baxter. 2008. "Gold Nanoparticles in Biology: Beyond Toxicity to Cellular Imaging." *Accounts of Chemical Research* 41 (12): 1721–30. doi:10.1021/ar800035u.

Oosterbaan, A. M., N. T. C. Ursem, P. C. Struijk, J. G. Bosch, A. F. W. Van Der Steen, and E. A. P. Steegers. 2009. "Doppler Flow Velocity Waveforms in the Embryonic Chicken Heart at Developmental Stages Corresponding to 5–8 Weeks of Human Gestation." *Ultrasound in Obstetrics and Gynecology* 33 (6): 638–44. doi:10.1002/uog.6362.

Pantazis, P., and W. Supatto. 2014. "Advances in Whole-Embryo Imaging: A Quantitative Transition Is Underway." *Nature Reviews Molecular Cell Biology*. doi:10.1038/nrm3786.

Parrish, T. B., D. R. Gitelman, K. S. LaBar, and M. Marsel Mesulam. 2000. "Impact of Signal-to-Noise on Functional MRI." *Magnetic Resonance in Medicine* 44 (6): 925–32. doi:10.1002/1522-2594(200012)44:6<925::AID-MRM14>3.0.CO;2-M.

Pawley, J. B., and B. R. Masters. 2008. Handbook of Biological Confocal Microscopy. *Journal of Biomedical Optics*. Vol. 13. doi:10.1117/1.2911629.

Planchon, T. A., L. Gao, D. E. Milkie, M. W. Davidson, J. A. Galbraith, C. G. Galbraith, and E. Betzig. 2011. "Rapid Three-Dimensional Isotropic Imaging of Living Cells Using Bessel Beam Plane Illumination." *Nature Methods* 8 (5): 417–23. doi:10.1038/nmeth.1586.

Raza, K. 2012. "Application of Data Mining in Bioinformatics." *Indian Journal of Computer Science and Engineering* 1 (2): 114–18. doi:10.4010/2016.1759.

Sarkar, R. R. 2016. "The Big Data Deluge in Biology: Challenges and Solutions." *Journal of Informatics and Data Mining* 1 (2): 1–3. doi:10.21767/2472-1956.100014.

Supatto, W., A. McMahon, S. E. Fraser, and A. Stathopoulos. 2009. "Quantitative Imaging of Collective Cell Migration during Drosophila Gastrulation: Multiphoton Microscopy and Computational Analysis." *Nature Protocols* 4 (10): 1397–1412. doi:10.1038/nprot.2009.130.

Swat, M. H., G. L. Thomas, J. M. Belmonte, A. Shirinifard, D. Hmeljak, and J. A. Glazier. 2012. "Multi-Scale Modeling of Tissues Using CompuCell3D." *Methods in Cell Biology* 110: 325–66. doi:10.1016/B978-0-12-388403-9.00013-8.

Thomasson, D. M., A. Gharib, and K. C. P. Li. 2004. "A Primer on Molecular Biology for Imagers: VIII. Equipment for Imaging Molecular Processes." *Academic Radiology*. doi:10.1016/j.acra.2004.07.008.

Yalcin, H. C., A. Shekhar, A. A. Rane, and J. T. Butcher. 2010. "An ex-Ovo Chicken Embryo Culture System Suitable for Imaging and Microsurgery Applications." *Journal of Visualized Experiments,* no. 44. doi:10.3791/2154.

Yu, Q., L. Leatherbury, X. Tian, and C. W. Lo. 2008. "Cardiovascular Assessment of Fetal Mice by In Utero Echocardiography." *Ultrasound in Medicine and Biology* 34 (5): 741–52. doi:10.1016/j.ultrasmedbio.2007.11.001.

Zaki, M. J., J. T. L. Wang, and H. T. T. Toivonen. 2003. "Data Mining in Bioinformatics." *ACM SIGKDD Explorations Newsletter* 5 (2): 198. doi:10.1145/980972.981006.

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

9 Machine Learning Approach to Overcome the Challenges in Theranostics

A Review

Bishwambhar Mishra, Sayak Mitra, Karthikeya Srinivasa Varma Gottimukkala, and Shampa Sen

CONTENTS

9.1 Introduction to Theranostics 146
9.2 Potential Targets for Theranostic Agents in Nuclear Medicine 148
9.2.1 Metabolic Active Radiopharmaceuticals 150
9.2.2 Specifically Binding Radiopharmaceuticals 151
9.2.2.1 Theranostic Application of Y-90 152
9.2.2.2 Iodine—The Classic Theranostic Pair 152
9.2.2.3 Ga-68, F-18/(Y-90), and Lu-177/Ac-225—Key Theranostic Players 152
9.2.2.4 Zr-89, Y-86/Y-90 152
9.3 Application of Theranostics 153
9.3.1 Neuroendocrine Tumors 153
9.3.2 PSMA—Theranostics 154
9.3.3 CXCR4—Theranostics 155
9.4 Major Problems 155
9.5 Transitioning into Clinical Reality—A Machine Leaning Approach 156
9.5.1 Positron Emission Tomography (PET)—Computed Tomography (PET/CT) 156
9.5.2 Single Photon Emission Computed Tomography—Computed Tomography (SPECT/CT) 157
9.5.3 Quantitative SPECT 157
9.5.3.1 Camera Normalization 158
9.5.3.2 Scatter Corrections 158
9.5.3.3 Camera Dead-Time 158
9.5.3.4 Bremsstrahlung Radiation 158
9.5.3.5 Quantitative Reconstructions 159

9.5.3.6 Quantitative Preclinical Imaging 159
9.5.3.7 Patient Specific Dosimetry/Dose Calculations 159
9.5.3.8 Image-Based Computational Tools..... 160
9.6 Conclusion 161
References..... 161

9.1 INTRODUCTION TO THERANOSTICS

Modern medicine is currently undergoing a paradigm shift from conventional disease treatments based on the diagnosis of a generalized disease state to a more personalized—and customized—treatment model based on molecular-level diagnoses. This involves the use of novel biosensors able to precisely extract disease-related information from complex biological systems. Moreover, with the recent progress in chemical biology, materials science, and synthetic biology it has become possible to simultaneously conduct diagnosis and targeted therapy (theranostics/theragnosis) by directly connecting the readout of a biosensor to a therapeutic output. These advances pave the way for more advanced and better-personalized treatment for intractable diseases, while producing fewer side effects. In this review, we describe the recent advances in the development of cutting-edge theranostic agents, which contain both diagnostic and therapeutic functions in a single integrated system.

By comparing the advantages and disadvantages of each modality, we discuss the future challenges and prospects of developing ideal theranostic agents for the next generation of personalized medicine. "Right drug, right patient, right moment, right space, and right dose" is a key concept of current personalized medicine to maximize the efficacy of treatment while minimizing side effects (Hood 2003). The formulations of theranostic agents are highlighted in Figure 9.1. A robust new paradigm for advancing treatments against cancers and other diseases is provided by the integration of multiple moieties such as imaging and therapeutics into a single unit. (Shah and Shah 2012).

FIGURE 9.1 Construction of the theranostics in a single integrated system.

In this context, theranostics have emerged as a new concept in next-generation medicine to combine simultaneous diagnosis and targeted therapy. Cutting-edge theranostic agents are designed to contain both diagnostic and therapeutic functions in a single integrated system capable of sensing the endogenous biomarkers of a host organism, and therefore can exert therapeutic functions with a minimal delay (Crawley and Thompson 2014). For example, it has become possible to entrap imaging agents and therapeutic drugs in a single integrated molecule, or nanoparticle, using the advances in material science and chemical biology; the drugs and agents are released when and where the system senses the presence of the disease.

The basic principle of theranostics has been in practice for decades, dating back to the 1940s, where the use of radioiodine was used to image and treat thyroid cancers. More recently, advances in molecular biology, genetics, and proteomics have provided a wealth of biochemical and functional information for cancers, which also includes the identification of many cell surface receptors that are overexpressed in tumor cells (Kelkar and Reineke 2011). Theranostics take the advantages of these cells surface receptors to visualize the site of diseases. More over, it is helpful for defining presence or absence of a molecular target, and delivering the cytotoxic payloads directly to tumor sites (Jeelani et al. 2014).

Depending on the application, molecular targeting agents can be combined with a wide variety of imaging agents including radionuclides (single photon emission computed tomography [SPECT], and positron emission tomography [PET]), optical probes (fluorescence), or metal chelates (MRI). The presence of a molecular target, as well as its distribution within a patient, can be provided to the clinicians by combining molecular imaging with a theranostic approach. Furthermore, theranostics has the potential to facilitate patient screening, guide clinical trial enrollment, and monitor the efficacy of therapeutics. A well-studied example of a theranostic target is the human epidermal growth factor receptor, Her-2. Her-2 is overexpressed in aggressive breast cancers with poor prognosis. Breast cancer can be identified in patients using a diagnostic test which detects the large amounts of Her-2, that is, guiding the treatment with a therapeutic monoclonal antibody against this target (Ferber et al. 2014).

The field of theranostics is rapidly facilitating the shift from "trial and error" medicine to personalized medicine, and holds great promise for improved patient outcomes. From the clinician's perspective, theranostics provide a valuable tool for identifying and selecting patients with a particular molecular phenotype indicative of positive response to treatment. This approach has the potential to help improve drug efficacy by understanding which patients might benefit the most from treatment (Jeelani et al. 2014). The working model of theranostics in the human system is illustrated by Figure 9.2.

The precise cell specific targeting properties of a theranostic may help to improve the safety profile of a drug and minimize off-target effects to normal tissues—often seen with various chemotherapies. Finally, from an economic perspective theranostics can lead to more cost-effective and efficient drug programs, guiding preclinical drug development—or clinical trial eligibility—to help maximize the probability of successful outcomes. The demanding aspects of theranostics, in terms of research, are highlighted in the published papers shown in Figure 9.3 (https://www.ncbi.nlm.nih.gov/pubmed/).

FIGURE 9.2 Schematic overview of theranostics as a working model in the targeted organ of human system.

FIGURE 9.3 Number of published papers in theranostics research until the present, according to PubMed.

9.2 POTENTIAL TARGETS FOR THERANOSTIC AGENTS IN NUCLEAR MEDICINE

For nuclear imaging, either gamma emitters or positron emitters are mandatorily required for detection and visualization through a gamma camera, SPECT, or PET. These noninvasive molecular imaging techniques are used for the diagnosis and monitoring of the subsequent therapy efficiency. The radionuclides most commonly used for SPECT are Tc-99m, In-111, and I-123/125 (Nayak and Brechbiel 2011). Ga-68, F-18, I-124, Zr-89, Y-86, and Cu-64 (Table 9.1) are the most commonly used radionuclides for PET.

TABLE 9.1
Applications of Theranostic Agents in Nuclear Medicine

S.L No	Diagnostic Agents	Therapeutic Agents	Mechanism of Reaction	Application Examples	References
1	Tc-99 m(ϒ) $t_{1/2} = 6.06$ h	Re-188(26% ß, 15% ϒ) $t_{1/2} = 17$ h	Chelator direct labelling (sulfhydryl group)	Antibodies metal particles diphosponates	Gustafsson et al. (2012) Häfeli et al. (2001) Uematsu (2005) García-Garayoa et al. (2007) Lange et al. (2016)
2	Cu-64(19% ß+ /38% ß− /43% ϒ) $t_{1/2} = 12.7$ h	Cu-67(ß−) $t_{1/2} = 61.8$ h Lu-177(ß−/ϒ) $t_{1/2} = 6.7$ d	Chelator direct labeling (nanoformulations)	Liposomes nanoparticles antibodies	Enrique et al. (2015) Szymański et al. (2012) Stockhofe et al. (2014) Loon et al. (2017)
3	Zr-89(23% ß+) $t_{1/2} = 78.4$ h	Y-90(>99% ß−) $t_{1/2} = 64$ h	Chelator direct labeling (nanoparticles)	Antibodies (i.e., Zevalin) nanoparticles	Perk et al. (2006) Kramer et al. (2017)
4	Y-86(32% ß+) $t_{1/2} = 14.7$ h	Y-90(>99% ß−) $t_{1/2} = 64$ h	Chelator	Small molecules (e.g., DOTATOC, PSMA inhibitors)	Rösch et al. (2017) Banerjee et al. (2015)
5	F-18(97% ß+) $t_{1/2} = 110$ min	I-131(90% ß−;10% ϒ) $t_{1/2} = 8$ d	Nucleophilic substitution electrophilic attack	ICF-15002	Rbah-Vidal et al. (2017)
6	In-111(>99.9%EC $t_{1/2} = 67.9$ h	In-111 (>99.9% EC; Auger electron emitter)	Chelator (DTPA)	PRRT	Gotthardt et al. (2006)
7	Ga-68(89% ß+) $t_{1/2} = 68$ min In-111(>99.9%EC) $t_{1/2} = 67.9$ h	Lu-177(ß−/ϒ) $t_{1/2} = 6.7$ d Y-90(>99% ß−) $t_{1/2} = 64$ h	Chelator (introduced via bifunctional linker)	PSMA-617	Gotthardt et al. (2006)

FIGURE 9.4 Number of published papers in nuclear medicine with theranostics until the present, according to Pubmed.

Therapeutic tracers are most likely radiolabeled with beta minus particles (ß–), which have a range between 1 and 10 mm with energies between 0.1 and 1 MeV. The most commonly used radionuclides are I-131, Lu-177, and Y-90. Cu-67 is acquiring more attention in preclinical studies, as it is chemically identical and exchangeable with its positron-emitting isotope Cu-64. I-131 is characterized by a 90% beta minus (ß−) emission with a mean tissue penetration of 0.4 mm and 10% gamma emission, which can be imaged with SPECT or a gamma camera. The mean tissue penetration of Y-90 in its pure beta minus emitter is 2.5 mm. The high penetration depth results in a crossfire effect resulting in damage to healthy organs depending on the tracers biodistribution (Morgenroth et al. 2017). In peptide receptor radionuclide therapy (PRRT), Y-90 was responsible for crossfire effect which could be compensated by Lu-177. Lu-177 has two advantages compared to Y-90: the mean tissue penetration of Lu-177 is 0.6 mm and it emits gamma rays, which allows for therapy monitoring and dosimetry evaluation. To apply the theranostic strategy to commonly used Tc-99m labeled probes, the generator nuclide Re-188 (W-188/Re-188 generator), which possess comparable coordination chemistry, is already being used in preclinical and early phase clinical studies for therapeutic application (Dash et al. 2013). Re-188 emits beta minus particles (25.6%) and gamma rays for imaging (15.1%). The demanding aspects is illustrated by published papers in Figure 9.4 (https://www.ncbi.nlm.nih.gov/pubmed/).

9.2.1 Metabolic Active Radiopharmaceuticals

Inorganic and organic materials are used to design nanotheranostics. Metal-based nanoparticles offer a route for direct radiolabeling. They are doped with radioisotopes and are activated through neutron reactions or proton beams. As shown for super paramagnetic iron oxide nanoparticles (SPIONs), radioactivity can be taken up

either via specific trapping or via ion exchange and simple equilibrium. SPIONs are being radiolabeled with Ra-223 and show promising stability of the radioactive particles in vitro (Broda et al. 2016). As an example, for activation through neutron reactions, gold nanoparticles can be activated and intrinsically labeled with β emitting Au-198 (t 1/2 = 2.7 d; energy <1 MeV). The particles are attracted due to their optical properties, as well as their size-dependent cytotoxicity, which can be enhanced for therapeutic use through radiolabeling (Same et al. 2016). Surface modification has allowed the process for targeting ligands or additional radiolabeling via specific coordination chemistry, and for optimization of pharmacokinetics and biocompatibility (Ernsting et al. 2013). Antibodies such as cetuximab can also target ligands. The antibodies can be radiolabelled with I-131 and be grafted to gold nanoparticles for theranostic application. Despite surface modifications, the quantum dots can be doped with radiometals like Zr-89 or Cu-64 by the process of ion exchange (Cheng et al. 2016).

9.2.2 Specifically Binding Radiopharmaceuticals

The basic architecture of theranostics is as follows (Figure 9.5):

- A targeting agent, which directs the theranostic agent to a molecular target on the surface of a cell or tumor.
- A chelate in the form of an imaging agent (which enables visualization of the target), or a therapeutic drug (for delivery of treatment to the target site).
- A linker which serves as a connection between the two entities.

The various applications of nuclear theranostics have been summarized in Table 9.1.

FIGURE 9.5 Schematic representation of radiopharmaceuticals used in target diseased cells.

9.2.2.1 Theranostic Application of Y-90

Y-90 is a rare-earth lanthanide, which emits high energy ß-particles and is used in radiotherapy. In PRRT, it was almost completely displaced by Lu-177, but is still commonly used in radio-immunotherapy and radio-immobilization therapy with Y-90 microspheres. While radio-immobilization therapy is usually planned with Tc-99m labeled macroaggregated albumin (MAA), the surrogate cannot predict side effects. Therefore, Y-90 has gained attention as a theranostic tool itself. Imaging of Y-90 can be performed via PET due to its internal pair production with detectable peaks of 511 keV (Goedicke et al. 2013), as well as detection of radiation with gamma cameras. Y-90 is also known to produce Cerenkov luminescence, which can be visualized with optical Cerenkov luminescence imaging (Wright et al. 2015).

9.2.2.2 Iodine—The Classic Theranostic Pair

Years before the term theranostic became viral in the medical field, the theranostic approach was already common practice in nuclear medicine. Radioiodine theranostics is a typical example regarding the concept of personalized medicine and is used extensively for the treatment of thyroid cancer. At the beginning, I-131 was used as a therapeutic and diagnostic tool, but due to its emission of ß-particles and high-radiation dose it cannot be considered as an ideal diagnostic agent. I-123, on the other hand, has a more efficient interaction with sodium iodide crystals, while showing no β-emission and a lower energy, which results in a more efficient collimation. Since I-123 provided radiation protection and superior images, it has been increasingly used as the isotope of choice for diagnostic imaging as a surrogate for I-131. Nonetheless, I-124 with its longer half-life and its suitability for PET has become an important tool in clinical and preclinical imaging (Silberstein 2012). Apart from thyroid cancer imaging, the iodine theranostic pairs are used for radio-labeling of targeted tracers found in different tumor types. Meta-iodobenzylguanidine (MIBG) is being used as a theranostic tool for endocrine tumors (Vöö et al. 2011; Sisson et al. 2012). Radiolabeling with I-124 or I-131 was also performed for the small molecule inhibitor of prostate-specific membrane antigen (PSMA) (Zechmann et al. 2014; Afshar-Oromieh et al. 2017).

9.2.2.3 Ga-68, F-18/(Y-90), and Lu-177/Ac-225—Key Theranostic Players

The key theranostic players in nuclear medicine are the small molecules with covalently attached chelators for labeling with radiometals. The nuclide of Ga-68 is used as positron emitter in diagnosis and visualization of receptors or antigen expression. Peptide receptor radionuclide therapy (PRRT) is a theranostic approach mainly targets somatostatin receptor subtypes 2 (SSTR2) for the detection of neuroendocrine tumors (NET). Peptides are covalently bound to the chelator DOTA (1,4,7,10-Tetraazacyclododecane-1,4,7,10-tetraacetic acid), which enables the coordination of Ga-68 for PET imaging and Lu-177 (and Y-90) for radiotherapy. The ability of all these derivatives to address somatostatin receptor expression is comparable to neuroendocrine tumors (Blankenberg and Strauss 2012; Maecke and Reubi 2017).

9.2.2.4 Zr-89, Y-86/Y-90

Zr-89 has a physical half-life compatible with the biological half-life and pharmacokinetics of immune conjugates. Additionally, Zr-89 has a comparatively

low positron energy of 356 keV, which results in high resolution PET images, and radiolabeling can be performed in a straightforward manner using DFO as a chelator (Reiner et al. 2015). For improved therapy selection and planning, short-lived radioisotopes might not be able to provide the information needed for efficient therapy. Long-lived PET isotopes can resolve the issue and reveal detailed information of biodistribution and radiation dose on healthy organs. Similarly to the copper isotopes, the half-life of several days is ideal for in vivo imaging of macromolecules and nanosized structures. One example would be Zr-89-labeled carbon nanotubes for the treatment of tumor vasculature (Reiner et al. 2015). In case of nanoparticle labeling, it is also possible to design a dual radiolabeled nanotheranostic bearing both Y-90 for therapy and Zr-89 for high resolution simultaneous imaging. However, due to the long half-life, the radiation dose needs to be observed carefully and the dose-limiting organ needs to be defined. For most macromolecules, the liver is considered as the critical organ. In some cases, accumulation of Zr(IV) cations in the bone can render the bone marrow as the dose limiting organ.

9.3 APPLICATION OF THERANOSTICS

9.3.1 Neuroendocrine Tumors

A neuroendocrine neoplasm or neuroendocrine tumor (NET) is a heterogeneous group of epithelial neoplasms predominantly characterized by neuroendocrine differentiation (Phan et al. 2010). These tumors are characterized by a high expression of somatostatin receptors (SSTRs) and specific markers, including chromogranin A and CD53 protein (Severi et al. 2017). Currently, NET is classified based on the grading (Ki67 and mitotic index [MI]) and location of the neoplasms. Although NETs can originate from any region of the body, 60%–70% derive from the gastroenteropancreatic system, called GEP-NET. Approximately 30% of NETs have associated symptoms; therefore, they are considered as functioning tumors. These symptoms are often related to the hormonal hypersecretion, which frequently include diarrhea, flushing, hyperglycemia, or hypoglycemia. NETs of the lungs are also known as pulmonary carcinoid tumors, which account for 1%–2% of all lung tumors and approximately 30% of all NETs. Pulmonary NETs are classified according to their mitotic count rate and the presence of necrosis (Phan et al. 2010).

Theranostics of NETs include diagnosis of the disease with PET/computed tomography (CT) using Ga-68-labeled somatostatin analogs. These bind specifically to different SSTR subtypes and allow the molecular imaging and characterization of NETs with a very high diagnostic sensitivity and specificity for the early identification of metastases (Modlin et al. 2010; Phan et al. 2010). This has been shown to have a high impact on patient management (Hofman et al. 2012). In pulmonary NETs, SSTR PET/CT has been shown to have a significant impact on treatment strategy in up to 18% of the patients (Prasad et al. 2015). This subsequently allows the best management approach for each patient (Kulkarni et al. 2014a), as well as the evaluation of treatment response posttherapy (Kulkarni et al. 2014b). Following the determination of SSTR expression using PET/CT, PRRT can be instituted using therapeutic radionuclides (e.g., beta- or alpha-emitting radioisotopes) labeled with the same tracer for personalized treatment.

9.3.2 PSMA—Theranostics

Prostate cancer (PCa) is a major health problem in men. Prognosis is dependent on the stage of the disease and the five-year survival rate for advanced, disseminated PCa is only 29% (Siegel et al. 2012). Metastatic castration-resistant PCa (mCRPC) is associated with poor prognosis and a diminished quality of life. Treatment options for mCRPC patients include taxane-based therapies (docetaxel, cabazitaxel) and novel second-line hormonal therapies (enzalutamide and abiterone), which all show moderate survival benefits. Unfortunately, those therapies are only temporarily effective and development of treatment resistance has been observed (Heidenreich and Porres 2014; Heidenreich et al. 2014). The bone-seeking alpha-emitting radiopharmaceutical radium-223 chloride (XofigoTM) has recently been introduced, resulting in a reduction of bone pain and a moderately improved survival rate in mCRPC patients with bone metastases (Hoskin et al. 2014; Humm 2015). However, the use of this promising radiopharmaceutical is limited to the treatment of bone metastases. Therefore, there is an urgent need to develop more specific therapies targeting PCa visceral lesions as well.

Recent developments in PCa-targeted radiopharmaceuticals have focused on prostate-specific membrane antigen (PSMA) ligands. PSMA is a favorable target for imaging and radionuclide therapy of PCa, because it is over expressed in 90%–100% of local PCa lesions, as well as in cancerous lymph node metastases and bone lesions. Furthermore, reports indicate that PSMA expression levels are further enhanced in high-grade, metastatic, and castration-resistant PCa (Bostwick et al. 1998).

Numerous studies have shown the value of radiolabeled PSMA-targeted agents to visualize PCa lesions, including PCa metastases (Barrett et al. 2013; Pandit-Taskar et al. 2015). Among the different PSMA-targeting tracers available, several tracers have made their way to the clinic. For PET imaging of PCa, Glu-NH-CO-NH-Lys-(Ahx)-68Ga-HBED-CC, also known as 68Ga-PSMA-11 or 68Ga- PSMA-HBED-CC, is the most commonly used compound (Rowe et al. 2015; Budäus et al. 2016). Unfortunately, HBED-CC (N,N'-bis[2-hydroxy-5-(carboxyethyl)benzyl] ethylenediamine-N,N'-diacetic acid), the chelator used in this tracer, is not suitable for radiolabeling with radiometals for therapeutic applications (e.g., 177Lu, 90Y, and 213Bi). More recently, a novel 18F-labeled PSMA inhibitor, 18F-DCFPyL, was introduced in the clinic (Szabo et al. 2015). For therapy, anti-PSMA monoclonal antibodies (J591) radiolabeled with 177Lu and 90Y (Tagawa et al. 2013; Vallabhajosula et al. 2016) and the small-molecule PSMA inhibitor [131I] MIP-1095 have shown promising therapeutic efficacy, although with moderate toxicity. Moreover, PSMA ligands with DOTA-derived chelators have been developed, which can be labeled with diagnostic radionuclides, for example, 68Ga for PET and 111In for SPECT, as well as therapeutic radionuclides, for example, 177Lu, 90Y or 213Bi for radionuclide therapy. Such theranostic compounds, that is, PSMA I&T and PSMA-617 are of great interest because they combine the potential of diagnosis and radionuclide therapy using one PSMA targeting molecule (Weineisen et al. 2014; Benešová et al. 2015).

Radio-ligand properties, for example, lipophilicity, target affinity, and metabolic stability have been carefully optimized for theranostic applications. Monoclonal antibodies have shown slow blood clearance, which may result in myelotoxicity.

Small-molecule PSMA inhibitors showed the advantage to localize rapidly in tumor lesions, including soft-tissue and bone metastases. On the other hand, these small inhibitors showed high and specific uptake in PSMA-expressing kidneys and salivary glands, which is of major concern for its application for radionuclide therapy. Strategies should therefore be investigated to limit radiation-induced damage in healthy organs. Here, we report on the preclinical application of the DOTAGA-chelated urea-based PSMA inhibitor PSMA I&T for SPECT/CT imaging and radionuclide therapy of PCa.

9.3.3 CXCR4—Theranostics

CXCR4 is a member of the chemokine receptor subfamily of seven transmembrane domain, G-protein coupled receptors, whose sole known natural ligand is CXCL12/SDF-1. CXCR4 is an unusual chemokine receptor by virtue of having expanded roles beyond leukocyte recruitment, including fundamental processes such as the development of the hematopoietic, cardiovascular, and nervous systems during embryogenesis. The receptor was first discovered as one of the coreceptors for HIV, and thereafter was also found to be expressed by multiple cancers including breast, prostate, lung, colon, and multiple myeloma.

A number of recent studies have correlated high levels of CXCR4 expression in cancers with poor prognosis and with resistance to chemotherapy, in part through enhancing interactions between cancers and stroma. A possible role for CXCR4 and chemokine receptors generally in cancer and metastasis was first suggested in the studies of breast cancer showing that the receptor plays a role in directing metastatic cells to CXCL12-expressing organs. Collectively, the data on CXCR4 in cancer suggest this receptor increases tumor cell survival and/or growth and/or metastasis, making it a potentially attractive therapeutic target.

Due to the role of CXCR4 in HIV, multiple CXCR4 antagonists—although not sufficient for the treatment of HIV, are currently being evaluated and/or used for stem cell mobilization and as antitumor therapy. Some of the antagonists were also shown in animal models to be of use in evaluating CXCR4 expression in whole tumors noninvasively by molecular imaging. The research on CXCR4 has been ongoing for the last decade and has yielded more than 11,398 papers in PubMed, as of December 2017.

The primary pathology, in which CXCR4 was found to be essential, is infection by human immunodeficiency virus (HIV), and this finding initiated vast research efforts on the receptor. The role of CXCR4 by the T-trophic HIV as a coreceptor was later identified as a late phase of the disease, while the main coreceptor of the virus during the beginning of an HIV infection is another chemokine receptor, CCR5. The detailed role of CXCR4 in HIV infection, and the switch of the virus from CCR5-dependent M-trophic form to CXCR4-dependent T-trophic form has been elaborately discussed by Elisa Vicenzi et al. (2013).

9.4 MAJOR PROBLEMS

The fundamental principle of theranostics has been followed since decades, with the utilization of radioiodine to diagnose (image) and manage disease. Since its inception

until now, theranostics has been fostering unremarkable tailored and targeted therapy. In the context of pharmacotherapy, the fundamental aspects of individual absorption, distribution, metabolism, and excretion are understood with regards to the complex interactions and detailed characterization, especially the functional variations. Interestingly, epigenetic variations and the impact of environmental factors, including the circadian rhythms, on an individual's response to drugs needs to be assessed specifically for each individual (Lütje 2015).

Apart from the above variations, educating the health care provider, payor, regulator, and the patient is also mandatory. Theranostics has an extended impact on patients in more than individual centric care. The patients always expect a positive outcome through the therapeutic diagnosis, which is an alarming aspects of theranostics. The existing less targeted and tailored pharmacotherapy has resulted in the loss for the overall economy of the pharmacological market accounting to $400 billion of total global pharmaceutical market, that is $825 billion.

The most common reason for failure and avoidable deaths—including expensive hospitalizations, has been found to be the side effects associated with less targeted nonspecific therapies, which are based on principles contrary to theranostics. Furthermore the age, gender, diet, and lifestyle of a patient including the intestinal microflora plays a significant role in the patient's response to a drug.

9.5 TRANSITIONING INTO CLINICAL REALITY—A MACHINE LEANING APPROACH

9.5.1 Positron Emission Tomography (PET)—Computed Tomography (PET/CT)

PET uses small amounts of radioactive materials called radiotracers, a special camera, and a computer, which evaluate your organ and tissue functions. By identifying body changes at the cellular level, PET may be used to detect the early onset of disease before it is evident on other imaging tests.

A PET scan creates pictures of organs and tissues in the body. First, a technician injects the patient with a small amount of a radioactive substance, which the organs and tissues pick up and absorb. Areas using more energy pick up more of the radioactive substance and thus cancer cells pick up a lot of it, as they tend to use more energy than healthy cells. Then a scan shows where the radioactive substance is present in the body. For this, a CT scan uses X-rays to create a 3D picture of the inner body, and it detects and shows anything abnormal, including tumors. Sometimes, a special dye, called as contrast medium is given before the scan to provide better details in the image (Rbah-Vidal 2017).

A PET-CT does not hurt but the patient needs to lie still for the entire scan. He/she might also need to keep their arms above the head, which could become uncomfortable. The technician generally asks patients to hold their breath at times because motion from breathing may cause blurry pictures. It is also advised to raise, lower, or tilt the table during the scan to allows the capture of pictures from different angles.

9.5.2 Single Photon Emission Computed Tomography—Computed Tomography (SPECT/CT)

SPECT-CT is where two different types of scans are taken and the images or pictures from each are fused or merged together to form an overlapping image. The fused scan can provide more precise information about how different parts of the body function and can clearly identify problems such as tumors (lumps) or Alzheimer's disease, etc. SPECT images are obtained following an injection of a radiopharmaceutical used for nuclear medicine scans. The injected medication sticks to specific areas in the body, depending on what radiopharmaceutical is used and the type of scan being performed. For example, it shows bones for a bone scan, and gall bladder and bile ducts for a hepatobiliary scan.

The radiopharmaceutical can be detected by a nuclear medicine gamma camera. The camera or cameras rotate over a 360° arc around the patient, allowing the reconstruction of an image in 3D. In computed tomography (CT) images are obtained while lying on a bed moving through a ring, or "donut" shaped X-ray machine. Again, the X-ray machine rotates over a 360-degree arc around the patient, allowing for image reconstruction in 3D. The X-ray machine from the CT scanner rotates much faster than the gamma camera; therefore, the CT part of the study takes less time than that of the SPECT study. The similarity between the SPECT and CT method of image processing allows the images to be combined. Combining the information from the nuclear medicine SPECT study and the CT study allows the information about function from the nuclear medicine study to be easily combined with the information about how the body structure "looks" in the CT study.

One must lie still in a ring-shaped scanner for at least 30–40 minutes. The first 3–5 minutes involves the CT scan component, while the remainder of the time is required for the SPECT study. It is very important to remain still for the entire duration of the two studies so that the SPECT and CT can be accurately combined. If not, the images from one study will not correspond exactly to the images from the other study, and may result in interpretation difficulties.

There are no risks involved in the nuclear medicine, that is, SPECT scan or the CT scan procedures. The tests involve small doses of ionizing radiation from the radiopharmaceuticals injected into the vein.

Importantly, the SPECT component of the test requires no additional injection of radiopharmaceuticals beyond what would otherwise be given for nuclear medicines tests. The CT is usually done using a low-dose radiation technique, which is around 20%–25% of the radiation exposure of a normal CT scan.

9.5.3 Quantitative SPECT

While accurate quantitation of activity distribution may be advantageous for many diagnostic imaging studies, it is absolutely essential for personalized dosimetry in radionuclide therapies. Factors affecting the measure of the quantitative SPECT are illustrated in Figure 9.6.

With the increased availability of hybrid SPECT/CT systems, fully quantitative determination of radiotracer distribution in tumors and critical organs has become both

FIGURE 9.6 Factors affecting the measure of the quantitative SPECT of various organ in the human system.

feasible and practical. This is required to develop methods to accurately reconstruct activity distributions, expressed in absolute units (Bq or Ci), from conventional clinical SPECT studies performed using both diagnostic and therapeutic radiotracers.

9.5.3.1 Camera Normalization

Once quantitatively accurate images are reconstructed, the counts in each voxel must be translated into activity values. This is done using an experimentally determined camera normalization factor. This is required to investigates the accuracy of the camera normalization factor, determined using the planar scan of a point source versus that obtained from a tomographic scan of a large phantom filled with activity.

9.5.3.2 Scatter Corrections

For quantitative imaging it is essential to remove all the scattered photons from the photopeak energy window, and several methods have been developed for this purpose. This project evaluates the accuracies of various simple methods (such as triple-energy-window) and compares them with more complex but also more reliable approaches (such as Monte Carlo and analytical photon distribution calculation). Corrections for self-scatter and high-energy scatter are being investigated.

9.5.3.3 Camera Dead-Time

Activities of radiotracers used in diagnostic imaging are low, resulting in low count rates with no camera dead-time. However, in typical radionuclide therapy procedures patients are injected with high activities (of the order of GBq), emitting high fluxes of gammas. In order to accurately quantify a patient's activity distribution in these situations, various investigations and developments are carried out in order to correct the reconstructed images for these dead-time losses.

9.5.3.4 Bremsstrahlung Radiation

Targeted radionuclide therapies typically use radioisotopes, which decay through β emissions (electrons). These electrons create Bremsstrahlung photons, which may affect the quantitative accuracy of images. This is required to investigate the characteristics of bremsstrahlung emissions created by ß-emissions in tissues, and to analyze the spectra recorded by SPECT cameras with different types of collimators.

9.5.3.5 Quantitative Reconstructions

A graphical user interface (GUI) named SPECToR allows the quantitative reconstruction of images from data acquired using different SPECT/CT cameras. Parameters of the reconstruction algorithm, as well as quantitative corrections can be varied. The comparison of quantitative accuracies of images, which are reconstructed using commercial software obtained from SPECToR.

9.5.3.6 Quantitative Preclinical Imaging

Development of new pharmaceuticals and new treatments prior to their introduction into clinics often involve the use of small animal models, which often lead to the crucial analysis of pharmacokinetic, targeting, and toxicity during preclinical imaging. Such imaging requires cameras with high resolution, providing accurate quantification of tracer uptakes. This is required to investigate the quality and quantitative accuracy of images obtained from the VECTor/CT camera using different diagnostic and therapeutic radioisotopes.

9.5.3.7 Patient Specific Dosimetry/Dose Calculations

As therapeutic uses of radionuclides in nuclear medicine have increased, the use of patient-specific methods for calculation of radiation dose has become more important. Most current resources make use of standardized models of the human body representing median individuals, but the use of image-based and more realistic models will soon take their place, and will permit adjustments to represent individual patients and tailor therapy planning uniquely for each subject.

One of the important beneficial applications of the use of radiation is in the healing arts. Unlike in many other uses of radiation, the patient receives a direct benefit from the study and therefore evaluation of the risk/benefit relationship is more straightforward. An important example is the use of radiopharmaceuticals, that is, nuclear medicine in both diagnosis and therapy. Diagnostic uses of radiopharmaceuticals are well established and are employed to evaluate a broad variety of patient conditions. Radiation doses for diagnostic agents are developed by studying the biokinetics of the radiopharmaceuticals in preclinical and clinical studies. In the therapeutic approach, extrapolation methods are applied to convert values measured in animal organs for humans, and in the diagnostics case, the quantitative data has been observed in the human subjects and can be used directly for input to dose calculations (Stabin 2006).

The basic goal of all forms of radiation therapy (using external or internal radiation sources) is to deliver a lethal radiation dose to the unhealthy tissues of concern while avoiding or limiting the expression of undesired effects in other normal tissues of the patient. Radioactive iodine (131 I) has been used for many years to treat benign thyroid disease with and without patient-specific treatment planning (Canzi et al. 2006; Carlier et al. 2006). Treatment of thyroid cancer with 131I NaI is the most common application of radionuclide therapy in nuclear medicine and has been used for many decades. Patient therapy is most often based on administration of fixed levels of radiation to all subjects, rather than targeting absorbed doses, although there have been exceptions.

I-131 labeled meta-iodobenzylguanidine (mIBG) has been used for many years in the treatment of adult and pediatric neuroendocrine tumors (including phaeochromocytoma, paraganglioma, and neuroblastoma), typically with administrations of 7.4 GBq to more than 30 GBq in adults. To treat cancer, several monoclonal antibodies have been developed.

A general equation for the absorbed dose rate in an object uniformly contaminated with radioactivity (for example an organ or tissue with radiopharmaceutical uptake) may be shown as:

$$\dot{D}_T = \frac{k\,A_S \sum_i y_i E_i \phi_i}{m_T}$$

where

D_T = absorbed dose rate to a target region of interest (Gy/sec)
A_S = activity (MBq) in source region S
y_i = number of radiations with energy E_i emitted per nuclear transition
E_i = energy per radiation for the ith radiation (MeV)
ϕ_i = fraction of energy emitted in a source region, which is absorbed in a target region
m_T = mass of the target region (kg)
k = proportionality constant (Gy-kg/MBq-sec-MeV)

The proportionality constant k includes the various factors needed to obtain the dose rate in the desired units from the units employed for the other variables, and it is essential that this factor is properly calculated and applied. The time versus activity curve for theranostics agents for a specific organ is given in Figure 9.7.

9.5.3.8 Image-Based Computational Tools

Several centers have implemented the use of image fusion techniques to develop 3D maps of the dose, instead of average organ dose, which is estimated from standard

FIGURE 9.7 Time/activity curve to study the activity of an organ in the human system.

models thatare generally available. This suggests the treatment planning for internal emitters may soon be far more sophisticated and similar to that used in external beam therapy for individualized patient therapy planning. Examples include the 3D-ID code from the Memorial Sloan-Kettering Cancer Center (Beauregard et al. 2011), the SIMDOS code from the University of Lund, the RTDS code at the City of Hope Medical Center, the RMDP code from the Royal Marsden Hospital, the DOSE3D code, and the PEREGRINE code developed by various researchers (Willowson et al. 2012).

Radiation dose calculations for radiopharmaceuticals have been standardized by the implementation and dissemination of tools such as the RADAR web site and the OLINDA/EXM software (Grbic et al. 2013). Current efforts suggest a move towards more image-based and patient-specific methods in internal dose calculations for therapeutic applications in nuclear medicine (e.g., the 3D-ID code). Current evidence in the literature strongly supports the idea that patient-specific dose calculations are needed to improve patient outcomes when internal emitters are used in therapy, as is commonly accepted in external radiation therapy (Lehmkuhl et al. 2013).

9.6 CONCLUSION

The usage of theranostic agents is thought to be an effective approach because it allows one to selectively deliver the active drug to the cancer site and to simultaneously monitor the therapeutic efficacy through the use of imaging technology. We hope that this review will be beneficial for researchers aiming to improve and expand the variety of theranostic agents by overcoming the drawbacks of each agent and using the merits of each agent synergically.

REFERENCES

Afshar-Oromieh A. et al. "Repeated PSMA-Targeting radioligand therapy of metastatic prostate cancer with ^{131}I-MIP-1095." *European Journal of Nuclear Medicine and Molecular Imaging* 44, no. 6, 2017: 950–959.

Banerjee S.R. et al. "Preclinical evaluation of ^{86}Y-labeled inhibitors of prostate-specific membrane antigen for dosimetry estimates." *Journal of Nuclear Medicine: Official Publication, Society of Nuclear Medicine* 56, no. 4, 2015: 628–634.

Barrett J.A. et al. "First-in-Man evaluation of 2 high-affinity PSMA-Avid small molecules for imaging prostate cancer." *Journal of Nuclear Medicine* 54, no. 3, 2013: 380–387.

Beauregard J.M., Hofman M.S., Pereira J.M., Eu P., Hicks R.J. et al. "Quantitative 177LuSPECT (QSPECT)imaging using a commercially available SPECT/CT system." *Cancer Imaging* 11, no. 4, 2011: 56–66.

Benešová M. et al. "Preclinical evaluation of a Tailor-made DOTA-conjugated PSMA inhibitor with optimized linker moiety for imaging and endoradiotherapy of prostate cancer." *Journal of Nuclear Medicine* 56, no. 6, 2015: 914–920.

Blankenberg F.G., Strauss H.W. "Nuclear medicine applications in molecular imaging." *Journal of Magnetic Resonance Imaging* 16, no. 4, 2012: 352–61.

Bostwick D.G., Pacelli A., Blute M., Roche P., Murphy G.P. "Prostate specific membrane antigen expression in prostatic intraepithelial neoplasia and adenocarcinoma: A study of 184 cases." *Cancer* 82, no. 11, 1998: 2256–61.

Broda J. et al. "Assessing the intracellular integrity of phosphine-stabilized ultrasmall cytotoxic gold nanoparticles enabled by fluorescence labeling." *Advanced Health Care Materials* 5, 2016: 3118–3128.

Budäus L. et al. "Initial experience of ^{68}Ga-PSMA PET/CT imaging in high-risk prostate cancer patients prior to radical prostatectomy." *European Urology* 69, no. 3, 2016: 393–396.

Canzi C., Zito F., Voltini F., Reschini E., Gerundini P. "Verification of the agreement of two dosimetric methods with radioiodine therapy in hyperthyroid patients." *Medical Physics* 33, no. 8, 2006: 2860–2867.

Carlier T. et al. "Optimized radioiodine therapy for Graves' disease: Two MIRD-based models for the computation of patient-specific therapeutic I-131 activity." *Nuclear Medicine Communications* 27, no. 7, 2006: 559–566.

Cheng L. et al. "Facile preparation of multifunctional WS2/WO*x* nanodots for chelator-free 89Zr-labeling and in vivo PET imaging." *Small* 12, no. 41, 2016: 5750–5758.

Crawley N., Thompson M. "Theranostics in the growing field of personalized medicine: An analytical chemistry perspective." *Analytical Chemistry* 86, no. 1, 2014: 130–160.

Dash A., Knapp F.F., Pillai M.R.A. "Targeted radionuclide therapy—an overview." *Current Radiopharmaceuticals* 6, no. 3, 2013: 152–180.

Enrique M.-A., Mariana O.-R., Mirshojaei S.F., Ahmadi A. "Multifunctional radiolabeled nanoparticles: Strategies and novel classification of radiopharmaceuticals for cancer treatment." *Journal of Drug Targeting* 23, no. 3, 2015: 191–201.

Ernsting M.J., Murakami M., Roy A., Li S.-D. "Factors controlling the pharmacokinetics, biodistribution and intratumoral penetration of nanoparticles." *Journal of Controlled Release* 172, no. 3, 2013: 782–794.

Ferber S. et al. "Polymeric nanotheranostics for real-time non-invasiveoptical imaging of breast cancer progression and drug release." *Cancer Letter* 352, no. 1, 2014: 81–89.

García-Garayoa E., Schibli R., Schubiger P.A., "Peptides radiolabeled with Re-186/188 and Tc-99m as potential diagnostic and therapeutic agents." *Nuclear Science and Techniques* 18, no. 2, 2007: 88–100.

Goedicke A. et al. "Study-parameter impact in quantitative 90-Yttrium PET imaging for radioembolization treatment monitoring and dosimetry." *IEEE Transactions on Medical Imaging* 32, no. 3, 2013: 485–492.

Gotthardt M., Kwekkeboom D.J., Feelders R.A., Brouwers A.H., Teunissen J.J. et al. "Nuclear medicine imaging and therapy of neuroendocrine tumours." *Cancer Imaging* 6, no. Spec No A, 2006: S178–S184.

Grbic S. et al. "Image-based computational models for TAVI planning: From CT images to implant deployment." *Medical Image Computing and Computer-assisted Intervention* 16, no. 2, 2013: 395–402.

Gustafsson A.M. et al. "Comparison of therapeutic efficacy and biodistribution of ^{213}Bi- and ^{211}At-labeled monoclonal antibody MX35 in an ovarian cancer model." *Cancer* 39, no. 1, 2012: 15–22.

Häfeli U., Pauer G., Failing S., Tapolsky G. "Radiolabeling of magnetic particles with rhenium-188 for cancer therapy." *Journal of Magnetism and Magnetic Materials*, 225, no. 1–2, 2001: 73–78.

Heidenreich A. et al. "EAU guidelines on prostate cancer. Part II: Treatment of advanced, relapsing, and castration-resistant prostate cancer." *European Urology* 65, no. 2, 2014: 467–79.

Heidenreich A., Porres D. "Prostate cancer: Treatment sequencing for CRPC—what do we know?" *Nature Reviews Urology* 11, 2014: 189–190.

Hofman M.S. et al. " High management impact of Ga-68 DOTATATE (GaTate) PET/CT for imaging neuroendocrine and other somatostatin expressing tumours." *Journal of Medical Imaging and Radiation Oncology* 56, no. 1, 2012: 40–7.

Hood E. "Pharmacogenomics: The promise of personalized medicine." *Environ Health Perspect* 111, no. 11, 2003: 581–589.

Hoskin P. et al. "Efficacy and safety of radium-223 dichloride in patients with castration-resistant prostate cancer and symptomatic bone metastases, with or without previous docetaxel use: A prespecified subgroup analysis from the randomised, double-blind, phase 3 ALSYMPCA trial." *The Lancet. Oncology* 15, no. 12, 2014: 1397–406.

Humm J.L., Sartor O., Parker C., Bruland O.S., Macklis R. "Radium-223 in the treatment of osteoblastic metastases: A critical clinical review." *International Journal of Radiation Oncology, Biology, Physics* 91, no. 5, 2015: 898–906.

Jeelani S. et al. "Theranostics a treasured tailor for tomorrow." *Journal of Pharmacy & Bioallied Sciences* 6, 2014: S6-S8.

Kelkar S.S., Reineke T.M. "Theranostics: Combining imaging and therapy." *Bioconjugate Chemistry* 22, no. 10, 2011: 1879–1903.

Kramer L. et al. "Quantitative and correlative biodistribution analysis of 89Zr-labeled mesoporous silica nanoparticles intravenously injected into tumor-bearing mice." *Nanoscale* 9, no. 27, 2017: 9743–9753.

Kulkarni H.R., Baum R.P. "Patient selection for personalized peptide receptor radionuclide therapy using Ga-68 somatostatin receptor PET/CT." *PET Clinics* 9, no. 1, 2014a: 83–90.

Kulkarni H.R., Baum R.P. "Theranostics with Ga-68 somatostatin receptor PET/CT: Monitoring response to peptide receptor radionuclide therapy." *PET Clinics* 9, no. 1, 2014b: 91–97.

Lange R. et al. "Treatment of painful bone metastases in prostate and breast cancer patients with the therapeutic radiopharmaceutical rhenium-188-HEDP." *Nuklearmedizin* 55, no. 5, 2016: 177–209.

Lehmkuhl L. et al. "Role of preprocedural computed tomography in transcatheter aortic valve implantation." *Rofo* 185, no. 10, 2013: 941–949.

Loon J. et al. "PET imaging of zirconium-89 labelled cetuximab: A phase I trial in patients with head and neck and lung cancer." *Radiotherapy and Oncology* 122, no. 2, 2017: 267–273.

Lütje S. et al. "PSMA ligands for radionuclide imaging and therapy of prostate cancer: Clinical status." *Theranostics* 5, no. 12, 2015: 1388–1401.

Maecke H.R. and Reubi J.C. "Somatostatin receptors as targets for nuclear medicine imaging and radionuclide treatment." *Journal of Nuclear Medicine* 52, no. 6, 2017: 841–844.

Modlin I.M., Pavel M., Kidd M., Gustafsson B.I. "Review article: Somatostatin analogues in the treatment of gastroenteropancreatic neuroendocrine (carcinoid) tumours." *Alimentary Pharmacology & Therapeutics* 31, no. 2, 2010: 169–88.

Morgenroth A., Vogg A.T., Neumaier B., Mottaghy F.M., Zlatopolskiy B.D. "Radioiodinated indomethacin amide for molecular imaging of cyclooxygenase-2 expressing tumors." *Oncotarget* 8, no. 11, 2017: 18059–18069.

Nayak T.K., Brechbiel M.W. "^{86}Y based PET radiopharmaceuticals: Radiochemistry and biological applications." *Medicinal Chemistry* 7, no. 5, 2011: 380–388.

Pandit-Taskar N. et al. "A phase I/II study for analytic validation of ^{89}Zr-J591 ImmunoPET as a molecular imaging agent for metastatic prostate cancer." *Clinical Cancer Research: An Official Journal of the American Association for Cancer Research* 21, no. 23, 2015: 5277–5285.

Perk L.R. et al. "Preparation and evaluation of 89Zr-Zevalin for monitoring of 90Y-Zevalin biodistribution with positron emission tomography." *European Journal of Nuclear Medicine Molecular Imaging* 33, no. 11, 2006: 1337–1345.

Phan A.T. et al. "NANETS consensus guideline for the diagnosis and management of neuroendocrine tumors: Well-differentiated neuroendocrine tumors of the thorax (Includes Lung and Thymus)." *Pancreas* 39, no. 6, 2010: 784–798.

Prasad V. et al. "Somatostatin receptor PET/CT in restaging of typical and atypical lung carcinoids." *EJNMMI Research* 5, 2015: 53.

Rbah-Vidal L. et al. "Theranostic approach for metastatic pigmented melanoma using ICF15002, a multimodal radiotracer for both PET imaging and targeted radionuclide therapy." *Neoplasia* 19, no. 1, 2017: 17–27.

Reiner T., Lewis J.S., Zeglis B.M. "Harnessing the bioorthogonal inverse electron demand diels-alder cycloaddition for pretargeted PET imaging." *Journal of Visualized Experiments* 96, 2015: 52335.

Rösch F., Herzog H., Qaim S.M. "The beginning and development of the theranostic approach in nuclear medicine, as exemplified by the radionuclide pair ^{86}Y and ^{90}Y." Ed. Klaus Kopka. *Pharmaceuticals* 10, no. 2, 2017: 56.

Rowe S.P. et al. "^{18}F-DCFBC PET/CT for PSMA-based detection and characterization of primary prostate cancer." *Journal of Nuclear Medicine: Official Publication, Society of Nuclear Medicine* 56, no. 7, 2015: 1003–1010.

Same S., Aghanejad A., Akbari Nakhjavani S., Barar J., Omidi Y.. "Radiolabeled theranostics: Magnetic and gold nanoparticles." *Bioimpacts* 6, no. 3, 2016: 169–181.

Severi S. et al. "Peptide receptor radionuclide therapy in the management of gastrointestinal neuroendocrine tumors: Efficacy profile, safety, and quality of life." *OncoTargets and Therapy* 10, 2017: 551–557.

Shah R.R., Shah D.R. "Personalized medicine: Is it a pharmacogenetic mirage?" *British Journal of Clinical Pharmacology* 74, no. 4, 2012: 698–721.

Siegel R. et al. "Cancer treatment and survivorship statistics, 2012." *CA: A Cancer Journal for Clinicians* 62, no. 4, 2012: 220–41.

Silberstein E.B. "Radioiodine: The classic theranostic agent." *Seminars in Nuclear Medicine* 42, no. 3, 2012: 3164–170.

Sisson J.C., Ynik G.A. et al. "Theranostics: Evolution of the radiopharmaceutical meta-iodobenzylguanidine in endocrine tumors." *Seminars in Nuclear Medicine* 42, no. 3, 2012: 171–184.

Stabin M.G. Internal Radiation Dosimetry. *Nuclear Medicine*, 2nd Ed., Vol 1, Chapter 22, 313–331, edited by RE Henkin, D Bova, GL Dillahay, JR Halema, SM Karesh, RH Wagner, AM Zimmer. Mosby, St. Louis, MO 2006.

Stockhofe K., Postema J.M., Schieferstein H., Ross T.L. "Radiolabeling of nanoparticles and polymers for PET imaging." *Pharmaceuticals* 7, no. 4, 2014: 392–418.

Szabo Z. et al. "Initial evaluation of [^{18}F]DCFPyL for prostate-specific membrane antigen (PSMA)-targeted PET imaging of prostate cancer." *Molecular Imaging and Biology: MIB: The Official Publication of the Academy of Molecular Imaging* 17, no. 4, 2015: 565–574.

Szymański P., Frączek T., Markowicz M., Mikiciuk-Olasik E., "Development of copper based drugs, radiopharmaceuticals and medical materials." *BioMetals* 25, no. 6, 2012: 1089–1112.

Tagawa S.T. et al. "Phase II study of Lutetium-177 labeled anti-prostate-specific membrane antigen (PSMA) monoclonal antibody J591 for metastatic castration-resistant prostate cancer." *Clinical Cancer Research: An Official Journal of The American Association for Cancer Research* 19, no. 18, 2013: 5182–5191.

Uematsu T. et al. "Comparison of FDG PET and SPECT for detection of bone metastases in breast cancer." *American Journal of Roentgenology* 184, no. 4, 2005: 1266–1273.

Vallabhajosula S. et al. "Radioimmunotherapy of metastatic prostate cancer with ^{177}Lu-DOTAhuJ591 anti prostate specific membrane antigen specific monoclonal antibody." *Current Radiopharmaceuticals* 9, no. 1, 2016: 44–53.

Vicenzi E., Liò P., Poli G. "The puzzling role of CXCR4 in human immunodeficiency virus infection." *Theranostics* 3, no. 1, 2013: 18–25.

Vöö S., Bucerius J., Mottaghy F.M. "I-131-MIBG therapies." *Methods* 55, no. 3, 2011: 238–245.

Weineisen M., Simecek J., Schottelius M., Schwaiger M., Wester H.-J. " Synthesis and preclinical evaluation of DOTAGA-conjugated PSMA ligands for functional imaging and endoradiotherapy of prostate cancer." *EJNMMI Research* 4, 2014: 63.

Willowson K. et al. "CT-based quantitative SPECT for theradionuclide 201Tl:experimental validation and a standardized uptake value for brain tumour patients." *Cancer Imaging* 12, 2012: 31–40.

Wright C.L., Zhang J., Tweedle M.F., Knopp M.V., Hall N.C. "Theranostic imaging of Yttrium-90." *Biomed Research International* 2015 Article ID 481279 doi:10.1155/2015/481279.

Zechmann C.M. et al. "Radiation dosimetry and first therapy results with a ^{124}I/^{131}I-Labeled small molecule (MIP-1095) targeting PSMA for prostate cancer therapy." *European Journal of Nuclear Medicine and Molecular Imaging* 41, no. 7, 2014: 1280–1292.

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

10 Emotion Detection System

Adrish Bhattacharya, Vibhash Chandra, and Leonid Datta

CONTENTS

10.1 Introduction 168
10.1.1 Background 168
10.1.2 Emotion Recognition—Putting the Smart in Smart Devices 168
10.2 Types of Emotion 169
10.2.1 The Basic Emotions 169
10.2.2 Emotion Categories 170
10.2.3 Plutchik's Model 170
10.2.4 PANA Model 170
10.2.5 Pad Emotional State Model 171
10.2.6 HUMAINE's Proposal for Emotion Annotation and Representation Language (EARL) 172
10.3 Speech Parameters 172
10.3.1 Amplitude and Pitch 172
10.3.2 MFCC Parameters 173
10.4 Choice of Parameters 174
10.4.1 Benefits of MFCC: Why MFCC is Used 174
10.5 Programming Tools Used 174
10.5.1 Training and Testing Dataset 174
10.5.1.1 Databases Used 174
10.5.1.2 Features and Target Labels 175
10.5.1.3 Train-Test Split 175
10.5.2 Building the Classification Model 175
10.6 Applications 176
10.6.1 HCI: HCI Applications of the System 176
10.6.2 Medical Applications 177
10.6.3 Responsive Systems 178
10.7 Conclusion 178
References 179

10.1 INTRODUCTION

10.1.1 Background

The recent past has witnessed enormous advancements pertaining to artificial intelligence, which has become the buzzword of scientists, technocrats, and students alike. We have created machines able to not only understand what is said to them, but which can also laugh, joke, or make sarcastic comments. Indeed, we have created smart machines able to respond to almost anything we want. First phones, after that, watches and now TVs, homes, and cars—every imaginable utilitarian device has become smart based on the exponential growth in artificial intelligence. However, as we delve deeper into the solutions these devices provide, a basic constraint emerges limiting the practical applications of these systems. This limiting factor is due to its inability to process paralinguistic features of commands. To simplify, our machines can't read between the lines. For instance, if you are tired from working late and place an order for a hot cup of coffee to a smart brewing machine, it wouldn't brew a nice cup of espresso. Instead, it would brew your favorite cappuccino, which is the most frequently ordered variety. But in the same situation, a close associate will definitely bring you a hot brew of black coffee. Similarly, there are numerous situations in day-to-day life the latest smart machines fail to address appropriately. A close friend is always better at recommending books and movies than even the most advanced recommendation engines powered by artificial intelligence. This is especially true when we are not using our regular devices. To summarize, even with our terrific developments in artificial intelligence and big data, these devices are nowhere as smart as we need them to be. It's quite ironic that with huge IQs, our systems hardly have any common sense! Let alone, any degree of emotional quotient.

10.1.2 Emotion Recognition—Putting the Smart in Smart Devices

From our introduction, we understand that to make machines truly smart, it is imperative to design systems able to understand human emotions without the users having to express them explicitly. Reflecting on our everyday experiences, it is easy to note that there is hardly any moment when we feel unemotional. Thus, as humans we generate abundant volumes of emotion related data that is open for exploitation.

However, the study of affective science is hardly a new field. Even ancient Romans and Greeks studied expressive speech and later on their manuals formed the basis for western philosophical work on the subject of the emotional appeal. Active research on recognition of emotions can be traced to the early 1960s when phoneticians, linguists, emotion psychologists, and engineers began exploring affective science.

In the twenty first century, it is the need of the hour to create systems that converge the results of the earlier generations while giving them the ability to efficiently utilize newer and emerging data types. Later in the chapter, we shall discuss various methods of speech analysis and explore the challenges they present. Initially, it's important to understand what constitutes emotion, that is, how to determine and contrast one emotion from other. Without a detailed understanding of emotions, it is not possible to teach machines to identify them. Therefore, let us go deeper into the classification of emotions in the next section.

10.2 TYPES OF EMOTION

It is seemingly a simple task for us to list the various emotions we experience—joy, melancholy, angst, fear, excitement, pleasure, anxiety, boredom, dismay, ecstasy, disappointment, delight, and so on. However, the boundaries between two different emotions are often fuzzy and not discrete. For instance, dismay and disappointment can be closely related, with one of them being the cause and other the effect. Therefore, when we venture into identifying emotions, the task presents newfound challenges not generally encountered in everyday life. In fact, the debate on classification of emotions has riddled experts till today. As a result, there is no unanimous consensus as how to classify emotions. However, two fundamental viewpoints of classification of emotions exist:

1. Differentiating emotions based on their constructs.
2. Dimensional categorization of emotions based on similarity.

10.2.1 The Basic Emotions

When we discuss basic emotions, we assume the discrete nature of emotions. The discrete emotion theory states that every human possesses a cultural-independent and discrete set of emotions. It is argued these basic emotions can be distinguished by an individual's biological processes (Colombetti 2009; Ekman 1992).

In 2017, Cowen and Keltner (2017) quoted:

> Across self-report methods, we find that the [2185] videos [selected and shown to volunteer subjects] reliably elicit 27 distinct varieties of reported emotional experience. Further analyses revealed that categorical labels such as amusement better capture reports of subjective experience than commonly measured affective dimensions (e.g., valence and arousal). Although reported emotional experiences are represented within a semantic space best captured by categorical labels, the boundaries between categories of emotion are fuzzy rather than discrete. By analyzing the distribution of reported emotional states we uncover gradients of emotion—from anxiety to fear to horror to disgust, calmness to aesthetic appreciation to awe, and others—that correspond to smooth variation in affective dimensions such as valence and dominance. Reported emotional states occupy a complex, high-dimensional categorical space.

The categories suggested are:

- A—admiration, adoration, aesthetic, appreciation, amusement, anger, anxiety, awe, awkwardness
- B—boredom
- C—calmness, confusion, craving
- D—disgust
- E—empathic pain, entrancement, excitement
- F—fear
- H—horror
- I—interest
- J—joy

- N—nostalgia
- R—relief, romance
- S—sadness, satisfaction, sexual desire, surprise

Smith's list of emotions—Watt Smith listed 154 different emotions, including foreign ones. This is one of the most comprehensive lists of emotions available.

Richard and Bernice Lazarus (Lazarus and Lazarus 1996), in their book *Passion and Reason,* listed 15 separate types of emotions including shame, anger, love, pride, relief, sadness, fright, gratitude, aesthetic experience, happiness, jealousy, anxiety, compassion, depression, envy, guilt, and hope.

William James proposed four basic emotions based on the bodily involvements of individuals—fear, grief, love, and rage.

Paul Ekman, Wallace Friesen, and *Phoebe Ellsworth* suggested six basic emotions—anger, disgust, fear, happiness, sadness, and surprise.

10.2.2 Emotion Categories

For a long time, researchers have attributed emotions according to multiple dimensions. The Wilhelm Max Wund of modern psychology, described emotions in three dimensions, namely "pleasurable or unpleasurable," "arousing or subduing," and "strain or relaxation" (Wundt 2009). Harold Schlosberg, in 1954, dubbed these three distinct dimensions of emotions as "pleasantness–unpleasantness," "attention–rejection," and "level of activation" (Schlosberg 1954). Unlike models of basic emotion theories, which propose different emotions arising from separate neural systems, the dimensional models argue for the presence of an interconnected neurophysiological system. This idea is central to the different affective states (Posner, Russell, and Petersona 2005).

These models aim to define human emotions by defining where they lie in these proposed dimensions. Although several models have been proposed by researchers, only a few remain accepted by the mainstream scientific community. Among them, the positive activation–negative activation (PANA) model and the vector model are the ones most widely accepted (Rubin and Talarico 2010). A short description of the Plutchick and PANA model has been given below.

10.2.3 Plutchik's Model

Robert Plutchik, who first proposed the model, argued for a mixed three-dimensional model fusing the discrete and dimensional theories. Plutchick had arranged the affective states within concentric circles, with the basic emotions inside the inner circles and the complex emotions in circles of larger radii (Plutchick 2001).

10.2.4 PANA Model

The PANA, or "consensual" model, was devised by Watson and Tellegen (Watson and Tellegen 1985). PANA model argues that the basic differentiating factor between emotional classes is due to positive (Figure 10.1) and negative (Figure 10.2) affect (Watson and Tellegen 1985).

Negative and forceful	Negative and uncontrollable	Negative thoughts	Negative and passive
• Anger	• Anxiety	• Pride	• Boredom
• Annoyance	• Embarrassment	• Doubt	• Despair
• Contempt	• Fear	• Envy	• Disappointment
• Disgust	• Helplessness	• Frustration	• Hurt
• Irritation	• Powerlessness	• Guilt	• Sadness
	• Worry	• Shame	• Agitation
			• Stress
			• Shock
			• Tension

FIGURE 10.1 Negative emotions.

Positive and lively	Positive thoughts	Quiet positive	Reactive
• Amusement	• Courage	• Calmness	• Interest
• Delight	• Hope	• Contentment	• Politeness
• Elation	• Humility	• Relaxation	• Surprise
• Excitement	• Satisfaction	• Relief	
• Happiness	• Trust	• Serenity	
• Joy			
• Pleasure			
• Affection			
• Empathy			
• Friendliness			
• Love			

FIGURE 10.2 Positive emotions.

10.2.5 Pad Emotional State Model

Albert Mehrabian and James Russel proposed the pad emotional state model that is basically a psychological model where 3 numerical dimensions are employed to categorize different emotions. Here three numerical dimensions are employed to categorize different. The pad model puts forward the following dimensions—*pleasure, arousal,* and *dominance*.

The *arousal–nonarousal scale* measures the emotion intensity. For example, worry and anxiety are both unpleasant emotions, however, anxiety has greater intensity compared to worry.

The *pleasure–displeasure scale* measures the pleasantness of any emotion. For example, worry and anxiety are both highly unpleasant emotions but happiness is a pleasant one (Albert 1980).

The *dominance–submissiveness scale* measures the controlling nature in any emotion. For example, annoyance is a dominant emotion, but embarrassment is inherently submissive (Albert 1980).

10.2.6 HUMAINE's Proposal for Emotion Annotation and Representation Language (EARL)

The human–machine interaction network on emotion (HUMAINE) suggests 48 different emotions types (Bales 2000).

10.3 SPEECH PARAMETERS

While discussing speech parameters, it is of the utmost importance to bear in mind what distinguishes one speech from another, and what remains the same. Since we are interested in the paralinguistic analysis of speech, the following features should be compared and contrasted for each emotion class.

1. Intonation
2. Duration
3. Intensity

Humans can easily identify these parameters but they have to be quantized for machines to identify and compare them. For this purpose, various speech parameters need to be considered before classifying any auditory signal. Mainly amplitude and pitch have to be evaluated for different emotions since they are the most basic of speech parameters. Apart from pitch and amplitude, other important parameters are influenced by the mental and physiological states of the speaker (Moriyama, Saito, and Ozawa 2001). Some of them include:

1. Intensity
2. Energy and power
3. Jitter
4. Pauses
5. Speaking rate
6. Mean
7. Variance
8. Mel-frequency cepstrum coefficient (MFCC) and its derivatives

10.3.1 Amplitude and Pitch

As noted earlier, to distinguish between different types of emotions from speech we need to consider a number of parameters; however, for most purposes, a binary classification of the positive and negative emotion state meets the required purpose. The positive emotions include happiness, surprise, excitement, etc., and negative emotions include sadness, fear, and angst among others.

While considering these two classes of emotions, we observe that amplitude and pitch can appreciably distinguish between them. In general, positive emotions are

characterized by an increased amplitude and pitch, while the negative emotions show relatively less amplitude and pitch. Another point of difference is that in positive emotions, the range of values of amplitude and pitch are both larger than those observed in negative emotions. Correspondingly, we can conclude that positive emotions are associated with high energy and vice-versa for negative emotions.

10.3.2 MFCC Parameters

The MFCC represents the short-term power spectrum on a nonlinear scale of frequency called the mel scale. The various terms in the mathematical representation are determined using a linear cosine transform from the logarithmic power-spectrum.

MFCCs collectively make up an MFC. In MFC, the frequency bands are equally spaced on the mel-scale, allowing it to correctly emulate the response of a human auditory system.

MFCCs is determined through the mentioned steps (Sahidullah and May 2012) (Figure 10.3):

1. Apply Fourier transforms on any windowed part of the required signal
2. Match the powers of the newly obtained spectrum onto the mel-scale
3. Take the logs of the powers at each of the mel frequencies
4. Determine discrete cosine transform of all the mel log powers
5. The amplitudes of the resulting spectrum give the MFCCs coefficient.

FIGURE 10.3 Schematic of MFCC process.

10.4 CHOICE OF PARAMETERS

10.4.1 Benefits of MFCC: Why MFCC is Used

As with any machine-learning endeavor, the choice of comparative parameters is of prime importance. Therefore, a clear understanding of the key differences between the various emotions is essential. As previously noted, differentiating among emotions is perhaps one of the murkiest and highly debatable fields, although there exist a few areas of mutual agreement—however limited. But one aspect of emotion recognition, which stands despite the challenges, is that every sample of the audio signal is processed by humans, and humans themselves hold the primary authority in categorizing them. This seemingly obvious fact holds great importance once we realize our ears are not equally sensitive to vibration in every frequency. Human ears have a limited capacity and detect frequencies only in a specific range. Additionally, not all frequencies in the audible range are perceived similarly. The higher and lower frequencies are subject to different sensitivities by the human ear.

Therefore, a device emulating the human brain analyzing audio signals for recognition of emotions must perceive the incoming signals as if received by the ear. Therefore, the MFCC parameters are employed for this purpose. Detailed understanding of the MFCC parameters is not required, but a primary knowledge is helpful (Kwon et al. 2003).

10.5 PROGRAMMING TOOLS USED

We are now aware of the various facets of emotions and their recognition (especially through speech) such as different types of emotion, several speech parameters and the benefit of using MFCC over any other parameter. So, let us put this theoretical knowledge towards a practical purpose in order to build a predictive model for emotion recognition using machine learning.

We primarily used the python programming language, along with several tools/modules, for the purpose of extraction of required features from the audio signals and for building our classification model and its evaluation (Casale et al. 2008; Mordkovich, Veit, and Zilber 2011).

10.5.1 Training and Testing Dataset

10.5.1.1 Databases Used

Before we proceed any further with the technical aspects of this project, we first require a dataset especially synthesized for the purpose of emotion detection. Fortunately, there are such datasets present (for free) on the Internet. Unfortunately, most of them became quite small in size because of our decision to only work towards solving the problem of binary classification (happy vs. sad); therefore, we decided to merge two of the available datasets; the details concerning them are stated below:

1. Savee database: This dataset contains voices recorded by four male speakers. Each speaker recorded 15 sentences for each of the seven emotion categories; namely, “anger,” “disgust,” “fear,” “happiness,” “neutral,” “sadness,” and “surprise” (Haq and Jackson 2010).

2. Emo-db corpus (berlin database of emotional speech): This dataset contains about 500 utterances spoken by actors conveying happy, angry, anxious, fearful, bored, disgusted, and neutral emotions. We had the freedom to choose utterances from 10 different actors and 10 different texts (Burkhardt et al. 2005).

Our final dataset contained 250 instances of speech audio, out of which 130 instances conveyed a "happy" emotion and the rest—120, were "sad." Training and testing sets were created from this merged database according to the methodology explained in the following sections.

10.5.1.2 Features and Target Labels

Every machine-learning algorithm requires a dataset for training. Another dataset—called testing dataset, may be used for subsequent evaluation of the model. Both of these sets contain features and target labels (if it is a supervised learning problem). Let us look into the components of the training and testing sets in detail, with respect to our classification problem.

Features: As already discussed, we exclusively used features derived from MFCCs. Using only the first 13 MFCCs for each audio signal, we calculated the mean, maximum, minimum, and variance for each coefficient, as well as its derivative across all frames. We also calculated the mean, variance, maximum, and minimum of the mean for each coefficient and its derivative. Therefore, 112-dimensional feature vector for each audio signal was created accordingly.

Target labels: We encoded our two classes of emotions such that "1" signified the emotion conveyed by the audio signal was happy, and "0" if sad. Therefore, each audio signal in the dataset had a "1" or "0" value associated to it.

The following Python modules were used for preparing this dataset: Librosa (Mcfee et al. 2015), numpy (Oliphant 2007), scikit-learn (Pedregosa et al. 2012), and matplotlib (Hunter 2007) and their dependencies.

10.5.1.3 Train-Test Split

Although it is possible, we cannot use the entire dataset for training our classification model since we won't be able to estimate its performance afterwards. Therefore, a common practice is to divide the dataset into two parts; one is used for training the model and the other for testing it, hence their names. Due to the small size of our database, we decided the split would be 80% and 20% for our training and testing datasets, respectively. The split was stratified, that is, the relative percentage of each class remained the same in both datasets.

10.5.2 Building the Classification Model

After our training and testing datasets were prepared, we needed an algorithm/s using with which we could build our models. The implementation of a large number of machine-learning algorithms are present on the Internet, and since we cannot train and evaluate all of them, we decided to work with the four most prominent ones based on previous research done in this area. These are LR(logistic regression) (Hosmer, Lemeshow, and Sturdivant 2013), SVM (support-vector machines) (Pan, Shen, and

FIGURE 10.4 Comparison of accuracy scores and logloss.

Shen 2012), k-NN (k-nearest neighbor) (Mucherino et al. 2009), and LGBM (light gradient boosted model) (Ke et al. 2017). For the evaluation and optimization of each of these models, a cross-validation technique was used, and then their accuracy and log-loss score (Vovk 2015) was calculated using the test set (refer Figure 10.4).

Results:

Accuracy score:

LGBM $\sim$ 95%
LR $\sim$ 83%

SVM $\sim$ 91%
KNN $\sim$ 85%

Logloss:

LGBM $\sim$ 0.09
LR $\sim$ 0.59

SVM $\sim$ 0.35
KNN $\sim$ 0.24

10.6 APPLICATIONS

10.6.1 HCI: HCI Applications of the System

Emotion recognition systems can be widely employed for increasing the depth of human–computer interaction (HCI). HCI has taken a central stage today, as far as smart and responsive systems are concerned. Without efficient HCI, a smarter world would remain

a dream. Although the advances in the field of smart systems are appreciable, there lies a massive opportunity for improvement mainly due the inability of our systems to understand true human needs. Our machines are only capable of understanding human wants and delivering upon them. We must grasp the fundamental difference between the two. After all, identifying the problem is central to solving it! (Cowie et al. 2001)

It is precisely this avenue that the emotion recognizing systems hope to tap into. Broadly speaking, they can be employed in almost any field of work to replace humans (at some point in the future perhaps), but six key areas have been identified and can be exploited using the present technology. They are:

1. Medicine
2. E-learning
3. Monitoring
4. Entertainment
5. Law
6. Marketing

10.6.2 Medical Applications

There exist numerous medical applications for emotion recognition systems. From direct delivery of treatment to patients to collecting valuable data for studying behavioral sciences, they can find uses in every realm imaginable (Lalitha et al. 2014).

Some of the main applications include:

1. Patient monitoring systems—they can be employed for keeping a 24/7 vigilance of patients, thus aiding in easier rehabilitation.
2. Companion bots—although JARVIS, from the popular sci-fi feature *Iron Man,* is still a distant dream, the emotion recognizing systems are a definite step towards it. Companion bots can be upgraded to understand human emotions making them capable of taking smarter and more important decisions. They can prove to be invaluable to the user, not only in performing certain tasks but also talking to them intelligently, thereby enhancing realism.
3. Counseling—perhaps the need of the hour, a smart counseling system can democratize access to a psychiatrist in the remotest corners of the world. Long-term analysis of a person's mental state can be valuable data for doctors who often need such information to make a better diagnosis.
4. Autism support—millions suffer from autism across the world, and despite sincere attempts some key challenges remain unsolved. Among them is the inability or difficulty autistic persons have to interpret emotions. This is a case efficiently tackled by these systems, which not only identify emotions for the user, but also help them with therapy.
5. Music therapy—this form of therapy has been clinically proven to induce appreciable degrees of positivity. It can help patients with the anxiety, depression, and stress of dealing with their conditions. Also, music therapy is advised for Alzheimer's patients.

10.6.3 Responsive Systems

A responsive device is often used to deploy an emotion recognition system, especially for HCI applications. There exist a variety of application of such systems including:

1. E-learning—this is a growing field but it is nothing revolutionary per se; however, smarter HCI can change that. Automated learning systems able to track the mood of the learner can deliver more interesting content to the student. Such systems, taking individual feedback from the students, can change their pace or method of teaching. These measures are impossible in a regular class. Even with personal tutors, human errors are bound to creep in making content curation a tough nut to crack (Shen, Wang, and Shen 2009).
2. Monitoring—a quick response, based on the anomalous behavior of a user, can be easily provided with the use of systems able to identify emotional states such fear or anger. These kinds of solutions find extensive applications in call centers and feedback systems, where angry customers can be dealt with more sensitively.
3. Entertainment systems—the true depth of scientific advancement is felt only when the results trickle down to the common person through everyday applications. Undoubtedly, entertainment devices including music players, TVs, video games, etc. are an indispensable part of our lives. Therefore, any disruptive technology must be able to bring changes in the way we interact with entertainment. Interestingly, all these entertainment devices can tune to the users' liking by using emotion recognition software. For instance, music players can suggest songs based on the mood of the user; the scriptwriter of a television series can study how the audience reacts to the plot, and modify it accordingly. Even recommendation engines can detect the mood of the users to suggest books or movies based on the current state of mind of the user. This not only improves the suggestions, increasing their relevance, but also eliminates the cold start problem of today's recommendation systems.
4. Marketing—just as in the case of television content writers, advertisement makers can readily use the data pertaining to the mood shifts of their audience to make better advertisements, which suit the users. The brands can focus on what's relevant to their customers. Today, the companies are mainly dependent on large-scale surveys to determine the likings of customers. This method is not only time consuming but also often ineffective because people can't always tell what they like most about a product—even if they want to. Our choices depend on various conscious and subconscious markers, which can be effectively determined by noticing our moods.

10.7 CONCLUSION

Emotion recognition systems have the ability to humanize digital interactions by creating artificial emotional intelligence in machines. Since our interactions with technology are becoming increasingly conversational, artificial emotional intelligence has become a critical component of today's technology. Although a lot

of work is going in this field of study and some amazing breakthroughs have been made, we have only scratched the tip of the iceberg. A lot of questions still remain unanswered and newer challenges crop up as we delve deeper into this field of study. However, these systems have the potential of disrupting industries from health care to education, and anything in between. Emotion recognition systems will not only revolutionize human interactions with machines but also allow machines to be more human.

REFERENCES

Bales, R.F. 2000. "Social Interaction Systems: Theory and Measurement: Book Review." *Group Dynamics: Theory, Research, and Practice* 4 (2):199–208. https://doi.org/10.1037/1089-2699.4.2.199.

Burkhardt, F., A. Paeschke, M. Rolfes, W.F. Sendlmeier, and B. Weiss. 2005. "A database of german emotional speech." In *Interspeech*, 5:1517–1520.

Casale, S., A. Russo, G. Scebba, and S. Serrano. 2008. "Speech Emotion Classification Using Machine Learning Algorithms." *2008 IEEE International Conference on Semantic Computing* 118 (13):167–74. https://doi.org/10.1109/ICSC.2008.43.

Colombetti, G. 2009. "From Affect Programs to Dynamical Discrete Emotions." *Philosophical Psychology* 22 (4):407–25. https://doi.org/10.1080/09515080903153600.

Cowen, A.S., and D. Keltner. 2017. "Self-Report Captures 27 Distinct Categories of Emotion Bridged by Continuous Gradients." *Proceedings of the National Academy of Sciences*, 201702247. https://doi.org/10.1073/pnas.1702247114.

Cowie, R., E. Douglas-Cowie, N. Tsapatsoulis, G. Votsis, S. Kollias, W. Fellenz, and J.G. Taylor. 2001. "Emotion Recognition in Human-Computer Interaction." *Signal Processing Magazine, IEEE* 18 (1):32–80. https://doi.org/10.1109/79.911197.

Ekman, P. 1992. "An Argument for Basic Emotions." *Cognition and Emotion* 6 (3–4):169–200. https://doi.org/10.1080/02699939208411068.

Haq, S., and Jackson, P.J.B. 2010. *Machine Audition: Principles, Algorithms and Systems*, 532. https://doi.org/10.4018/978-1-61520-919-4.

Hosmer, D.W., S. Lemeshow, and R.X. Sturdivant. 2013. *Applied Logistic Regression*, Third Edition. Wiley Series in Probability and Statistics. https://doi.org/10.1002/0471722146.

Hunter, J.D. 2007. "Matplotlib: A 2D Graphics Environment." *Computing in Science and Engineering* 9 (3):99–104. https://doi.org/10.1109/MCSE.2007.55.

Ke, G., Q. Meng, T. Wang, W. Chen, W. Ma, and T.-Y. Liu. 2017. "A Highly Efficient Gradient Boosting Decision Tree." *Advances in Neural Information Processing Systems* 30 (Nips):3148–56. http://papers.nips.cc/paper/6907-a-highly-efficient-gradient-boosting-decision-tree.pdf.

Kwon, O.-w., K. Chan, J. Hao, and T.-w. Lee. 2003. "Emotion Recognition by Speech Signals." *Eighth European Conference on Speech Communication and Technology*, 125–28. http://ergo.ucsd.edu/~leelab/pdfs/ES030151.pdf.

Lalitha, S., A. Madhavan, B. Bhushan, and S. Saketh. 2014. "Speech Emotion Recognition." *2014 International Conference on Advances in Electronics, Computers and Communications (ICAECC)* 7 (Table I):1–5. https://doi.org/10.1109/ICAECC.2014.7002390.

Lazarus, R.S., and B.N. Lazarus. 1996. *Passion and Reason*. Oxford University Press, Washington, DC.

Mcfee, B., C. Raffel, D. Liang, D.P.W. Ellis, M. Mcvicar, E. Battenberg, and O. Nieto. 2015. "Librosa: Audio and Music Signal Analysis in Python." *Proc. of The 14th Python in Science Conf*, no. Scipy:1–7.

Mordkovich, A., K. Veit, and D. Zilber. 2011. "Detecting Emotion in Human Speech." http://cs229.stanford.edu/proj2011/VeitZilberMordkovich-DetectingEmotionInHumanSpeech.pdf.

Moriyama, T., H. Saito, and S. Ozawa. 2001. "Evaluation of the Relation between Emotional Concepts and Emotional Parameters in Speech." *Systems and Computers in Japan* 32 (3):56–64. https://doi.org/10.1002/1520-684X(200103)32:3<56::AID-SCJ5>3.0.CO;2-B.

Mucherino, A., P.J. Papajorgji, and P.M. Pardalos. 2009. "K-nearest neighbor classification." In *Data Mining in Agriculture*: 83–106, Springer, New York, NY.

Oliphant, T.E. 2007. "Python for Scientific Computing." *Computing in Science and Engineering* 9 (3):10–20. https://doi.org/10.1109/MCSE.2007.58.

Pan, Y., P. Shen, and L. Shen. 2012. "Speech Emotion Recognition Using Support Vector Machine." *International Journal of Smart Home* 6 (2):101–8. https://doi.org/10.5120/431-636.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2012. "Scikit-Learn: Machine Learning in Python." *Journal of Machine Learning Research* 12:2825–30. https://doi.org/10.1007/s13398-014-0173-7.2.

Plutchik, R. The Nature of Emotions. American Scientist. Archived from the original on July 16, 2001.

Posner, J., J.A. Russell, and B.S. Petersona. 2005. "The Circumplex Model of Affect: An Integrative Approach to Affective Neuroscience, Cognitive Development, and Psychopathology." *Dev Psychopathol* 17 (3):715–34. https://doi.org/10.1007/s10955-011-0269-9.Quantifying.

Rubin, D.C., and J.M. Talarico. 2010. "NIH Public Access" 17 (8):802–8. https://doi.org/10.1080/09658210903130764.A.

Sahidullah, M., and G. Saha. 2012. "Design, Analysis and Experimental Evaluation of Block Based Transformation in MFCC Computation for Speaker Recognition." *Speech Communication* 54 (4):543–65. https://doi.org/10.1016/j.specom.2011.11.004.

Schlosberg, H. 1954. "Three Dimensions of Emotion." *Psychological Review* 61 (2):81–88. https://doi.org/10.1037/h0054570.

Shen, L., M. Wang, and R. Shen. 2009. "Affective E-Learning: Using 'emotional' Data to Improve Learning in Pervasive Learning Environment Related Work and the Pervasive E-Learning Platform." *Educational Technology & Society* 12:176–89. https://doi.org/citeulike-article-id:7412147.

Vovk, V. 2015. "The Fundamental Nature of the Log Loss Function." *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9300:307–18. https://doi.org/10.1007/978-3-319-23534-9_20.

Watson, D., and A. Tellegen. 1985. "Toward a Consensual Structure of Mood." *Psychological Bulletin*. https://doi.org/10.1037/0033-2909.98.2.219.

Wundt, W. 2009. "Outlines of Psychology (1897)." *Found. Psychol. Thought A Hist. Psychol.* 36–44. https://doi.org/10.1007/978-1-4684-8340-6_7.

11 Segmentation and Clinical Outcome Prediction in Brain Lesions

Sharmila Nageswaran, S. Vidhya, and Deepa Madathil

CONTENTS

11.1 Introduction 181
11.2 Background 182
11.2.1 Brain Lesion 182
11.2.2 Classification Methods 183
11.2.2.1 Parametric Classification 183
11.2.2.2 Nonparametric Classification 184
11.2.3 Regression Methods 184
11.2.4 Neural Network 184
11.3 Segmentation of Brain Lesion 186
11.4 Outcome Prediction of Brain Lesion 187
11.5 Case Studies 187
11.5.1 Ischemic Stroke Segmentation Challenge 188
11.5.2 mTOP (Mild Traumatic Brain Injury Outcome Prediction) Challenge 189
11.6 Further Reading 190
Acknowledgment 190
Disclosure 190
References 190

11.1 INTRODUCTION

The diagnosis and management of brain lesions have been most challenging for many decades. Understanding of the brain is another domain, or challenge which has multiple focuses. To treat any brain lesions, the disease or condition needs to be identified precisely. It is not possible to confirm the diagnosis by an invasive procedure when it happens to be a neural system and so is the brain. Noninvasive and precise diagnostic procedures are directly needed and prediction of treatment's outcome is what is prediction of prognosis is based on the estimation of pathology done using these non invasive procedures. Hence, diagnostic imaging plays a vital role in deciding the treatment and its outcome. Segmentation and clinical outcome prediction has been a topic of interest for clinicians, health care practitioners, patients,

and researchers for many decades. However, the only outcome of importance to the patient is the understanding about the disease and its progression. It will help them plan their treatment and future.

This chapter provides an insight into the segmentation of brain images from different perspectives. It is aimed to bring together specialized researchers from different disciplines who can deal with problems from an entirely different perspective but share a common aim towards this book. The chapter is divided into the following five sections:

1. Introduction
2. Background
3. Segmentation of brain lesion
4. Outcome prediction of brain lesion
5. Case studies

11.2 BACKGROUND

A lesion can be defined as a region in a tissue, or an organ, which has been damaged through disease or injury. Diagnosis of lesions in real time, using reliable algorithms, has been the main focus of the latest developments in medical image processing. In this area, the major focus of research has been the detection of lesions using MR (magnetic resonance) and CT (computed tomography) images. Usually CT and MRI are used to examine the anatomy of brain lesions. Magnetic resonance imaging (MRI) is a technology used in biomedical applications to detect and visualize minute details of the internal body structure. This technique detects differences in the tissues, and is a far better technique compared to CT in terms of image properties and quality. Cancer imaging and lesion detection are major applications related to MRI [1]. Since MRI provides more accurate results and does not involve any radiation, it has an advantage over CT scans. MRI is a technology using magnetic fields and radio waves.

The ability to separate the nuclei and the cells from the rest of the image content is one of the most important abilities in medical imaging and diagnostic systems. This process is called "segmentation." It is the most important factor in the design of an effective and robust diagnostic system. It is more efficient to segment an image with lesion than the one without a lesion. In medical image systems, segmentation of 3D images play a vital role which occurs before the implementation of algorithms for its segmentation based on the recognition of a type of lesion. on recognition of the lesion. Many statistical parameters like the size, volume, density, etc. can be inferred with accuracy from segmented 3D images. The accuracy in measurement of the size, shape, and appearance of the lesion is an important parameter in the diagnosis of brain lesions.

11.2.1 Brain Lesion

Brain lesions do not have a predefined shape often present a quite abnormal structure, making them difficult to detect and measure. It is therefore very important, from

a researcher's point of view, to have an automated software package having the capability to detect brain lesions of any shape, volume, and size . There are a few important types of brain lesions, which need to be understood before processing MR or CT images. The types of brain lesion are:

1. Traumatic
2. Malignant
3. Benign
4. Vascular
5. Genetic
6. Immune
7. Plaques
8. Brain cell malfunction or death and
9. Ionizing radiation

The most common types are the traumatic, benign, and malignant lesions. The traumatic lesion is a lesion occurring due to an injury such as a road accidents or a gunshot wound on the brain, etc. The benign lesion typically does not expand in an abrupt way and does not further affect the healthy neighboring tissues, also it does not expand to the nonadjacent tissues. A malignant lesion grows with the passage of time and ultimately results in the death of the patient. Severe progressing disease is the term used for malignant lesions by medical practitioners. It is typically used for the description of a cancer.

11.2.2 Classification Methods

Accurate detection of lesion in the brain is important for humans. Additionally, the lesion has to be classified precisely. Automated detection and prediction of a lesion is something expected from every clinical setup. In medical imaging, after the image is captured by the imaging system it is digitized and sent to a computer. It is then segmented and processed further to obtain features from the segmented image. In brain images, the fundamental task is to obtain useful features from the white matter, gray matter, and cerebrospinal fluid. Manual segmentation is a traditional method used for segmenting a MR image, which is more time-consuming. Segmentation approaches vary from among experts [2]. Therefore, there is need to have an efficient and strong computer-based system to accurately examine the brain lesion statistics. The success of classification of a lesion completely depends on the accuracy of segmentation. Classification can be parametric or nonparametric.

11.2.2.1 Parametric Classification

When the estimated output of segmentation has more similarity to the data available from a known class, or set of well-classified lesions, then the approach becomes parametric. The goal is fixed in most of the parametric approaches and the learning of the classifier network is usually supervised, requiring an abundant data set available for training and testing. Most benign lesions fall into this category.

11.2.2.2 Nonparametric Classification

When the estimated output is new, and if multiple conditions of a particular pathology are to be listed with a similar output, then a new class is formed and added to the available set of parametric classes. However, when there is no existence of another similar estimation, then it paves the way for a nonparametric approach. This kind of classification is based upon the fact that whatever is deviating from the available models will all be collectively grouped under nonclassified models. Based on the responses from experts (neurosurgeon or neuroradiologist), they will be classified as a particular type of lesion. Certain malignant tumors and disease conditions expected to have unpredicted courses of progression will fall into this category.

11.2.3 Regression Methods

Regression methods or analysis is a form of predictive modeling, which brings out a relationship or correlation between a dependent and an independent variable. Autoregression or AR modeling is the most commonly used regression method in biological studies. AR is a stochastic approach used in statistical calculations. Here, the future values are estimated based on a weighted sum of past values. An autoregressive process operates under the premise that past values have an effect on current values, and current values will have an effect on future values. A series of images taken during the course of a lesion will help in predicting the outcome. This prediction method may be based on a linear, stepwise, or parabolic regression analysis. All this would completely be determined by the set of predictions following the segmentations. These regression analyses are thus used in predicting the outcome and recovery from a stroke. Therefore, it helps the clinicians in the proper planning and execution of the treatment.

11.2.4 Neural Network

Neural networks have met with a considerable amount of interest in predictive methods like regression, classification, and artificial neural networks (ANNs) for outcome prediction. This is because they offer a number of theoretical advantages. ANNs have a much wider range of flexibility to fit models to the available data when the patterns are not very obvious [3]. Apart from this, the major advantages of using ANNs in modeling are that nonlinear and random relationships can be seen between independent and dependent datasets. Also, they do not require any kind of distributional assumption. Furthermore, they also allow interactions between dependent variables. A large number of clinicians have aware of the power of ANNs as a medical analysis tool. The main issue called the “black box” has been stated by a number of statistical framework developers for ANNs [4–7]. The success of any new technology—for its integration into clinical practice—depends mostly on the number of trails in the published literature. Apart from having a large number of publications reporting on the use of ANNs in medicine, the number of clinical trials remains to be less [8–10].

ANNs have found application in nonlinear signal processing, classification, and optimization. In many of these applications their performance has been considered as superior to the classical and traditional linear approaches. In processing of biomedical

signals like ECG and EEG, the most common networks used are multilayer perceptron (MLP), learning vector quantization networks, and radial basis function (RBF). The perceptron consists of neurons interconnected to different layers in which each neuron is a represented as a function given as:

$$y = f\left(w_0 + \sum_{i=1}^{N} w_i x_i\right)$$

where w_i is the assigned weight to the input x_i, and f is a function, which can be linear or nonlinear. In the case of nonlinearity, f is defined frequently as the logistic function, which is given by

$$f(u) = 1/(1+e-u) \quad \text{or} \quad f(u) = \tanh(u)$$

In a similar way, networks with RBF are implementations given by a function as,

$$y(n) = \sum_{i=1}^{N} w_i \exp\left(-\frac{x(n)-c_i}{\sigma_i}\right)$$

where *x(n)* denotes an input data vector given to the network. *N* is the number of neurons, and w_i is the coefficients, c_i is the center vector, and σ_i are the standard deviations, which denote the network parameters. The exponential can be replaced by functions like wavelets. In Figure 11.1, an MLP based neural network consisting of three layers is shown. Eight to ten linear neurons are present at the input layer, which are signals delayed by time and are samples of inputs. Also, there are three to five logistic nonlinear neurons in the hidden layer and the output layer contains a linear neuron.

In some methods, where the network is trained before the detection is done using selected samples while in other networks, online training is done. This would be the reason for the ability of neural networks to adapt to signal statistics. RBF networks are very closely related to fuzzy logic methods. The RBF networks have advantages over the MLP networks in their ability in predicting the network parameters. This helps to produce reliable and more predictable results.

FIGURE 11.1 Typical MLP network.

11.3 SEGMENTATION OF BRAIN LESION

It is the most challenging task to segment the lesion from its background, especially when the lesion is in multiple locations. Following a good technique during the data acquisition will ease the complexity involved with the processing. However, in medical imaging technology, the dosage of exposure is strictly controlled, as to avoid the ill effects of overexposure to radiation. Hence, more sophistication is expected in processing. The stages of image processing are given in Figure 11.2.

Segmentation of a hemorrhagic stroke image is not very challenging compared to segmentation of the image of an ischemic stroke. The former has a wider intensity variation. The clot or hemorrhage, which is rich in hydroxyl ions, responds well to the MRI and is captured as white region whereas the rest of the brain appears to be grey. Because of this demarcation segmentation is comparatively simpler. However, based on the strength of the magnetic field used in the acquisition machine, contrast enhancement will be done in preprocessing. A sobel operator and a median filter are

FIGURE 11.2 Stages of the brain image processing.

commonly used in preprocessing, and logical operators are widely used for feature extraction. By varying the threshold of these operators, different regions and different contrasts can be segmented.

Segmentation of an ischemic stroke image is more challenging, as there is no well-defined boundary of lesion or an absence of wider variation between the intensities in the regions of interest and those surrounding them. Because of atrophy the cell lacks hydroxyl ions and will show up as darker regions, while the rest will show up as grey. Furthermore, there is no fixed intensity level for the other regions, hence it is difficult to understand if the darker shades in other regions are because of illness or due to lower hydroxyl ion content (within physiological limits as a result of aging or any other disease). This problem can be minimized to a good extent by a suitable preprocessing method. Therefore, the precision in segmentation now depends on the effectiveness achieved in contrast enhancement. Usage of an appropriate threshold will further reduce the complexity of segmentation.

Classification of a lesion is purely based on segmentation. Based on the accuracy of segmentation, the classification would be ideal. As discussed earlier, it could either be parametric or nonparametric.

11.4 OUTCOME PREDICTION OF BRAIN LESION

The accuracy of prediction of clinical outcome is again based on the methods adapted for a lesion segmentation and classification. Other than the type of lesion, it is usually the area, volume, depth information, and its location, which are predicted by image processing methods. This would help the clinicians to plan for the treatment, surgery, follow up, and expected prognosis of a management protocol. The knowledge base linked with outcome prediction will have to undergo frequent upgrading so as to include newer types of lesions or illnesses. Sometimes in the clinical setup doctors feel that every lesion is unique. These unique lesions are the most vulnerable for false prediction, as there will not be any relevant data to confirm its prediction theory. Among the various brain lesions, the least challenging ones are tumors, while the most challenging ones are congenital and degenerative disorders combined with ischemic disorders.

11.5 CASE STUDIES

Any research topic is incomplete without its supporting documents, and case studies are the most significant reports with regards to any writing on image processing. Clinically, strokes can present as (1) hemorrhagic strokes—due to trauma or cardio vascular accidents following hypertension, and (2) ischemic strokes—due to vasoconstriction, extreme hypotension, hypoglycemia, and myocardial infarction leading to emboli lodged in cerebral vessels. Stroke or hemiplegia is the partial paralysis of the contralateral side of the body with respect to its controlling cerebral hemisphere. Irrespective of the type or cause the presentation of the condition will be paralysis, and hence they are collectively described as "stroke." Although the result is paralysis, the management of the stroke and prevention of its reoccurrence is based on the type of stroke. Hence, it is very significant to identify the type of stroke very

precisely. The classification of the lesion is the crucial step, which determines the success rate of the treatment.

11.5.1 Ischemic Stroke Segmentation Challenge

The MRI image of an ischemic stroke is shown in Figure 11.3. The image shows certain dark regions marked within a circle in Figure 11.3a, which is identified as the region with the lesion, or the region of interest. This is often misread as atrophy of the brain, which could either be pathological or physiological as in degeneration due to aging. However, in the case of ischemic strokes there exist mild difference in the contrast between the region of interest and its surrounding. The contrast variation is used as a basis for further processing in image segmentation. However, this difference is well appreciated in case of acute lesions. In the case of a chronic lesion, the demarcation eventually reduces. The reason could be that the surrounding region is also becoming involved in the lesion, as the cause is not treated until diagnosed. As with any segmentation process, the first step is to reduce the noise and enhance the contrast. The same is done in this case too!

Preprocessing is done by region growing followed by contrast stretching, as the original image has normal regions, as well as regions with lesions in similar grey levels. Logical operations were performed with zero padded image, and image compliment was done using thresholding. Finally, image cropping was achieved

FIGURE 11.3 (a) Original image, (b) region growing, (c) contrast stretching, (d) logical operation and binary conversion, (e) image compliment and thresholding, and (f) segmented image.

with boundary conditions. This will segment the region of interest. All the images produced during this process are shown in Figure 11.3.

11.5.2 mTOP (Mild Traumatic Brain Injury Outcome Prediction) Challenge

Traumatic brain injuries are not very challenging in terms of lesion prediction but when the lesion is mild it is seldom challenging. The challenges can be overcome by appropriate thresholding and processing.

Figure 11.4a shows a hemorrhagic stroke image used for segmentation. As there is a variation in the image contrast, stretching is done as a part of preprocessing. The outcome of preprocessing yields an image having good demarcation between the region of interest and other regions. Therefore, image compliment is done to get an image showing further variation in contrast. As a part of the segmentation process, binary conversion with suitable thresholding is carried out to obtain appropriate segmentation. All the images rendered during this process are shown in Figure 11.4 [11].

FIGURE 11.4 (a) Original image, (b) contrast stretching, (c) image complement, and (d) image segmentation.

11.6 FURTHER READING

Various researchers have worked on different aspects of image processing. Nevertheless, every image is unique and challenging, and thus processing has to be performed discretely for every image. Based on the requirement, analyzing the contrast level, histogram of the image to be processed, a researcher can read the relevant work published by an author in the same domain. A few authors have done segmentation based on image texture [11,12] while few others have done it based on contour. It all depends on the specific image.

ACKNOWLEDGMENT

The authors would like to thank Kondaveti Revanth and Abhishek Patil, who have contributed to the topics on case study and MLP network, respectively.

DISCLOSURE

All the original Images were taken from https://emedicine.medscape.com/article/338385-overview; "Stroke Imaging," accessed on December 4, 2017. This is an open source article and the authors and chief editor have nothing to disclose. Terms and Policy of the site had no objection statement relating to the reuse of these images. The MATLAB 2016 version of software was used to process the images.

REFERENCES

1. Devos, A. and L. Lukas, 2004, Does the combination of magnetic resonance imaging and spectroscopic imaging improve the classification of brain tumors?, On Page(s): 407–410, Engineering in Medicine and Biology Society, 2004. *IEMBS '04. 26th Annual International Conference of the IEEE*, Sept 1–5, 2004.
2. Masroor Ahmed, M. and D.B. Mohamad, 2008, Segmentation of brain MR images for tumor extraction by combining Kmeans clustering and PeronaMalik Anistropic diffusion model, *Int. J. Image Process.* 2(1), 27–34.
3. Ripley, B.D., 1996, *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, U.K.
4. Biganzoli, E., P. Boracchi, L. Mariani and E. Marubini, 1998, Feed forward neural networks for the analysis of censored survival data: A partial logistic regression approach. *Stat. Med.* 17(10), 1169–1186.
5. Ripley, B.D. and R.M. Ripley, 2001, Neural networks as statistical methods in survival analysis. In: Dybowski, R. and V. Gant (Eds.), *Clinical Applications of Artificial Neural Networks*. Cambridge University Press, Cambridge, pp. 237–255.
6. Lisbosa, P.J., H. Wong, P. Harris and R. Swindell, 2003, A Bayesian neural network approach for modelling censored data with an application to prognosis after surgery for breast cancer. *Artif. Intell. Med.* 28(1), 1–25.
7. Bishop, C.M., 2004, Error functions. In: *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, U.K., pp. 230–236.
8. Gant, V., S. Rodway and J. Wyatt, 2001, Artificial neural networks: practical considerations for clinical applications. In: Dybowski, R. and V. Gant (Eds.), *Clinical Applications of Artificial Neural Networks*. Cambridge University Press, Cambridge, U.K., pp. 329–356.

9. Lisboa, P.J., 2002, A review of evidence of health benefit from artificial neural networks in medical intervention. *Neural Netw.* 15(1), 11–39.
10. Lisboa, P.J. and A.F.G. Taktak, 2006, The use of artificial neural networks in decision support in cancer: A Systematic Review. *Neural Netw.* 19, 408–415.
11. Ali, A.H., S.I. Abdulsalam and I.S. Nema, 2015, Detection and segmentation of hemorrhage stroke using textural analysis on brain CT images. *Int. J. Soft Comput. Eng. (IJSCE)* ISSN: 2231-2307, 5(1), 11–14.
12. Ali, A.H., S.I. Abdulsalam and I.S. Nema, 2015, Detection and segmentation of ischemic stroke using textural analysis on brain CT Images. *Int. J. Sci. Eng. Res.*, ISSN 2229-5518, 6(2), 396–400.

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

12 Machine Learning Based Hospital-Acquired Infection Control System

Sehaj Sharma, Prajit Kumar Datta, and Gaurav Bansal

CONTENTS

12.1 Introduction 194
12.2 Types of NI 194
12.2.1 Central Line-Associated Bloodstream Infections (CLABSI) 195
12.2.2 Catheter Associated Urinary Tract Infections (CAUTI) 196
12.2.3 Surgical Site Infections (SSI) 197
12.2.4 Ventilator Associated Pneumonia (VAP) 198
12.3 NI Programs 199
12.3.1 The Canadian Nosocomial Surveillance Program (CNISP) 199
12.3.2 The Failure of a NI Program in a Hospital 200
12.3.3 The Success of a NI Program in a Hospital 200
12.3.4 NI Control Methods 201
12.3.4.1 Standard Precautions 201
12.3.4.2 Transmission-Based Precaution 201
12.3.4.3 Immunization and Vaccination Programs 201
12.3.4.4 Education and Training of Health Care Staff 202
12.3.5 Antimicrobial Use and Resistance 202
12.3.6 Antibiograms to Control Antibiotic Resistance 203
12.3.7 Cross-Infection Programs 203
12.4 ML and NIs 203
12.4.1 Type of Data 204
12.4.1.1 Structured Data 204
12.4.2 Unstructured Data 205
12.4.3 Data Gathering Activity 205
12.4.4 Data Sources 205
12.4.5 Variety of Data 206
12.4.6 Data Collection Period 206
12.4.7 Data Collection Locations 206
12.4.8 Data for All Output Types 206
12.4.9 Data Exclusions 207
12.4.10 Data From Interview 207

12.4.11 The Imbalanced Data Problem 207
12.4.12 Data Preparation Activity 207
12.4.12.1 Handling Missing Values..... 207
12.4.12.2 Removing Redundant Variable..... 208
12.4.13 Implementation 208
12.4.13.1 One Class Classification Implementation..... 208
12.4.13.2 One Class Classifier: SVM..... 208
12.4.13.3 One Class Classifier: Parzen Density Estimator..... 210
12.4.13.4 Text Classification Method 211
12.4.13.5 Linear Regression and Artificial Neural Network..... 213
12.5 Conclusion 213
References..... 214

12.1 INTRODUCTION

The goal of every health care institution is to discharge you healthier than you came in. However, this goal is often not accomplished because of HAI, also known as NI. HAI or NI is a category of infections a patient acquires upon visiting or being admitted to a health care facility. These NI's happen due to several reasons like lack of sanitation in hospitals or clinics, health care facilities becoming a congregation of pathogens due to several patients with a variety of illnesses, and in general suppressed immunities of patients making them more susceptible to NI. A World Health Organization (WHO) study—across developed and developing countries, reported that approximately 8.5% of patients admitted suffered some form of NI.

ML is a very popular buzzword in scientific circles. The applications of it are ubiquitous. ML is an application of Artificial Intelligence (AI) ,which provides the system the ability to learn patterns from the data provided to it, and then to make decisions and forecasts based on it. This chapter examines the various attempts by several researchers to manage NI better.

The chapter aims at educating researchers in reasonable depth about NI and the various ML attempts by several researchers to manage and control NI. This should help the reader develop further NI control programs. The types of NI are discussed with their specific causes. The chapter will then lucidly discuss the various NI control programs and their successes and failures. The final part of the chapter discusses the various ML attempts on NI-related data, and the implication of their results. It is hoped this chapter educates and inspires the reader to conduct novel work on solving the problem of NI.

Please note that the terms HAI and NI will be interchangeably used throughout this chapter, but they mean and imply the same thing.

12.2 TYPES OF NI

This section of the chapter describes the various types of NI. Discussing this is especially important since it will help the reader gain a comprehensive perspective

of the problem. Discussing the problem in depth is aimed to help the reader develop techniques and protocols that are unique and generalised depending on the demographic he/she is targeting.

Section 12.2.1 will discuss four major categories of NIs in reasonable depth.

12.2.1 Central Line-Associated Bloodstream Infections (CLABSI)

CLABSI is a category of NI reporting the highest mortality rates [1–3,8]. Approximately 18% of the patients infected with CLABSI succumb to it [8]. Central lines are tubes, which provide a patient with nutrition, fluids, and medicines in the case the patient isn't able to consume them orally. Catheters are used for hemodynamic monitoring, plasmapheresis, apheresis, hemodialysis, tissue and organ transplantation, administration of liquids, blood products, chemotherapy, antibiotics, and parenteral nutrition [3]. Technically, a central line is a catheter typically attached to a patient's pulmonary artery or vein. Central lines are also called central venous catheters (CVCs). There are five types of central lines or CVCs:

i. Nontunneled CVC [10]: This catheter is placed into a large vein near a patient's neck, chest, or groin. They are usually used for long-term intravenous therapy [3]. They are the most common CLABSI causing CVC [3].
ii. Pulmonary artery catheters (PAC) [3,11]: A PAC is inserted through a central vein into the right side of the heart and into the pulmonary artery. PAC's are customarily used to quantify stroke volume, intracardiac pressures, etc. Unlike other CVCs, which are therapeutic devices, PAC's are diagnostic tools. Fortunately enough, the studies have observed PACs do not increase the chances of NI or mortality [11].
iii. Peripherally inserted CVC (PICC) [3,12,13]: PICCs are a fairly recent type of CVC. Unlike other CVCs inserted from the neck, chest, or groin, which were a source of great discomfort, PICCs are inserted via the veins in the forearm with the help of ultrasound or fluoroscopy [12]. They are believed to have a lower incidence of CLABSI [3]. However, this claim is disputed by Johansson et al. [12].
iv. Tunneled CVC [3,14]: Unlike nontunneled CVC, the cuff of a tunneled CVC is under the skin. This helps secure the catheter in place and prevents infection [14]. A tunneled CVC is reported to have lower incidence of CLABSI [3].
v. Totally implantable venous access devices (TIVAD) [15]: This CVC in particular was reported to have the lowest incidence of CLABSI and also gives the patient a greater degree of freedom. It is the technique of choice for cancer patients requiring chemotherapy [15].
vi. Umbilical catheters: An umbilical catheter is inserted in one of the two main arteries of the umbilical cord. It is used mostly in newborn babies to provide them with the necessary fluids, nutrition, and medicines.

A mass scale European study suggested that nearly 55% of the patients admitted were prescribed a central line [9]. CLABSI occurs when microorganisms move to the CVC tip [3]. They reach the CVC tip in the following ways:

i. From the microbes on patients own skin [3].
ii. An infected infusate [3].
iii. A pathogen already present in the patient's blood, which infects the patient when his immunity is low during a disease or postsurgery.
iv. Contamination by health care workers during preparation of fluids [3].
v. Due to inadequate sanitization while administering the CVC [3].

The most common microorganisms causing CLABSI are coagulase-negative staphylococci, Staphylococcus aureus, Enterococci, Enterobacteriaceae, Pseudomonas, and Candida [3].

The spread of CLABSI can be controlled by:

i. Catheter material chosen, coated with antiseptic, etc. [16].
ii. Catheter insertion site. The subclavian site is preferred [16].
iii. Ultrasound guided placement [16].
iv. Sterilization of the person inserting the catheter. Wear masks, caps, disinfect hands, etc. [16].

12.2.2 Catheter Associated Urinary Tract Infections (CAUTI)

This is the most commonly contracted NI [4]. Urinary transmitted infections (UTI) account for up to 13% or more of all reported NI's. CAUTIs are usually caused by bacteria native to the patient's urinal tract. In normal circumstances, this bacteria is flushed out by the flow of urine. However, catheters have imperfect drainage of urine and create the perfect environment for bacteria to flourish. This can develop into several diseases like orchitis, epididymitis, and prostatitis in males, and pyelonephritis, cystitis, and meningitis in all patients [1]. The two types of CAUTI are [17]:

i. Asymptomatic bacteriuria (ASB): Asymptomatic bacteriuria refers to bacteria in the urine at levels often regarded as clinically consequential (>100 000 colony composing units per milliliter of urine) in patients with no symptoms suggestive of urinary tract infection. It becomes more prevalent with age. This situation is almost universal with catheterized patients [18].
ii. Symptomatic urinary tract infection [17]: Alongside asymptomatic bacteriuria, a patient suffering from symptomatic UTI will also suffer from high fever.
iii. Bacterimia [17]: Less than 3% of patients suffering from ASB progress into bacterimia. It is a bloodstream infection and can cause mortality on a few occasions.

The most common organisms causing CAUTI are Escherichia coli, Enterococci spp, Staphylococcus, Pseudomonas aeruginosa, Candida, and Proteus mirabilis. The spread of CAUTI can be controlled by:

i. Selection of catheter: Catheters with the smallest gauge should be preferred. Interestingly, disinfected catheters didn't show a reduction in CAUTI [17].
ii. Catheter insertion must be done with clean hands, sterilized equipment, barrier precautions, and other general hygiene practices [17].
iii. Catheter maintenance carried out with appropriate hand hygiene, closed drainage system, replace system if it breaks during cleaning, and secure the catheter properly [17].

12.2.3 Surgical Site Infections (SSI)

SSI is the second-most common NI [1,5]. Within the United States alone 300,000 to 500,000 SSI's occur annually [5]. Between 5% and 10% of patients develop an SSI in the United Kingdom [18]. SSIs cause a financial setback of up to GBP 1 billion annually on the National Health Services [18]. Despite the hygiene of hospitals and procedures being impeccable in the developed countries, the incidence of SSI is high in those countries. We leave it to the reader's imagination what this plight is in developing and underdeveloped nations where hygiene isn't prioritized. Nearly 77% of the deaths associated to surgery are causes by SSIs [18]. The types of SSI are:

i. Superficial incisional: These SSIs happen at the level of the skin and subcutaneous tissue. They are indicated by redness, pain, heat, or swelling at the site of the incision, or by the presence of pus [18].
ii. Deep incisional: These SSIs occur at fascial and muscle layers. They are designated by presence of pus or an abscess, pyrexia with tenderness of the wound, or a disseverment of the edges of the incision exposing the deeper tissues [18].
iii. Organ or space infection: These SSIs occur at any component of the anatomy other than the incision opened or manipulated during the surgical procedure, for example the joint or peritoneum. They are designated by a loss of function of a joint, abscess in an organ, localized peritonitis or accumulation. Ultrasound or CT scans substantiate the infection [18].

The causes of SSI are:

i. Patients' own bacterial microflora of skin penetrates the surgical incision [18].
ii. Infected surgical equipment [18].

SSIs are primarily caused by a group of microorganisms called the Gram-positive cocci, mainly Staphylococcus aureus. Gram-negative aerobic bacteria can cause SSIs

after intestinal surgery. Other bacteria can also become a reason for SSI due to the weakened immunity of the patient [18]. The strategies used to control and treat SSI are:

i. Optimization of patient health prior to surgery like stopping smoking, improving nutrition to prevent anemia, etc [18].
ii. Sanitization of surgical site and equipment [18].
iii. Extensive surveillance of NI's [18].
iv. Once infected with SSI, the medical team initiates antibiotic therapy [18].
v. Drainage of pus upon SSI [18].
vi. Cleaning of waste and dead tissue around the surgery wound [18].
vii. Specialist wound care services upon infection [18].

12.2.4 Ventilator Associated Pneumonia (VAP)

VAP occurs with patients put on mechanical ventilators through endotracheal tubes or tracheotomy tube for at least 48 hours [1,6]. There are two major categories of VAP:

i. VAP because of non-Multi Drug Resistant (MDR) Pathogens: These pathogens are fairly manageable to treat. The pathogens that fall under this category are Streptococcus pneumoniae, other streptococcus SPP, Hemophilius influenza, MSSA, Antibiotic—sensitive Enterobacteriaceae, Escherichia Coli, Klebsiella pneumoniae, Proteus SPP, Enterobacter SPP, Serratiamarcescens [6].
ii. VAP because of MDR Pathogens: As evidenced by the category, treating this type of VAP is fairly challenging. The unfortunate fact is that VAP is increasingly associated with MDR. These pathogens aren't recommended to be treated via broad-range antibiotics. The pathogens that fall under this category are Pseudomonas aeruginosa, MRSA, Acinetobacter SPP, antibiotic-resistant Enterobacteriaceae, Enterobacter SPP, ESBL—positive strains, Klebsiella SPP, Legionella, pneumophila, Burkholderiacepacia, and Aspergilius SPP [6].

The strategies to manage and treat VAP are slightly more complex than the rest. To be able to develop protocols to treat VAP, one must understand and further study the currently used strategies, which are [6]:

i. Allowing the patient to breathe through nasal or facemasks, which are not noninvasive.
ii. Short antibiotic courses.
iii. Feed the patient through a catheter, postpyloric feeding.
iv. Avoiding accumulation of gastric fluids.
v. Decontamination of the gastric tract with nonabsorbable antibiotics.
vi. Avoiding gastrointestinal feeding.
vii. Disinfecting the medical team.
viii. Disinfecting equipment.
ix. Elevating the head by 30–40 degrees.
x. Avoiding heavy sedation.

Up to 50%–70% of patients infected with VAP succumb. Other than a high probability of mortality, there are several other complications affecting patients suffering from VAP like:

i. Prolonged stay on mechanical ventilation and ICU.
ii. On a few occasions, necrotizing pneumonia and pulmonary hemorrhage occur due to pseudomonas aeruginosa [6].
iii. Bronchiectasis and pulmonary scarring leading to recurrent pneumonias, catabolic state in malnourished patients.

Like SSI, VAP too inflicts high mortality rates and leads to higher treatment costs [7].

This section has discussed the different types of NIs and their management strategies. Section 12.3 covers various programs setup by hospitals and governments to control NI.

12.3 NI PROGRAMS

This particular section discusses the strategies currently in place to manage and mitigate NIs [19]. In bits and pieces, this section will also cover some of their history, which led to the methods currently used today.

Some of the most insightful information on how the NI program has been setup in Canada was covered by Paton et al. Based on U.S. data, it had been estimated that 5%–10% (180,540 to 361,082 patients) of patients admitted to a hospital contracted some form of NI. This problem was compounded by the fact that the US public health care establishment spent around US$4,532,000,000 per annum [19]. These are numbers that could stagger any health care establishment.

Out of this need, the Canadian public health care system setup the Bureau of Communicable Disease Epidemiology (BCDE) in 1980. Over its 37-years history, BCDE has initiated several programs to control infections. One of their key programs, initiated on June 1984, was the Canadian NI Surveillance Program (CNISP). Revisions and updates were continuously made to the CNISP. One of the significant ones was the recommendation from an ad hoc committee in 1993 to implement a national strategy to control NI and antimicrobial resistance [19].

12.3.1 The Canadian Nosocomial Surveillance Program (CNISP)

The BCDE setup a network of "sentinel hospitals" to become data collection centers in the CNISP. The first step of CNISP was to have sentinel hospitals acquire data to study the nationwide and regional NI trends, identify the magnitude of the problem and pathogens spreading NI. In 1995, there was widespread survey and study on methicillin-resistant *Staphylococcus aureus* and *Clostridium difficile* [19].

The next step of the CNISP was to survey the effectiveness of the currently employed techniques. This step is key to assure the effectiveness of the program. The third step was to implement the necessary health care facilities and observe their effect on the patient population [19].

We must take note of the in-built self-improvement mechanism of the CNISP. Such programs have demonstrated steadily decreasing rates of NI.

Several countries have formulated such NI programs. However, the success of these programs depends on adherence to the NI prevention protocols. The lack of adherence is discussed by Lizandra Ferrera et al. [32] in great depth. The study has been analyzed and summarized below.

12.3.2 The Failure of a NI Program in a Hospital

This section discusses and demonstrates how lack of compliance to a NI program can give a frighteningly high incidence of NI. Handwashing is one of the cheapest and easiest methods of controlling NI. This is because hands of a health care worker, or patient, are the primary ways of interacting with the environment. This makes it really easy for NI's to travel via touch [20].

In this Brazilian study, two observers were trained to collect data on the prevalence of NI and the adherence to hand sanitization from four different departments, that is, clinical, surgical, pediatric, and intensive care unit (ICU). The health care workers were surveyed twice a day. The standard of hand hygiene compliance was defined by washing hands after any contact with the patient or any inanimate object in the patient's room [20].

This compliance data was further classified into shift, sex, designation of health care worker, and the risk of cross-risk infection: highest risk being physically contacting the patient, or patient fluids, and lowest risk being deemed an indirect contact with a patient during hospital maintenance [20].

The handwashing initiative was marketed to several categories of health care workers. They were exposed to several presentations and illustrated posters to educate them about NI and handwashing. Even bottles of alcohol hand rub were distributed to the workers to encourage sanitizing hands [20].

After the hand-sanitizing program was introduced in this hospital, the average compliance increased from 21% to 24%. A 3% increase is not noteworthy. However, the compliance in nurses was up to 83.3%. Given that this initiative wasn't successful across all categories of health care workers, the decrease in NI was only 4.1% [20].

The lack of compliance to hand washing has not been discussed by this study. However, this resulted in high rates of NI and an increased spending on treating them. This also elongated the stay of patients in the hospital.

12.3.3 The Success of a NI Program in a Hospital

The Brazilian study mentioned above was inspired by the success of another Geneva-based study [21]. This study implemented a very similar hand hygiene routine to that of the Brazilian study.

Over a period of three years, handwashing compliance steadily improved from 48% to 66%. It was interesting to note that compliance was very high among nurses, but very poor among doctors. The reason for this disparity has not been explored in the Geneva-based study [21].

It was uplifting to note that the incidence of NI dropped from 16.9% to 9.9% in three years. The consumption of alcohol-based handrub solution increased from 3.5 to 15.4 L per 1000 patient-days in those three years [21].

The studies discussed in Sections 12.3.2 and 12.3.3 clearly show how effective a NI program can be if implemented correctly. Hopefully, in the future we will have sufficient studies to evaluate the success or failures of NI programs, and the motivations behind their success of failure.

After highlighting the importance of an effective NI program, we will now describe the different methods used to regulate the spread of NI.

12.3.4 NI Control Methods

A NI control program is categorized and subcategorized in the following manner:

12.3.4.1 Standard Precautions

i. Hand hygiene: It is the simplest and most effective way to control NI. The repercussions of its failure and success have been discussed in Sections 12.3.2 and 12.3.3 [1,22].
ii. Respiratory hygiene: Health care workers must wear masks as and when required, to prevent the workers from transmitting infections to the patient and vice versa [1,22].
iii. Personal protective equipment: Equipment like rubber gloves, hazardous material suits, etc., must be worn when dealing with contagious diseases [1,22].
iv. Injection safety: Only fresh syringes must be used and must be disposed of after use. Catheters must be disinfected before inserting. The prick site must be disinfected before being pricked [1,22].
v. Medication storage and handling: Medicines must be stored in areas with the least chances of contamination [1,22].
vi. Cleaning and disinfection: The hospital environment and equipment in general should be regularly cleaned and disinfected. This also extends to proper waste disposal [1,22].

12.3.4.2 Transmission-Based Precaution

i. Contact precautions: Patients must be avoided from being touched directly. Rubber gloves and adequate sanitization must be implemented before contact is made [1,22].
ii. Droplet precautions: In diseases like pneumonia, etc., this takes special priority. Masks and hazmat suits must be worn as the case demands [1,22].
iii. Airborne precautions: In diseases like swine flu, common cold, etc., the usage of masks is recommended. Adequate ventilation is also recommended [1,22].

12.3.4.3 Immunization and Vaccination Programs

These programs, when conducted at the national level, help the general population to have immunity against deadly pathogens. The effect of this immunity also protects them from NI when admitted to hospitals [1].

12.3.4.4 Education and Training of Health Care Staff

As seen in Section 12.3.3, when done right it can make a significant difference to NI statistics [1,22].

Even if the above control guidelines are adhered to, there is a lot which remains to be tackled in order to solve this challenge. Ever since the discovery of penicillin in 1928, humankind seemed to be winning the war against diseases. However, the tables were soon turned back as the microbes acquired resistance to these drugs. Section 12.3.5 examines antimicrobial use and resistance.

12.3.5 Antimicrobial Use and Resistance

Given the size and reproduction rates of microbes, they are ubiquitous [1]. They are found almost everywhere including the South Pole. While some microbes have helped forward our civilization, there are several others which are an existential bane to mankind.

Since the discovery of penicillin, antibiotics have been used extensively. They have perhaps been the biggest revolution to the pharmaceutical and health care industry. Diseases previously causing certain mortality (e.g., typhoid) are now easily manageable because of antibiotics.

Antimicrobials are a group of medicines which kill microbes. Disinfectants and antiseptics kill a wide range of microbes. Antibiotics kill or restrict growth of bacterial infections. Antifungals kill or restrict the growth of fungi. Similar descriptions could be assigned to antivirals and antiparasitics. While some antimicrobials are available to public, others can only be taken when prescribed by a licensed doctor upon diagnosis of the patient's symptoms.

Over time, with the abuse of antimicrobials, several strains of microbes have developed resistance to antimicrobials. The following fact reminds us of the gravity of this calamity. Antibiotic resistance is responsible for the death of a child every five minutes in Southeast Asia [23].

Self-medication with antibiotics, incorrect dosage, prolonged use, lack of standards for health care workers, and misuse in animal husbandry are the main factors responsible for increase in resistance. This resistance threatens the effective control against bacteria causing UTI, pneumonia, and bloodstream infections. Highly resistant bacteria such as MRSA, or multidrug-resistant Gram-negative bacteria, are the cause of high incidence rates of NIs worldwide [1,24].

Several policies have been issued by many countries and international bodies (like the WHO) to control antibiotic resistance. Most of these policies emphasize better civic hygiene, access to clean potable water, and vaccinations to prevent infections in the first place. They further elaborate on guidelines to instruct physicians on the administration of antibiotics. Pharmacists are regularly issued notices to not sell antibiotics to the general public without a doctor's prescription. The adherence to these guidelines varies from region to region.

Due to the more pertinent challenges of NI, certain advanced control strategies have been developed. These advanced strategies, however, do not undermine the earlier-mentioned strategies in any manner. The advanced infection control strategies

only serve to augment the effect of basic infection control strategies and methods. The following sections cover advanced NI control programs.

12.3.6 Antibiograms to Control Antibiotic Resistance

Patients who acquire NI's usually have suppressed immunities because of their current ailment. The growing threat of antibiotic/antimicrobial resistance compounds the problem further. To control this situation, antibiograms are used.

Simply defined, antibiograms is a summary of the effectiveness of antimicrobial drugs towards specific microbial strains. It evaluates the local and wide area effects of different drugs. This helps clinicians track antimicrobial resistance and prescribe specific drugs for specific microbes [25,26].

There are several automated tools to help clinicians and researchers generate and analyze antibiograms. One such software is WHONET [25]. It is a free downloadable software developed by WHO's Collaborating Centre for Surveillance of Antimicrobial Resistance.

Using such software, researchers generate antibiograms with a biyearly or yearly frequency. These are analyzed and the results provided to doctors to help them prescribe the correct antimicrobial. Antibiograms in conjunction with the regular NI control practices make for a formidable combination to control NI.

12.3.7 Cross-Infection Programs

Simply defined, cross infection is the spread of microbes from one person to another. This not only leads to the spread of infection, but also cross colonization and cross breeding to create new strains of microbes [22]. The addition of this effect with the already existing causes of NI compounds the problem of controlling NI. However, this can be controlled by implementing basic NI control strategies.

To develop the most effective NI program, health care establishments mix and match a variety of these methods and factors. While developing any of these programs, the regional peculiarities must be kept in mind. Regional peculiarities include the attitude of people's culture, economic conditions of the region, and population and technical variances like microbial susceptibility in different regions using antibiograms. Also, methods should be figured out to ensure compliance with NI control programs.

Although a lot has been achieved by the medical fraternity to handle NI during hospitalization, the advancement of computer software technology in the last two to three decades has significantly contributed in this field. In Section 12.4, we will discuss some of the ML studies on predicting, controlling, and mitigating NI.

12.4 ML AND NIs

Hospital information systems are available in most hospitals today. The digital data available from these systems has pioneered the way for automatic surveillance

systems. It has also added to the research for the automatic detection of NI and their trends. A lot of efforts around the globe have been based on this. Most of the systems, implement rule-based systems, but the evolution of ML in last decade has made it possible to use ML for NI. It can be used in mainly following areas regarding NI:

i. HAI prone score of the facility, which will eventually calculate magnitude of the problem and identify the HAI proactively.
ii. Identifying the trends in HAI across a range of hospitals or geographical territories.
iii. Proactive treatment for HAI.
iv. Understand the cost implication of HAI and its saving.
v. Evaluation of HAI prevention program.

12.4.1 Type of Data

As for any ML project, it is significant to have a substantial amount of data to develop any model. For an NI ML model preparation, a wide variety of data can be found, which can be primarily divided into structured and nonstructured categories.

12.4.1.1 Structured Data

12.4.1.1.1 Hospital Records

These days most of the hospitals have computer programs for patients' record keeping. These systems are called Hospital information systems. This data is mainly considered as structured data because it has some predefined format. For example,

i. A record for Patient Detail may have data looking like:
   1. Patient ID, Patient Name, Patient Address, Patient Phone No, Patient Date of Birth, Age, Sex, and Patient Emergency Contact Number.
ii. A record for Patients' past treatment and hospitalization may look like:
   1. Patient ID, Date of Checkup, Doctor ID, Time of Checkup, Patient Compliant, Prescription Type, and Prescription Details.
iii. A record of Patient's hospitalization may look like:
   1. Patient ID, Date of Admission, Doctor ID, Time of Admission, Disease ID, Patient Record, Cause of Admission, Cause of Discharge, admission diagnosis, previous surgery, previous intensive care unit stay, exposure to antibiotics, antacid, and drugs.

It should be noted that all of these records are structured because they have a defined format, as some fields can only hold numbers (Patient Phone No) and some fields can only hold characters (patient names). We also define their size or length; for example, a phone number can't be greater than 10 characters. The structured fields are easily managed with any database management system (DBMS), which have been the backbone of almost all hospital information systems.

12.4.2 Unstructured Data

This is the highly untapped category until a couple of years ago, but the new computer science techniques like ML and AI supported by humongous computational power and storage capacity has made it possible to understand and work upon unstructured data. For medical records, the unstructured data may look like:

i. Doctor prescription
ii. Doctor diagnosis
iii. Radiologist diagnosis
iv. X-ray reports
v. Patient discharge summary
vi. Interview with hospital staff

A very paramount technique for understanding medical prescriptions, notes, diagnoses, and discharge summaries is called natural language processing. It is the technique by which a computer program can understand and process the naturally indited notes. It is highly unstructured because there is no fine-tuned type or size of these notes. The NLP applications are different from conventional computer applications because computers traditionally require humans to interact with them in a precise programming language, which is unequivocal and highly structured. On the other hand, human verbalization is not always same. For example, we can pass on the same message in different ways while verbalizing or inditing. Hence, the interaction is equivocal and the linguistic structure can depend on many involute variables, including slang, regional dialects, and gregarious context. The purport of an NLP application is to understand all of this and engender sentiment, entity, context, and meaning out of it. All this should be automated and precise.

Advanced neural networks interpret physician notes, X-rays reports etc., fairly accurately. It is quite an advanced subject in medical research, and work is currently in progress.

12.4.3 Data Gathering Activity

It is also important to understand the data collation activity required to create a better, robust, accurate, and acceptable ML model. In case of NI we need to have a variety of data from all possible sources:

i. Hospitals
ii. Clinics
iii. Pharmacy
iv. Diagnostic labs

12.4.4 Data Sources

It should be noted that most of the research published on NI revolves around studies carried out in very small facilities, and hence it may inhibit the ability to make better

ML models. For example, most of the study are done in university hospitals and may not include all aspects in their studies. The more randomness in gathered data, the better the model would be. But there are many constraints to gather the data from various sources. For example, the study [27] was done on 688 patients. There are some studies focusing on both internal and external hospitals. For example, in a study conducted in Taiwan [30], the internal test set was for 461 hospitalizations whereas the external test set was for 2500 hospitalizations.

12.4.5 Variety of Data

It is of significant importance to understand that one model prepared by using only one type of data may not be suitable in all situations. For example, if the available information is primarily based on a study in an underdeveloped country then it is highly possible to have a higher tendency towards hygiene-related risks, causing CLABSIs. Similarly, on the flip side, if the data is collated only from a clinical facility which doesn't provide a tenured hospitalization facility then it will lack any SSI type of NI. There are some studies limited to only ICU records, while some covers entire hospitalizations. Which variety of data to use entirely depends on the problem at hand and the accessibility to data.

12.4.6 Data Collection Period

Another important angle for the data collation is that it should be collated over a period of time, and should not be considered as a onetime activity. This will remove any kind of bias being introduced in the data. It is quite critical that the data should be collected over a continuous period of time for better results. In most of the research conducted, studies were conducted over six days of hospitalizations for a period of 6 months.

12.4.7 Data Collection Locations

It is of paramount importance that the data should be collected from geographically spread hospital facilities. There are high chances that a particular region suffers through significant poverty, lack of education and modern facilities, which makes it more prone for NI. Any model prepared on that data only may have some bias towards showing NI in the patients. The opposite of this also holds true. It may be possible that an ultramodern hospital has high standards, and hence the data collected there may have minimum NI instances available in it. In order to generate a trained model it should have a sufficient number of NI infection cases.

12.4.8 Data for All Output Types

Another part of the data collection is that it should cover all the types of NIs. If the ML model is based on only one type of NI then it can't be used to identify other types of NI, as the model will not have sufficient data to train itself to identify those NI cases.

12.4.9 Data Exclusions

While we are gathering all of the data from all possible sources, it is also important to note that we may have to skip some data. For example, if all the data for hospitalizations is gathered then the patient record stays for less than 48 hours can be removed from the data of this study. It will result in a significant chunk of data being removed, but that is not useful for studying NI. Hospitalizations of less than 48 hours are not involved in the final dataset, as they contain too little information. That data may be useful for some other study, but it is not required for NI since 48 hours is too short for any NI infection.

As you will read later in this chapter in Section 12.4.13.4, the researchers drop some data, which are not useful to them. We need to understand that all data is not useful data.

12.4.10 Data From Interview

A lot of data about NI can be generated by interviewing nurses, patients, and their doctors.

12.4.11 The Imbalanced Data Problem

The main difficulty in this type of data (as in several medical diagnostic applications) is its highly skewed class distribution. As found in study [1], out of 683 patients only 11% (75 in number of the total) were infected and the rest were fine. The quandary of imbalanced datasets is very critical for the applications where the goal is to increase the apperception of the minority class (the infected patient class in the mentioned study) to its best. Here, we try to find other ways of biasing the modeling to support sensitivity (i.e., capacity to recognize positives), as predicated on a one-class Parzen density estimator and a one-class SVM.

12.4.12 Data Preparation Activity

As we have gathered data now, it's important to first prepare the data for the ML model. This comprises various activities depending on the type of data. For example, structured data will have a different set of preparation activities as compared to unstructured data. The data preparation activities for structured data are described below, whereas those for unstructured data are described in the Text Classification Section (12.4.13.4).

12.4.12.1 Handling Missing Values

As described in study [28], there were several variables having missing values during the data gathering activity. The missing values can be due to several reasons and are quite common in daily life. It may be due to erroneous or missing measurements. Help from medical experts would be required to identify the importance of these missing values. There are lots of ways to replace missing values with the most approximate values. In study [28], the missing values were replaced with the class-conditional mean for continuous variables, and the class-conditional mode for nominal ones.

12.4.12.2 Removing Redundant Variable

When the data is being collected, there is a normal tendency to collect as much data as possible for all the fields required. But all the fields (this can also be understood as variables, columns, attributes, and features) are not equally important from an ML perspective. Redundant variables should be removed. For example, if the two variables are highly correlated (in simple terms—have the same impact on output) then only one of them can be used, thus making the other redundant. As described in study [28], initially there were 83 variables at the start of the study, however, after careful examination they were able to remove 34 redundant variables and thus used only 49 variables for the study.

12.4.13 Implementation

In this section, we will mainly focus on four methods for discovering NI from patient data. Each of the methods is unique, and generates new results. The methods are:

i. One class classifier: SVM
ii. One class classifier: Parzen density estimator
iii. Text classification
iv. Linear regression and artificial neural network

12.4.13.1 One Class Classification Implementation

Normally, a classification method is two-class classification, but the one-class approach is particularly attractive in situations where cases from one class are difficult to obtain for model construction, which is the case of NI. The one-class classification is the ability to separate between new cases similar to members of the training set and all other cases that can occur. While it looks similar to conventional classification problems (i.e., two-class), one-class classification has a different way to train the classifier. The classifier of one-class classification is trained only by cases from the majority class, and it is never trained from the cases of the minority class. The purpose for the classifier is to develop the boundary separating those two classes based only on data that lies on one side of it (on which the classifier is trained). It must therefore estimate the boundary separating those two classes based only on data which lie on one side. The objective is to identify patients with one or more Nis, based on clinical and other data collected during the survey.

As we will see, the results received by one-class classifiers are better than the standard 2-class SVMs, with some trade-off between specificity and sensitivity.

12.4.13.2 One Class Classifier: SVM

The major arduousness intrinsic in the data (as in many medical diagnostic applications) is the highly skewed class distribution. As observed in study [1], out of 683 patients only 75 (11% of the total) were infected and 608 were not. The quandary of imbalanced datasets is concretely crucial in applications where the goal is to maximize apperception of the minority class (which is the infected patient class in our case). In this chapter, we investigate another way of biasing the modeling process

to boost sensitivity (i.e., capacity to apperceive positives) predicated on one-class Parzen density estimator and one-class SVM [1], which can be trained to differentiate between two classes on the substructure of the examples from a single class (in this case, only "normal" or noninfected patients). The infected ones are identified as "abnormal" cases or outliers, which deviate significantly from the mundane profile.

The one-class approach engendered a sensitivity of 92.6%, whereas a standard two-class SVMs scored a baseline sensitivity of 50.6%. Clearly, the results are in favor of one-class SVM.

To train one-class SVM classifiers researchers used an RBF kernel. Generalization error was estimated using fivefold cross-validation. In these type of cases (as seen in Parzen density estimator also), a tenfold cross validation is not used because the number of infected cases in every fold will be much less and will not help. Hence, the complete dataset was randomly partitioned into five subsets. The error rates estimated on the test sets were then averaged over the five iterations.

As described in research [27], a widely used performance metric in classification is accuracy. As the name suggest, accuracy means the fraction of correctly classified data points in the test set. But it is important to note that in highly unbalanced class, such metrics might be misleading. Let's take an example, on a dataset with a 95%–5% class distribution (assume 95% are normal patients not infected by NI, whereas 5% are infected by NI), it is straightforward to attain 95% accuracy by simply assigning each new case to the majority class. But if you think more about it, the impressive accuracy of 95% for such a solution is inacceptable as a medical diagnosis, as the classifier would have failed to recognize the infected cases, which are a minority of 5%.

To discuss alternative performance criteria we adopt the standard definitions used in binary classification. These performance criteria are industry standards to evaluate and validate various ML models.

TP and TN stand for the number of true positives and true negatives, respectively. Or the positive/negative cases recognized as such by the classifier.

FP and FN represent, respectively, the number of misclassified positive and negative cases.

Sensitivity: In two-class problems, the accuracy rate on the positives is defined as: TP/(TP + FN). It is important to understand that here FN represents those cases which are actually positive, but our model has falsely termed them as negative.

Specificity: The accuracy rate on the negative class, also known as specificity, is: TN/(TN + FP). It is important to understand that here FP represents those cases which are actually negative, but our developed ML model has falsely termed them as positive.

Classification accuracy is simply (TP + TN)/TP + TN + FP + FN.

Findings: Table 12.1 lists the results of summarized performance for the one-class SVMs. This shows the best results obtained through training classifiers by utilizing various parameter configurations on noninfected cases only. So it is clear the highest sensitivity is achieved when both ν and σ are minuscule. ν is termed as the upper limit on the fraction of outliers which can be ignored. The values of sensitivity and specificity have an intriguing relationship. Let's endeavor to understand it. The sensitivity of 92.6% designates that the model is very good at identifying the authentic

TABLE 12.1
Performance of One-Class SVMs for Different Parameter Settings Utilizing an RBF Gaussian Kernel

Parameters		Accuracy	Sensitivity	Specificity
ν	σ	%	%	%
0.05	**10^{-4}**	**74.56**	**92.60**	**43.73**
	0.1	75.49	80.60	65.60
	0.15	72.51	70.39	74.40
	0.17	71.69	66.94	77.87
0.2	10^{-4}	75.69	79.28	68.27
	0.06	74.97	77.14	69.87
	0.07	74.97	76.82	70.40
	0.1	74.36	74.67	72.27

positive cases as positives; however, at the same time the specificity of 43.73% denotes that the model is also calling negative cases withal as positive cases. It clearly shows that the model is partial towards relegating most of the cases as positives.

But as you move across Table 12.1, there is a tradeoff, as the value of sensitivity decreases and the value of specificity increases. It signifies that model is now relegating more cases into a negative category, which it was not doing earlier.

12.4.13.3 One Class Classifier: Parzen Density Estimator

As performed in the study at University Hospital of Geneva (HUG), a one-class Parzen density estimator [28] was developed. As suggested above, the one-class classifier is always trained on only one class and in this study the classifier was trained on only noninfected patient data, however, the test set contained data for both infected and noninfected patients. A fivefold cross validation method was utilized for model validation where the consummate data set was divided into five subsets. On every iteration, one subset (containing one fifth of the samples) was kept aside as a test data set and the remaining four were put into a training set. The detailed explication of the Parzen density estimator is outside the scope of this book.

The one-class Parzen density estimator showed the best results were achieved by training the classifiers through utilization of different parameter configurations on noninfected cases only. The one-class approach, like this one, leads to consequential amendments in sensitivity over classical symmetrical SVMs. But a major constraint of one-class Parzen density estimator lies in the requisite of high computational systems during the testing phase.

Experimental results reported in the paper [28] are emboldening. From the perspective of sensitivity, one-class Parzen density estimators procure the highest caliber (88.6%) observed by the authors throughout a series of studies on the quandary. However, the price paid in terms of loss in specificity is quite exorbitant, and domain experts should decide about the high apperception rate.

12.4.13.4 Text Classification Method

As mentioned in [29], this study focuses on the application of text classification using SVM and gradient tree boosting. This study is quite different from the above two because it is based on unstructured data. This method uses some of the text mining processes required only for text classification and not for structured data (Figure 12.1).

Let's try to understand what the researchers have done in [29] and understand the flowchart above.

DR stands for daily patient record whereas HR is hospitalization record. All DRs of a patient together amount to the patient's HR.

This research is based on 120 inpatients at one hospital in Sweden. Not all information stored in the patients' EHRs was considered valuable by the physicians for detecting HAI.

Thus, a subset of information was retrieved:

- Journalanteckning (Engl.: record notes),
- Läkemedelsmodul (Engl.: drug module),
- MikrobiologiskaSvar (Engl.: microbiological result) and
- Kroppstemperatur (Engl.: body temperature).

FIGURE 12.1 Text-classification approach for automatically detecting HAI.

The observation extracted from these modules contains few structured and few unstructured data. As we have already expounded on earlier, the structured data refers to the stored data in predefined fields, such as International Relegation of Diseases–10th Revision (ICD-10) diagnosis codes, medication, or body temperature. But the most intriguing part of this research [29] is the unstructured data, which refers to textual notes indited by medicos, such as daily notes or microbiological results. Since some of the 120 patients were hospitalized multiple times during the five-month period of records we received, our dataset comprises 213 HRs.

The analysis done by researchers [29] revealed that only 128 of 213 HRs contained HAI diagnoses (positive examples). Those records represent the HAI class. According to the assessment, the remaining 85 HRs contained no HAI diagnoses (negative examples), thus representing the NoHAI class. The dataset was not balanced, but instead, it was skewed toward the infected class, which is incipient to this relegation. In our earlier relegation, the data was skewed towards the NonNI class. We only utilized the class containing HAI for presage and not the class without HAI.

Since the data is unstructured (medico notes), we have to introduce some incipient terms here:

Term frequency (TF). In this method (TF 1000), the 1000 most frequent terms, were culled predicated on their TF. TF refers to the simplest weighting scheme, where the weight of a term is equivalent to the number of times the term occurs in a document [22,23].

Lemmatization and stemming. Lemmatization describes the process of reducing a word to a mundane base form, customarily its dictionary form (lemma). For instance, hospitals, hospital's → hospital.

Lemmatizers

[in]: having

[out]: have

In a simplistic way, stemming can be considered as a process to remove the ending part of some words like "ing."

Stemmers

[in]: having

[out]: hav

Stop word removal. Stop words are terms regarded as not conveying any significant semantics to the texts or

Phrases. For example, the words like "The, a, an, etc." can be removed during stop word removal.

Infection-specific terms. As done in research [29], medical experts help is normally sought to create a bag of words associated with the presence of NI in patients records. For example, catheter, ultrasound, surgery, fever have high chances to be present in patient's record in a case of NI. An automatic synonym extractor can be used, but it requires supervision from a domain expert. In this case, a final list of 374 terms (words) was developed.

Term frequency–inverse document frequency (TF-IDF). TF-IDF, is a method to understand the importance of a word in a document against a corpus.

The TF-IDF value increases proportionally with the number of times a word appears in a document, but is often offset by the frequency of the word in the corpus. This is extremely important to understand that some words appear very frequently in written or spoken communication.

12.4.13.5 Linear Regression and Artificial Neural Network

As referred to in article [29], In a study conducted in Taiwan, [31] linear regression (LR) and ANNs were used to predict HAI. The system used structured data, which comprises 16 features extracted from patient records, ranging from demographic, procedural, and therapeutic features to features concerning the general health status of the patient.

ANNs are trained using back-propagation and conjugate gradient descent. In order to evaluate the system, internal (same hospital) and external (another hospital) data sets were used. For the internal test set of 461 hospitalizations, the best result was produced using the ANN approach, reaching a recall of 96.64 percent and a specificity of 85.96 percent. For the external test set consisting of 2500 hospitalizations from different hospitals, LR gave the best result with a 82.76 percent recall and 80.90 percent specificity.

12.5 CONCLUSION

It must be understood that the different set of classifiers give different results and there is a lot of trade-off involved. For example, in the study conducted at the University Hospital of Geneva, a one-class Parzen density estimator [28] produced the highest level (88.6%) sensitivity observed by the authors among a series of studies on the problem. However, they lost specificity, thus domain experts must decide if the high recognition rate is worth the cost of treating those patients who are not infected by NI. It should also be understood that high false positives means the hospital will be considered as a breeding zone of NI in various reports, which may not be beneficial to the hospital especially when it is reported, since it is false.

A survey conducted by WHO in 2002 found that for 55 hospitals in 14 countries, an average of 8.5 percent of all hospital patients suffer from HAI.

To sum up, this chapter focuses on deploying four ML algorithms to hospitalization records:

- One-class classifier: SVM
- One-class classifier: Parzen density estimator
- Text classification
- LR and ANNs

There is always a tradeoff between sensitivity and specificity of the ML models, which requires very careful manual examination by medical experts to select the best model. But as described in the chapter, there are various means to increase the ML model accuracy, for example:

- Efficient and elaborative data gathering
- Data processing/preparation/feature selection and reduction

- Fine Tuning ML model parameters
- Validation of models by fivefold or tenfold
- Comparison of models and selecting the best one

Though in most of the studies, the data set of patient records was minute in size (thousands only), it nevertheless demonstrated the potential of applying ML techniques to patient records, including the structured and unstructured components to identify NI. The major hurdle, typical in medical diagnosis, is the quandary of rare positives.

Algorithms with high recall are especially felicitous for the screening of infections. The overall goal will always be obtaining high recall (approaching 100%) with the highest precision possible for HRs. This will enable us to implement a system able to can screen all hospitalization records and filter out all HSRs containing HAI. This would reduce hospital staff workload tremendously, as they would only need to analyze those HSRs preselected by the system.

REFERENCES

1. Khan HA, Baig FK, Mehboob R. Nosocomial infections: Epidemiology, prevention, control and surveillance. *Asian Pac J Trop Biomed*. 2017;7(5):478–482, ISSN 2221–1691, https://doi.org/10.1016/j.apjtb.2017.01.019.
2. Rabindran, Gedam DS. Central line associated bloodstream infections. *Int J Med Res Rev* 2016;4(8):1290–1291. doi:10.17511/ijmrr.2016.i08.37.
3. Nelson AD. Central Line Associated Bloodstream Infections, *Fellows Conference*, University of Utah School of Medicine. 06.27.2012.
4. CDC. Urinary tract infection (catheter-associated urinary tract infection [CAUTI] and noncatheter associated urinary tract infection [UTI] and other urinary system infection [USI]) events. Atlanta, Georgia: CDC; 2016
5. Anderson DJ. Surgical site infections. *Infect Dis Clin North Am*. 2011;25(1):135–153, ISSN 0891–5520, https://doi.org/10.1016/j.idc.2010.11.004. (http://www.sciencedirect.com/science/article/pii/S0891552010000917) Keywords: Surgical site infection; Health care–acquired infection; Outcome.
6. Bansode BR. Ventilator—Associated Pneumonia (VAP), Medicine Update 2011, The Associations of Physicians of India.
7. Morillo-García Á, Aldana-Espinal JM, de Labry-Lima AO et al. Hospital costs associated with nosocomial infections in a pediatric intensive care unit. *Gac Sanit*. 2015;29(4):282–287, ISSN 0213-9111, https://doi.org/10.1016/j.gaceta.2015.02.008. (http://www.sciencedirect.com/science/article/pii/S021391111500031X) Keywords: Intensive Care Units, Pediatric; Nosocomial infection; Health care costs; Hospital costs; Cohort studies; Regression analysis; Unidad de CuidadosIntensivosPediátricos; Infección nosocomial; Costes; Costeshospitalarios; Cohortes; Regresión.
8. Stevens V, Geiger K, Concannon C et al. Inpatient costs, mortality and 30-day re-admission in patients with central-line-associated bloodstream infections. *ClinMicrobiol Infect*. 2014;20:O318–O324. DOI: http://dx.doi.org/10.1111/1469-0691.12407.
9. Zarb P, Coignard B, Griskeviciene J et al. The European Centre for Disease Prevention and Control (ECDC) pilot point prevalence survey of health care-associated infections and antimicrobial use. Collective, *Euro Surveill*. 2012;17:20316. https://doi.org/10.2807/ese.17.46.20316-en.

10. Taslakian B, Chokr J. Placement of nontunneled central venous catheter. In: Taslakian B, Al-Kutoubi A, Hoballah JJ, editors. *Procedural dictations in image-guided intervention: nonvascular, vascular and neuro interventions*. New York: Springer; 2016. p. 379–81.
11. Rajaram SS, Desai NK, Kalra A et al. Pulmonary artery catheters for adult patients in intensive care. *Cochrane Database Syst Rev*. 2013;2:CD003408.
12. Johansson E, Hammarskjold F, Lundberg D, Arnlind MH. Advantages and disadvantages of peripherally inserted central venous catheters (PICC) compared to other central venous lines: A systematic review of the literature. *Acta Oncol*. 2013;52:886–92.
13. Chopra V, Flanders SA, Saint S. The problem with peripherally inserted central catheters. *JAMA*. 2012;308(15):1527–1528. doi:10.1001/jama.2012.12704.
14. Tunneled Central Venous Catheter (CVC) Placement, Wexner Medical Center, The Ohio State University, patienteducation.osumc.edu.
15. Di Carlo I, Cordio S, La Greca G et al. Totally implantable venous access devices implanted surgically a retrospective study on early and late complications. *Arch Surg*. 2001;136(9):1050–1053. doi:10.1001/archsurg.136.9.1050.
16. Frasca D, Dahyot-Fizelier C, Mimoz O. Prevention of central venous catheter-related infection in the intensive care unit. *Crit Care* 2010;14(2):212.
17. Nicolle LE. Catheter associated urinary tract infections. *Antimicrobial Resistance and Infection Control*. 2014;3:23. doi:10.1186/2047-2994-3-23.
18. Bagnall NM, Vig S, Trivedi P. Surgical-site infection. *Surgery* (Oxford). 2009;27(10):426–430, ISSN 0263-9319, https://doi.org/10.1016/j.mpsur.2009.08.007. (http://www.sciencedirect.com/science/article/pii/S0263931909001744) Keywords: cost; NICE guidelines; prevention; surgical-site infections.
19. Paton S. Nosocomial infection program. *Can J Infect Dis*. 1995;6(2):73–75.
20. de Almeida e Borges LF, Rocha LA, Nunes MJ, Filho PPG. Low compliance to handwashing program and high nosocomial infection in a Brazilian hospital. *Interdiscip Perspect Infect Dis*. 2012;2012:5, Article ID 579681. doi:10.1155/2012/579681.
21. Pittet D, Hugonnet S, Harbarth S et al. Effectiveness of a hospital-wide programme to improve compliance with hand hygiene. *The Lancet*. 2000;356(9238):1307–1312.
22. Hota B. Contamination, disinfection, and cross-colonization: Are hospital surfaces reservoirs for nosocomial infection? *Clin Infect Dis*. 2004;39(8):1182–1189. https://doi.org/10.1086/424667.
23. Singh PK. *Antibiotics, handle with care*. Geneva: WHO; 2016. [Online] Available from: http://www.searo.who.int/mediacentre/releases/2015/antibiotics-awareness-week-2015/en/.
24. WHO. *Antimicrobial resistance*. Geneva: WHO; 2014. [Online] Available from: http://www.searo.who.int/thailand/factsheets/fs0023/en/.
25. Joshi S. Hospital antibiogram: A necessity. *Indian J Med Microbiol*. 2010;28:277–80.
26. Saranraj P, Stella D. Antibiogram of nosocomial infections and its antimicrobial drug resistance. *Int J Pharm Biol Arch*. 2011;2(6):1598–1610.
27. Cohen G, Hilario M, Sax H, Hugonnet S, Pellegrini C, Geissbuhler A. An application of one-class support vector machine to nosocomial infection detection. *Studies in Health Technology and Informatics*. 2004;107:716–20. doi: 10.3233/978-1-60750-949-3-716.
28. Cohen G, Sax H, Geissbuhler A. Novelty detection using one-class Parzen density estimator. An application to surveillance of nosocomial infections. *Studies in Health Technology and Informatics*. 2008;136:21-6. doi: 10.3233/978-1-58603-864-9-21.
29. Ehrentraut C, Ekholm M, Tanushi H, Tiedemann J, Dalianis H. Detecting hospital-acquired infections: A document classification approach using support vector machines and gradient tree boosting. *Health Informatics Journal*. 2018;24(1):24–42. http://doi.org/10.1177/1460458216656471.

30. Chang YJ, Yeh ML, Li YC et al. Predicting hospital-acquired infections by scoring system with simpleparameters. *PLoS ONE*. 2011;6(8):e23137.
31. Dalal MK, Zaveri MA. Automatic text classification: a technical review. *Int J Comput Appl*. 2011;28(2):37–40.
32. de Almeida e Borges LF, Rocha LA, Nunes MJ, Filho PPG. Low compliance to handwashing program and high nosocomial infection in a Brazilian Hospital. *Interdisciplinary Perspectives on Infectious Diseases*. 2012; Article ID 579681, 5 pages. doi:10.1155/2012/579681.

13 No Human Doctor
Learning of the Machine

Leonid Datta, Emilee Datta, and Shampa Sen

CONTENTS

13.1 Introduction 217
13.2 ML: The Approach Towards Automation 218
13.3 Different Categories of Algorithms 218
13.3.1 Regression 218
13.3.2 KNN Classification 219
13.3.3 Decision Tree (DT) 220
13.4 Deep Learning: Association with the No-Human-Doctor Concept 221
13.5 Methodologies 222
13.6 Case Study 223
13.6.1 Da Vinci Robotic Machine 223
13.6.2 Other Case Studies 224
13.7 Future Research 225
13.8 Conclusion 226
Disclaimer 226
References 226

13.1 INTRODUCTION

The human mind has a unique quality for learning. This learning starts from the very first day and continues throughout life. The learnt knowledge is used for future work, which is commonly termed experience. While comparing a machine with the human mind, scientists found an important gap between them, as the machine had no learning capability or no capacity to analyze previously performed tasks (Solomonoff 1957; Case and Smith 1983). The machines were specifically coded for a task, but threw exceptions for other tasks assigned to them. This explicit gap was analyzed and learning was considered as an important and new task to be performed by the machine. Thus, the learning factor of a machine was introduced, which was a big step towards automation and ML. Initially the learning was knowledge-based, which later turned into a data-driven knowledge acquired system. The system continuously collects data either through sensors (for embedded systems) or through the input data (for software-based systems). The data set is analyzed and through different steps, knowledge is acquired (Yusuff et al. 2012). This knowledge is used as a learning factor.

The first step in approaching the concept of a "no human doctor" is learning, and "learning" initiates the availability of the data sets. From a specific data set, the behavior prediction becomes the unique capacity of the proposed system. For instance, classification is used as the basic system for predicting the stage of cancer diagnosis (André et al. 2009), while in case of observations in hospitals, the data are recorded at every moment and the patients are continuously supervised. If any kind of outliers are found in the data, or the values are found to be beyond a specific limit, the system alerts the doctor. Using the ML algorithms available at a basic level, the first step towards the automation can accomplished (Aha 1997).

13.2 ML: THE APPROACH TOWARDS AUTOMATION

ML deals with the algorithms able to automate a task. Towards the approach of no-human-doctor, the available set of data is used to make the prediction and different sensors are used to sense or input the required data. Learning algorithms can be classified into two types—supervised and unsupervised learning. In the case of supervised learning, a separate data set is available, which is taken from the previous records to train the new system. The data may also have uncleaned data or outliers. The approach of data analysis differs depending on the size of data (Jacobs et al. 1991; Jordan and Jacobs 1994), whereas unsupervised learning deals with problems where no training data set is available. Supervised learning mainly deals with problems regarding prediction of values, but unsupervised learning deals with classification problems.

Normally, before the data analysis a data set is cleaned using algorithms. This cleaning includes steps like removing null values, removing outliers, etc. After the data set is cleaned, it is used as a training data set or knowledge base for algorithms (Abe 1997; Angluin 1988).

13.3 DIFFERENT CATEGORIES OF ALGORITHMS

13.3.1 Regression

As mentioned earlier, classification techniques can be used to predict whether a tumor is malignant or benign. In other words, the probability of the tumor belonging to either type is calculated (Specht 1991).

But in the case of regression, we take prediction as the value found through the equation of a graph. The values available are the training data set. If we consider the case study of cancer prediction using different variables, regression can be the way. Yusuff et al. (2012) used a logistic regression technique in their experiments.

The question may arise as to how accurate the prediction is or the specific values taken may also vary in parameters. To measure this, a term, cost-function, has been defined to measure the extent to which the prediction is not accurate. If the cost-function value is large, then the prediction is less accurate.

If we consider a hypothesis $y = h(x)$ where x is the independent variable and y is the value to be predicted, then for a set of values $\times$ [represented as matrix X containing n values] and a given set of $(n-1)$ values of y, and the predicted value [represented as matrix Y containing n values], then the cost-function is defined as $J = (1/2n)\ \mathrm{sum}(h(x) - y)^2$.

FIGURE 13.1 Variation of cost-function with learning factor.

FIGURE 13.2 Cost-function with n-iterations.

Since it is a quadratic equation, the graph plotted must be parabolic in shape and open upwards, and thus one global minima must exist. So, the parameters must be chosen in such a way that we reach the point of global minima with minimum no of iterations.

If α is considered as the learning factor for the algorithm and J as the cost function, then considering a linear relation we predict a linear equation $\theta(j) := \theta(j) - (*(d/d\,\theta)(J))$. Since for continuous iterations the value of $d/d\theta$ (j) becomes gradually less, the learning factor α is kept constant. Figure 13.1 shows two cases where the learning factors are chosen either too big or too small. At a certain number of iterations the global minima is met. At that point the $\tan(\theta)$ value of the tangent becomes zero and continuously two iterations produce the same value. At this point the global minima is reached. Figure 13.2 describes this condition.

13.3.2 KNN Classification

Among the different basic algorithms available, K-nearest neighbor (KNN) is the most important one because of its simplicity. When KNN algorithm mainly deals

FIGURE 13.3 Flowchart of KNN algorithm.

with graphical representation and find to which a specific data plot belongs when more than one set of data is available (Zhang and Zhou 2007). More straightforward to define, it deals with classification problems by finding its nearest neighbor (Cover and Hart 1967).

KNN uses a user-defined variable k, which should be fixed before the algorithm is applied. "k" is chosen as an odd number, which defines the number of final nearest neighbors to be found. After the k has been defined, the graphical distance between the points of the knowledge data set and the new defined point are calculated. These distances are observed and the k number of points having shortest distance value are considered. Since the k is odd, the set containing k nearest neighbors will have one majority group. The new element will be considered as the member of the majority group. This is initially done for classification between two sets but it can be extended to n classification problems. Figure 13.3 shows a flowchart containing the steps of a KNN algorithm.

Towards the approach of no-human-doctor, this KNN can be used as a very basic technique to examine the patient behavior recorded through the sensors and also will help in immediate prediction. The training data set can be dynamically allocated while implementing, and the data set can be updated continuously along with the process. This approach will boost the diagnosis problem. While the initial implementation was with two variables only, the concept can be extended to a *n*-variable concept. Figure 13.4 describes an example with two data set and two cases with k = 5 and k = 9.

13.3.3 Decision Tree (DT)

Based on the data collected, we need a decision-making tool to predict the final result. For predicting a result, we observe the behavior of the patient termed as symptoms. Based

FIGURE 13.4 Example of KNN with two k values.

on the symptoms, the nature of the disease can be predicted (Tompson et al. 2014). For predicting the decision, a tree-like structure is formed, which helps to make it easy for the system—a DT (Iba and Langley 1992). It contains several nodes and these are divided into three types. One is the root node—from this node we start our iteration. Another is the leaf node where the iteration finishes. The remaining ones are called just nodes.

The DT is a valuable tool for object recognition. This object recognition can be applied for detecting an affected place (e.g., ulcer) and also for automatic analysis of X-ray or other scan reports (Haussler 1992; Freund and Mason 1999).

In the case of X-ray analysis, object recognition helps in finding if any object is stuck inside the body or even in the case any cracks are found. The image is taken and the gray scale values are plotted in a matrix for detection. The image is transferred through different filters and then the DT algorithm is applied (Clare and King 2001; Comite et al. 2010).

13.4 DEEP LEARNING: ASSOCIATION WITH THE NO-HUMAN-DOCTOR CONCEPT

Deep learning is a broader arm of ML technology, which emphasizes more detailed learning using different categories of algorithms. The main algorithms currently being researched are image restoration, speech recognition, visual processing, and more importantly natural language processing (Collobert et al. 2011). Natural language processing broadly contains emotion/sentiment analysis, information retrieval, spoken language understanding, and context analysis. This can be applied in the no-human-doctor approach for understanding the critical level of a patient's condition, to process and detect the initial symptoms of the patients. Collected data from this section can be used as the training data set for speech recognition, and initial symptoms can be verified through an image recognition system. As in the case of cancer patients, the initial scan reports can be verified through software to

FIGURE 13.5 Object identification in X-ray images.

judge the condition. If we take the example of scanning the X-ray reports, then visual image processing can be a path to process the reports and check for cracks in bones or find objects stuck inside the patient. Figure 13.5 shows the application of object recognition in a scanned report of teeth.

For treatment of psychiatric patients, AI has shown great success in detecting the types of depression. Using learning algorithms on the observation data captured while the brain is at rest, researchers have categorized various subtypes of depression as mixtures of anxiety and lack of pleasure. On the other hand, Alzheimer's prevention should be started at a very early stage; the disease can be detected through data analysis of multicenter DTI Data (Dyrba et al. 2013). On the other hand, issue-regulated splicing code can be a good example of the use of deep learning in automation (Leung et al. 2014).

For long-term treatment, as in cases of schizophrenia, AI can be used to detect the initial symptoms and to predict the likelihood of a previously unseen patient having schizophrenia. Even the severity of symptoms, such as inattentiveness and bizarre behavior, could be predicted from brain scans through image recognition and object detection (Krizhevsky, Sutskever, and Hinton 2012).

13.5 METHODOLOGIES

For implementation of a simple regression technique, one of the simplest tools is GNU octave. GNU octave deals with data sets as matrix elements and using matrix multiplication commands.

On the other hand, GNU octave provides an interface to plot a 3D graph with a broader view, making it convenient to visualize the plotted data for finding the required prediction.

As seen in Figure 13.6, the typed command is

```
[x,y] = meshgrid(-2:.2:2);
g = x.* exp(-x - y);
surf(x, y, g)
print -deps graph.eps
```

For the equation of a random prediction of independent variable g, with dependent variables x and y as $g = x*(e^{-x-y})$.

FIGURE 13.6 Graph of g = x*(exp(−x−y)) plotted through GNU octave.

Similarly, in case of a KNN algorithm, different categories of distances are considered depending on the problem. To name a few of the distance measurement methods, we include Euclidean, Mahalanobis, cosine, correlation, spearman, etc. Among these, Euclidian distance measures the graphical distance between the point and the data set, while Mahalanobis distance measures the number of standard deviations the point covers from the mean.

13.6 CASE STUDY

13.6.1 Da Vinci Robotic Machine

The da Vinci robotic machine, more popularly known as the da Vinci Surgical System, helps doctors to perform surgeries which are critical to perform with hands, especially those dealing with nerves (Gettman et al. 2002).

A machine is needed in order to handle surgeries dealing with sensitive organs and body parts. The da Vinci system typically has four hands to perform the job as a peripheral device. The hands are controlled by a doctor sitting at the back of the machine, where he/she has a 3D view of the surgery being performed. It is superior to working with a microscopic glass because the latter typically reduces the 3D nature of the picture and the sense of depth is less accurate. Along with a 3D view, it has a speaker and a microphone to communicate with the other team members who are handling the system. Beneath this, the machine contains two small input devices which are free to move 360°, and the sensors are attached to the thumb and index fingers of both hands. It typically works as if the surgery was being performed by those hands only. The doctor's legs control the movement of the camera. The system contains a large output screen where the preset functions are displayed continuously. Figures 13.7 and 13.8 illustrate the arrangement of the surgery system. This machine helps to perform the surgery with a minimum of discomfort to the patient and a minimal chance of error by the doctor.

From the case study of the da Vinci surgical system, it is clear a surgery can be performed by a peripheral device through a control unit. The next generation system

Front view

Doctor controlling movement

FIGURE 13.7 Front view of the da Vinci Robotic Machine.

can thus be designed to perform surgeries without the direct control of a human. The camera output can be directly sent for analysis by image object detection and the pathway must be preset to the data. The data collected through the graphical analysis of the image can be used to set the next path. The movement of the handle can be decided through the DT system.

On the other hand, for balloon surgery the movement of the balloon can be fixed using object detection on the image collected through the camera of the balloon head.

13.6.2 Other Case Studies

The smart tissue autonomous robot can be considered an important milestone for automated surgery systems. It uses a 3D-sensed imaging method to guide the robot for performing surgeries. The robotic machine performed open surgery, laparoscopic and robot-assisted surgery demonstrating a performance measure deemed good enough.

Orthopilot (Kowalski and Górecki 2004) is another example, although designed in the 1990s. It had many disadvantages including that of cost, but it showed the

FIGURE 13.8 Arrangement of the da Vinci Robotic Machine.

path towards the automation of surgeries. This was especially the case for total knee arthoplasty, unicondylar knee arthoplasty, etc. where it performed well. Orthopilot was implemented with a good navigation system, which opened the way for later robotic designs (Figure 13.9).

The data collected from different sources concerning performance of surgeries using robotic machines were analyzed using ML algorithms for performance evaluation. The LikeSmart Tissue Autonomous Robot was estimated to perform with 70%–80% accuracy range. This analysis helps in improving surgical skills of the machines.

13.7 FUTURE RESEARCH

In the proposed system, future work may include more regression techniques where nonlinear regression follows more complex approaches, including n-dimensional cases.

FIGURE 13.9 Front view of Orthopilot.

Even the support vector machine can be implemented in that case. Different categories of clustering algorithms, including probabilistic clustering, can be a convenient way for grouping tasks. Rather than directly predicting the group, fuzzy logic is one way to classify the behavior of a disease in probabilistic form. These probabilistic models, if implemented, can be used as new data sets for training and predicting trends or unknown patterns in a model. ML thus has a great future in automation of medical treatment by enhancing existing algorithms to a greater accuracy level.

13.8 CONCLUSION

Robotic machines including the da Vinci robotic surgeon have shown the path for automation of medical treatment. With the progress of ML tools and algorithms, the analysis of data has become more convenient; finding the unknown and hidden patterns in a data set has become relatively easy (Wang et al. 2010). Complex diseases like Alzheimer's or schizophrenia are being predicted through data analysis, which was initially difficult to predict with symptoms only. On the other hand, the progress of language processing and object recognition is boosting the automation field for symptom analysis, X-ray report checking, etc. The system can thus be implemented using more advanced algorithms of deep-learning technology towards the no-human-doctor concept, which will benefit society and science alike.

DISCLAIMER

The image of teeth for object recognition was taken from https://emedicine.medscape.com; accessed on Dec 31, 2017. This is an open source article and the authors and chief editor have nothing to disclose. Terms and Policy of the site had no objection statement relating to the reuse of the images.

REFERENCES

Abe, N. 1997. "Towards Realistic Theories of Learning." *New Generation Computing* 15 (1): 3–25. doi:10.1007/BF03037558.

Aha, D. W. 1997. "Lazy Learning." *Artificial Intelligence Review* 11: 7–10. doi:10.1007/978-94-017-2053-3.

André, F., S. Michiels, P. Dessen, V. Scott, V. Suciu, C. Uzan, V. Lazar et al. 2009. "Exonic Expression Profiling of Breast Cancer and Benign Lesions: A Retrospective Analysis." *The Lancet Oncology* 10 (4). Elsevier Ltd: 381–90. doi:10.1016/S1470-2045(09)70024-5.

Angluin, D. 1988. "Queries and Concept Learning." *Machine Learning* 2 (4): 319–42. doi:10.1023/A:1022821128753.

Case, J., and C. Smith. 1983. "Comparison of Identification Criteria for Machine Inductive Inference." *Theoretical Computer Science* 25 (2): 193–220. doi:10.1016/0304-3975(83)90061-0.

Clare, A., and R. D. King. 2001. "Knowledge Discovery in Multi-Label Phenotype Data," *In: Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery,* pp. 42–53. doi:10.1007/3-540-44794-6_4.

Collobert, R., J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. "Natural Language Processing (Almost) from Scratch." *Journal of Machine Learning Research* 12: 2493–537. doi:10.1.1.231.4614.

Comite, F. D., R. Gilleron, M. Tommasi, F. D. Comite, R. Gilleron, M. Tommasi, and Learning Multi-label Alternating, and Decision Trees. 2010. "Learning Multi-Label Alternating Decision Trees from Texts and Data To Cite This Version: HAL Id: Inria-00536733."

Cover, T., and P. Hart. 1967. "Nearest Neighbor Pattern Classification." *IEEE Transactions on Information Theory* 13 (1): 21–27. doi:10.1109/TIT.1967.1053964.

Dyrba, M., M. Ewers, M. Wegrzyn, I. Kilimann, C. Plant, A. Oswald, T. Meindl et al. 2013. "Robust Automated Detection of Microstructural White Matter Degeneration in Alzheimer's Disease Using Machine Learning Classification of Multicenter DTI Data." 8 (5). doi:10.1371/journal.pone.0064925.

Freund, Y., and L. Mason. 1999. "The Alternating Decision Tree Learning Algorithm." *Proceeding of the Sixteenth International Conference on Machine Learning*, 10 str. doi:10.1007/s13398-014-0173-7.2.

Gettman, M. T., R. Neururer, G. Bartsch, and R. Peschel. 2002. "Anderson-Hynes Dismembered Pyeloplasty Performed Using the Da Vinci Robotic System." *Urology* 60 (3): 509–13. doi:10.1016/S0090-4295(02)01761-2.

Haussler, D. 1992. "Decision Theoretic Generalizations of the PAC Model for Neural Net and Other Learning Applications." *Information and Computation* 100 (1): 78–150. doi:10.1016/0890-5401(92)90010-D.

Iba, W., and P. Langley. 1992. "Induction of One-Level Decision Trees (Decision Stump)." *ML92: Proceedings of the Ninth International Conference on Machine Learning*, Aberdeen, Scotland, July 1–3, 1992.

Jacobs, R. A., M. I. Jordan, S. J. Nowlan, and G. E. Hinton. 1991. "Adaptive Mixtures of Local Experts." *Neural Computation* 3: 79–87.

Jordan, M. I., and R. A. Jacobs. 1994. "Hierarchical Mixtures of Experts and the EM Algorithm." *Neural Computation* 6 (2): 181–214. doi:10.1162/neco.1994.6.2.181.

Kowalski, M., and A. Górecki. 2004. "Total Knee Arthroplasty Using the OrthoPilot Computer-Assisted Surgical Navigation System." *Ortop Traumatol Rehabil* 6 (4): 456–60.

Krizhevsky, A., I. Sutskever, and G. E. Hinton. 2012. "ImageNet Classification with Deep Convolutional Neural Networks." *Advances In Neural Information Processing Systems*, 1–9. http://dx.doi.org/10.1016/j.protcy.2014.09.007.

Leung, M. K. K., H. Y. Xiong, L. J. Lee, and B. J. Frey. 2014. "Deep Learning of the Tissue-Regulated Splicing Code." *Bioinformatics* 30 (12): 121–29. doi:10.1093/bioinformatics/btu277.

Solomonoff, R. J. 1957. "An Inductive Inference Machine." *IRE Convention Record, Section on Information Theory*. http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:An+Inductive+Inference+Machine#0.

Specht, D. F. 1991. "A General Regression Neural Network." *Neural Networks*, IEEE Transactions on 2 (6): 568–76. doi:10.1109/72.97934.

Tompson, J., A. Jain, Y. LeCun, and C. Bregler. 2014. "Joint Training of a Convolutional Network and a Graphical Model for Human Pose Estimation." *Advances in Neural Information Processing Systems*, 1799–807. http://arxiv.org/abs/1406.2984.

Wang, Y., Y. Fan, P. Bhatt, and C. Davatzikos. 2010. "High-Dimensional Pattern Regression Using Machine Learning: From Medical Images to Continuous Clinical Variables." *NeuroImage* 50 (4). Elsevier Inc.: 1519–35. doi:10.1016/j.neuroimage.2009.12.092.

Yusuff, H., N. Mohamad, U. K. Ngah, and A. S. Yahaya. 2012. "Breast Cancer Analysis Using Logistic Regression 1,2." *International Journal of Recent Research and Applied Studies* 10 (1): 14–22. http://www.arpapress.com/volumes/vol10issue1/ijrras_10_1_02.pdf.

Zhang, M.-l., and Z.-h. Zhou. 2007. "M L-KNN: A Lazy Learning Approach to Multi-Label Learning" 40: 2038–48. doi:10.1016/j.patcog.2006.12.019.

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

14 The IoT Revolution

Adrish Bhattacharya and Denim Datta

CONTENTS

14.1 What Is IoT? 230
14.1.1 Advantages 231
14.1.1.1 Smart Technology 231
14.1.1.2 User Engagement 231
14.1.1.3 Enhanced Data Collection 232
14.1.2 Limitations 233
14.1.2.1 Security 233
14.1.2.2 Privacy 234
14.1.2.3 Flexibility 235
14.1.2.4 Compliance 235
14.2 Components 236
14.2.1 AI and ML 236
14.2.2 Connectivity 236
14.2.3 Sensors and Modules 237
14.2.3.1 Optical Sensors 237
14.2.3.2 Chemical Sensor 237
14.2.3.3 Image Sensors 237
14.2.3.4 Accelerometer Sensors 237
14.2.3.5 Gyroscope Sensors 237
14.2.3.6 Water Quality Sensor 237
14.3 Microcontrollers for IoT 238
14.3.1 Arduino Uno 238
14.3.2 Arduino MKR1000 238
14.3.3 Raspberry Pi 2 Model B 238
14.3.4 Raspberry Pi 3 239
14.3.5 Intel Edison 239
14.3.6 Pinguino 45K50 239
14.3.7 Teensy 2.0 239
14.3.8 Nanode 240
14.3.9 Mediatek Linkit One 240
14.4 Applications and Everyday Electronics 240
14.4.1 Home Automation 241
14.4.2 Media and Advertisement 241
14.4.3 Manufacturing and Industry 242
14.4.4 Agriculture 243
14.4.5 Energy Management 243
14.4.6 Environmental Monitoring 244

14.4.7 Medical and Health Care 244
14.4.8 Transportation 244
14.5 Conclusion 245
References 245

14.1 WHAT IS IoT?

> The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.
>
> **Mark Weiser**

With the emergence of smart home solutions and the variety of smart health wearables in the market today, IoT (Ashton 2009) has brushed aside all skepticism to become the most disruptive technologies of the twenty-first century.

It is predicted that by 2020 about 7.6 billion people will be connected via 50 billion devices worldwide. Business analysts predict IoT will lead to an economic growth of 4.6 trillion dollars globally in the public sector by 2020. No wonder IoT has become the buzzword of the decade. It is perhaps the only technology which has successfully leveraged on the advancements in big data, artificial intelligence, and embedded systems (Wigmore 2014) to create new avenues for democratizing technology. Even preliminary discussions about IoT reveal its *behind the scene nature* with the simplest of IoT implementations focusing on smart systems.

Looking back we find that one of the first IoT devices was a Coke dispensing machine at the Carnegie Mellon University, which could not only keep records of its inventory, but could also determine if the drinks recently loaded were chilled. Besides being a pretty simple idea, it highlights the concept of an intelligent solution using the Internet. In fact it was Mark Weiser's 1991 paper on ubiquitous computing, "The Computer of the twenty-first Century," as well as academic venues such as UbiComp and PerCom who produced the contemporary vision of IoT (Weiser 1991; Mattern and Floerkemeier, 2010).

Although a variety of definitions exist, the term *Internet of things* or IoT was recognized officially in 2005, when the *ITU Internet Reports* defined it as:

> A new dimension has been added to the world of information and communication technologies (ICTs): from anytime, any place connectivity for anyone, we will now have connectivity for anything…. Connections will multiply and create an entirely new dynamic network of networks—an Internet of Things.

From these introductory definitions, we have come a long way since Kevin Ashton (then at Proctor and Gamble) coined the term "Internet of Things." Apart from IoT, similar concepts have been called *Internet of Everything (IoE)*, *Internet for things,* or *Smart Things,* etc.

When a number of physical devices fuse into a network capable of synchronizing several sensors (or actuators) and software components together for data exchange, the complete system is dubbed as the "Internet of things."

They can be considered as an "inextricable mixture of hardware, software, data, and service." A smart thing (Santucci, 2015) is one that connects to the Internet and

with other devices to exchange data to create an integrated environment (Vermesan and Friess 2013).

In simple terms, IoT provides an end-to-end solution, where the device acquires information continually (through sensors), processes them to produce meaningful data, which can be fed to other devices (as input to process further), or can be used directly by users. Due to the wide applicability of such an end-to-end solution, IoT will become an integral part of technology in the coming decades. Some argue that if all objects and people were equipped with identifiable tags such as RFID, computers would be able to manage and inventory them (Magrassi and Berg 2002; Wood 2015).

Since IoT is posed to be *the foundation for future infrastructure* and improve our quality of life in multiple areas (some of which are currently being envisioned, such as smart cities and IIoT), it is crucial we form a strong understanding of the challenges encountered and the advantages gained through an IoT-based process.

14.1.1 Advantages

The above definitions of IoT and the "things" in IoT provide us with a clear understanding of the core idea. After that brief discussion on the nature of IoT, we can form a structured idea about the nature of problems IoT can solve. Bearing in mind the characteristics of IoT, it is easy to understand that the way IoT interacts with other devices produces some unique advantages over other existing technology. Some of them have been mentioned below.

14.1.1.1 Smart Technology

The idea of smart technology has been around for some time. Computers are smart, our phones have becomes smarter, even our watches and TVs have become smart. Therefore, the claim of a newer, smarter technology can't account for its market disruptive behavior on just its claims of being smart. Speaking specifically of IoT, we are aware that machine learning (ML) and artificial intelligence (AI) form a key component of the technology. The main difference between traditional smart devices and AI powered devices is the latter evolves with the user. AI has has the potential of eliminating humans from the scene altogether—even though it may take another century, but who knows!

AI forms a fundamental part of IoT and when we consider a technology developed for evolving to future trends, we can truly say that finally we have something smart enough to be called truly *smart*.

14.1.1.2 User Engagement

Passive user engagement remains one of the key problems of analytics today. It not only leans on indirect methods of obtaining required data, but also suffers from incorrect conclusions due to the lack of exact data. These drawbacks are characteristic of any passive data collection system. The only method of overcoming these concerns is replacing passive data collection by "*active data collection*." It removes blind spots in data collection by increasing the *scope* of collection. IoT adds new areas of machine-to-machine interaction and human-to-machine interaction, thereby adding new opportunities for data assimilation. In today's world, humans are not the

only users of technology—machines have become the primary user with 50 billion machines expected to be connected to the Internet by 2020. With the ever-increasing number of devices in our lives and the advent of cheap and reliable sensors, operating machines has become more engaging and IoT has added a plethora of opportunities to improve upon them.

14.1.1.3 Enhanced Data Collection

Consider the following data:

1. Internet users send 204 million e-mails, tweet 278,000 times, and upload over 200,000 photos (just on Facebook) per minute.
2. By 2021, the estimated number of RFID tags sold is expected to reach 209 billion, from a mere 12 million in 2011.
3. Experts opine the *big data* industry is posed to reach US$54.3 billion (by 2017) in revenues, from US$10.2 billion (2013)—a fivefold increase (Figure 14.1).

The vastness of the information is mind blogging. It shows not only are we generating massive amounts of data every second, but ignoring this data can prove counterproductive. The kinds of conclusions that can be drawn from it have the potential of changing the very fabric of human–machine interaction. Also, we realize that the data is collected through active engagement with the users. It supports our previous claim of active user engagement for improving the scope of data analytics. The kind of data we are collecting each second has been available only due the efficient deployment of IoT solutions at specific locations. IoT solutions have made it simpler for machines to mine data from newer sources. In fact, IoT can be said to create sources of data collection. The IoT has thus created a more detailed and more exact picture of our world.

FIGURE 14.1 Growth of M2M connections since 2003.

14.1.2 Limitations

Like any other technology, IoT is no magic wand. Although it ventures into areas deemed unfathomable earlier, it brings along multifaceted solutions, and its very own set of limitations (Hendricks, 2015). If these are not understood completely, it can render the IoT revolution useless. Addressing these concerns, a study was conducted by Noura Aleisa and Karen Renaud at the University of Glasgow entitled "The Internet of things' potential for major privacy invasion is a concern." Also, Louis Basenese, an investment director at the *Wall Street Daily*, expressed that "Despite high-profile and alarming hacks, device manufacturers remain undeterred, focusing on profitability over security. Consumers need to have ultimate control over collected data, including the option to delete it if they choose… Without privacy assurances, wide-scale consumer adoption simply won't happen"(Basenese, 2012). According to the 2016 Accenture Digital Consumer Survey, which was taken by over 28,000 consumers across 28 countries, security "has moved from being a nagging problem to a top barrier as consumers are now choosing to abandon IoT devices and services over security concerns."

Therefore, we must open ourselves to these possible challenges and fully understand them. Among the many limitations in an IoT-based system, the most important are those involving security and privacy. Let us have a look at them one by one (Figure 14.2).

14.1.2.1 Security

Our basic understanding of IoT tells us it creates an ecosystem of continually connected devices communicating over numerous networks, which leaves the network exposed to various kinds of attacks: more devices lead to more opportunities for compromise. IoT systems' nature of *amorphous computing* is a major security concern for many. As it does not always allow the patch fixes to reach every subsystem in the entire net of devices (Franceschi-Bicchierai, 2015; Kingsley-Hughes, 2015). When security patches do not reach older, outdated systems, some estimate it renders 87% of active

FIGURE 14.2 Major limitations of IoT.

devices vulnerable (Tung, 2015). Additionally, it allows for postsale manipulations by corporations (Walsh, 2016).

The lack of strong security protocols fuel increased risks of other types of compromise, namely, sensor-related risks and ecosystem risks.

Sensor-related risk: IoT devices are susceptible to various types of sensor-related risks such as:

1. *Counterfeit products*: Duplicate products ingrained with malicious programs. At times only certain components are modified to allow unauthorized access to the system or subsystems. Identity spoofing is a commonly used technique for gaining such access.
2. *Information extraction*: Malicious code designed to extract sensitive information from any sensor or connected device(s).

Let us consider a situation where the ABC Corporation wants to use sensors to track its equipment. Now, for ABC Corporation, it is convenient and economical to adapt existing sensors to the IoT system designed for its procedures. But the sensors themselves have intelligence; they can independently process information and take their own course of action. Since the sensors and systems are designed independently, the generic sensors may not employ the strict security protocols the corporation requires. If malicious actors were able to break into such a device, they may cause it to malfunction or fail. These actors might even make attempts to steal sensitive data, manipulate other sensors within the network, or adversely impact plant operations.

The above example showcases how the lack of proper planning regarding the deployment of IoT solutions—for increased efficiency in operation—can ultimately prove counterproductive. The other kind of security risk is the ecosystem risk a compromised system poses.

14.1.2.1.1 Ecosystem Risk

It is the weakest link that determines the strength of a chain. Therefore, as corporations extend IoT systems to third parties, the user data flows through multiple devices and/or databases where each one of those devices or databases may be under independent third party control. These third parties include but are not limited to vendors, supply chains, network operators, cloud services providers, device manufacturers, and end users who may not recognize the implications of the lack of secure strategies used by each stakeholder. The complex nature of IoT ecosystems make it imperative for businesses to understand their shared responsibility in creating a robust security infrastructure capable of tackling threats of all kinds.

14.1.2.2 Privacy

Although the IoT claims to revolutionize active user engagement, it has already proved to be a great tool for passive engagement with users—especially in situations where a single device is used by multiple users, as for instance in public places.

Privacy concerns have nudged many experts into believing that infrastructures based on big data, such as IoT, are inherently incompatible with the concept of privacy. The recent example of hoardings and billboards having cameras hidden in them—to

track the demographics of all the commuters who showed a considerable interest in the specific advertisement, has been cited to support the claims of "invasion of public space." Considering the smart home systems, we can easily realize how the security and privacy of most households are susceptible to compromise by simple analysis of the smart home system traffic patterns.

The Internet of Things Council's comparison between Bentham's Panopticon and the increasing use of IoT-based digital monitoring systems is not unfounded. Editorials published in *WIRED* have expressed major concerns over this issue, one stating "*You aren't just going to lose your privacy, you're going to have to watch the very concept of privacy be rewritten under your nose*" (Webb, 2015). The American Civil Liberties Union (ACLU) has stated "*Chances are big data and the Internet of things will make it harder for us to control our own lives, as we grow increasingly transparent to powerful corporations and government institutions that are becoming more opaque to us*" (Howard, 2015).

Hence, we observe that despite the immense potential offered by IoT for empowering citizens and making governments transparent, there are mammoth-sized privacy threats such as the potential for *social control through political manipulation*. In fact, many fear the advent of IoT can lead to an Orwellian dystopia.

14.1.2.3 Flexibility

One of the major concerns regarding viability of an IoT-enabled system lies in its inability to integrate seamlessly into another system. Due to the lack of interoperability standards (Kovach, 2013; Raggett, 2016; Piedad 2015), it is a common belief that the potential of IoT may never be realized. IoT systems suffer heavily from platform fragmentation and the lack of technical standards (Ardiri, 2014; Bauer et al., 2015; Wallace, 2016; Wieland, 2016).

The incompatible and often conflicting protocols employed make it difficult to extend an IoT system (Franceschi-Bicchierai, 2015; Kingsley-Hughes, 2015). The key aspect of this problem can be ascribed to platform fragmentation, which is a situation where the sheer variety in IoT systems makes the task of developing consistent applications working with inconsistent platforms difficult (Brown, 2016). With proprietary systems there is always the risk of the technology fading out. These issues pertaining to the flexibility of IoT-based systems is a major hurdle in achieving mass adoption.

14.1.2.4 Compliance

IoT aims to democratize technology but the vastness of government regulations make compliance for any technology a tedious job. Adding to it the range of security and privacy concerns, we end up with a potentially groundbreaking system tangled in legal knots. Confusing terminology in the field of IoT is another hassle, which makes compliance difficult. However, regarding IIoT, the Industrial Internet Consortium's Vocabulary Task Group has created a "common and reusable vocabulary of terms," which ensures "consistent terminology" in all publications by the Industrial Internet Consortium.

Owing to the increasingly tougher regulation in the fields of privacy, future technologies like IoT, big data, and AI must prove themselves against stringent moral and ethical questions, as well as legal norms.

14.2 COMPONENTS

The concept of IoT originated from the amalgamation of various technologies such as AI (and ML), real-time analytics, wireless communication, and embedded systems. The advances in all of these fields have contributed to the current boom in IoT.

14.2.1 AI AND ML

The essence of IoT is in making objects "smart" and AI plays a pivotal role in achieving this, as AI and ML improve both accuracy and speed in big data analytics. Machine learning can be defined as "*a field that deals with the construction and study of systems that can learn from data, rather than follow only explicitly programmed instructions*." It can help corporations and governments alike in taking millions of collected data points and reducing them to meaningful information. It aims to analyze data collected in order to detect patterns, which can be further used to be learned from and for making better decisions. The realization of the IoT vision can be successful only through leveraging its ability of gaining the hidden insights obtained from the enormous and growing sea of data available. Owing to the sheer volume of data available today, it's time to let the machines take the front seat in pointing out the areas where opportunities truly lie.

14.2.2 CONNECTIVITY

IoT creates small networks between subsystems for the exchange of information (Figure 14.3). Therefore, it becomes imperative to have knowledge about the various means and protocols of connectivity. There are various short, medium, and

FIGURE 14.3 Components of an IoT system.

long-range connectivity solutions based on the needs of the users. A comprehensive list of commonly used wireless systems is provided below for quick reference.

- *Short-range systems*: Radio-frequency identification or RFID tags, near-field communication or NFC, Wi-Fi, Wi-Fi Direct and Li-Fi (Light-Fidelity), ZigBee, Bluetooth, QR codes, barcodes, and Z-Wave.
- *Medium-range systems*: LTE-Advanced, HaLow.
- *Long-range systems*: Long-range Wi-Fi connectivity, low-power wide-area networking (LPWAN), and very small aperture terminal (VSAT).

14.2.3 Sensors and Modules

Sensors form a key component of IoT systems, since in most designs they are the primary source of data collection. Without sensors it would be nearly impossible to collect the variety of data forming the basis of IoT systems. The following is a list of some of the most extensively used sensors in the IoT industry.

14.2.3.1 Optical Sensors

Measures different quantities simultaneously. The technology behind this sensor allows it to monitor electromagnetic energy, which includes electricity and light among others.

14.2.3.2 Chemical Sensor

Chemical sensors transform information regarding chemical properties such as the concentration of ions, chemical activity, etc. into useful signals.

Major uses of chemical sensors include CO detectors, glucose detectors, pregnancy test kits, and others.

14.2.3.3 Image Sensors

The image or imaging sensor determines and transmits information constituting any image. An image sensor gives an output of small packets of current.

14.2.3.4 Accelerometer Sensors

An accelerometer measures linear acceleration of any movement.

14.2.3.5 Gyroscope Sensors

The gyroscope, aka gyro, tracks information pertaining to rotation that an accelerometer can't determine. The output of a gyro is generally in the form of an angular rotational velocity.

14.2.3.6 Water Quality Sensor

These sensors are used to measure temperature, dissolved oxygen, conductivity, pH, and so on of the water sample. Major applications of water quality sensors include among others determining soil quality, monitoring water quality, water treatment, and modeling an ecosystem.

Other commonly used sensors include:

- Temperature sensors
- Proximity sensor
- Pressure sensor
- Gas sensor
- Smoke sensor
- IR sensors
- Motion detection sensors
- Humidity sensors

14.3 MICROCONTROLLERS FOR IoT

IoT prototyping boards play today an essential role in simplifying the prototyping process of IoT-based systems. These boards are *microcontrollers* and *microprocessors* with chipsets to handle wireless connections. These development boards with Cloud IoT platforms enable the quick deployment of IoT processes. The commonly used development boards include the following.

14.3.1 Arduino Uno

Arduino Uno is among the preferred development boards. It is an open-source development board based on ATmega328P.

It has both analog and digital pins. Another aspect of this board is it can be expanded using shields, that is, another board can be plugged into Arduino Uno to add new functionalities such as GSM, WIFI, etc.
Specifications:

Operating voltage: 5 V
8bit
16 MHz

14.3.2 Arduino MKR1000

Arduino MKR1000 is one of the latest boards based on the Atmel ATSAMW25.

Just like Arduino UNO, this board supports analog and digital pins. Additionally, it can be powered using an external Li-Po battery.
Specifications:

Operating voltage: 3.3 V 32bit
48 MHz WIFI

14.3.3 Raspberry Pi 2 Model B

Raspberry can be considered as a small computer owing to it's features. It is powered by Linux.

Specifications:

- Quad-core BCM2837, 900 MHz
- 1Gb RAM
- Four USB 2.0 ports
- 40 GPIO pins
- HDMI and RCA video output

14.3.4 Raspberry Pi 3

The newer Raspberry Pi 3 Model B has been built on its predecessors' features fused with a faster processor on board to enhance clock speed. It as the following specifications:

- Micro USB power source up to 2.5A
- 1.2 GHz 64-bit quad-core ARMv8 CPU
- 40 GPIO pins
- Four USB 2.0 ports
- 802.11n WLAN
- Bluetooth 4.1
- Bluetooth Low Energy

14.3.5 Intel Edison

Intel Edison is a very powerful IoT development board and is available in several variants.

Specifications:

- Intel Atom
- 500 MHz dual-core x86 1GB RAM
- Wi-Fi 802.11n
- Bluetooth v4.0

14.3.6 Pinguino 45K50

Built by Pinguino, it gives students and designers from the art community a powerful microprocessor. It has the following features:

- 8-bit 12 MIPS processor, 48 MHz
- 17 digital pins for both input and output
- 5 shared analog inputs
- 2 PWM outputs
- UART for serial data transfer

14.3.7 Teensy 2.0

The sole reason for developing Teensy 2.0 was to offer a great alternative to Arduino. It not only offers Arduino-like features but can also run the Arduino IDE allowing

Teensy 2.0 to access Arduino's extensive library. It flaunts 25 input and output pins and is powered by a 16 MHz AVR processor.

14.3.8 Nanode

The Nanode is a great alternative to the Arduino Uno, Mega, and Yun. The features of the Nanode include:

- ATMega328P, 16 MHz
- Red and green LEDs for program diagnostics
- Mini USB power connector
- ENC28J60 Ethernet controller, 25 MHz

14.3.9 Mediatek Linkit One

The Linkit One is based on the smallest SOC and comes with compatible Arduino pinout features. Linkit One can be used for rapid prototyping of connected IoT systems owing to its rich connectivity features. Its software development kit (SDK) comes with libraries for connecting the board to AWS and PubNub.

- MT2502A based chipset, 260 MHz
- GPS, GSM, GPRS, WiFi, and Bluetooth
- Supports Arduino shields

14.4 APPLICATIONS AND EVERYDAY ELECTRONICS

IoT has taken the center stage in making the 4th industrial revolution a reality. It can be exploited by almost every industry to deliver more efficient solutions, increase their scale of operation, and improve profitability (Westerlund et al., 2014).

Newer areas for deployment of IoT services are cropping up routinely, which undoubtedly means the *sky is the limit* for IoT applications.

The sheer variety of IoT applications creates a need to classify them. Although many different aspects can be considered for the classification of IoT applications, the most popular scheme is based on the end user, which is

- Consumer application—for the general consumer. For example: home automation solutions, wearable technology, etc.
- Enterprise (business) application—for providing business solutions to a myriad of enterprises. It is estimated that by 2019, enterprise IoT (EIoT) will account for 9.1 billion devices. For example: manufacture control, automatic shipment tracking, etc.
- Infrastructure applications application—for maintenance and improving infrastructure. For example: smart traffic control, automatic toll collection, etc.
- Health care application—for improving efficiency, accuracy, and precision of current medical instruments through responsive and integrated systems. For example: remote health monitoring system, wearable heart rate monitor, etc.

Listed below are some fields transformed by the ongoing IoT revolution. The IoT is finding newer applications every day with improvements in technology and innovative deployments. The given list is only meant as a reference to enable readers to grasp the scope of the practicality of IoT-based systems.

14.4.1 Home Automation

Smart home or home automation have been the most googled words last year in reference to IoT applications. An ever-growing number of consumers are excited about them, partly because almost every major electronics company, including Google and Amazon, have invested heavily in this area. Another strong reason for the general public to advocate for domotics (another name for home automation) is because of the plethora of opportunities it creates. The capabilities of even the most basic of smart home systems feel otherworldly. A recent study by Intel has revealed that an astonishing 71% of consumers expect to see "*at least one smart home device in every home as early as 2025.*" The buzz around Amazon Alexa and Google Home indicates we are at the cusp of a home automation revolution.

IoT-based home automation systems monitor and control the electrical and mechanical aspects of any place of residence including private and public areas. These aspects include lighting, air conditioning, and security solutions (Kang et al., 2017). The most appealing feature of these systems is their ease of use. However, the omnipresent security threats looming at the edges and the high initial establishment costs often outweigh the benefits.

Some commonly implemented features of home automation include:

- Providing assistance to the elderly by utilizing assistive technology to accommodate specific requirements such as voice control for users with limited sight and mobility, monitoring systems for medical emergencies, etc.
- Connecting the user with a home appliance regardless of geographical distance.
- Providing the user with the latest news while he/she is busy performing other chores (Figure 14.4).

14.4.2 Media and Advertisement

IoT has enabled a more ambitious targeted media era. The last few years have witnessed rapid development in the fields of conversation tracking, click rate, and so on. The data collected through these studies of consumer behavior has opened boundless opportunities for gaining meaningful insights into the salient traits of customers. The media industry is primarily concerned with maximizing the effect of advertisements and their content through known customer habits in line with their now predictable behaviors. IoT-based systems have allowed corporations to fine tune their advertisements according to individual needs, thereby improving customer experience. These detailed reports also enable business houses to make a better analysis of customer preferences and improve their own business strategies in the long run.

FIGURE 14.4 Common features of home automation.

Other than corporations, government agencies, TV soaps, and so on also stand to gain from this ever-growing range of insights into the masses. For instance, governments can now predict with far larger accuracy the behavior of their citizens during specific situation; TV soaps can alter the story of the show according to the perceptions of their viewers and so on.

14.4.3 Manufacturing and Industry

The manufacturing industry packs a series of processes within its scope of operation including maintaining logistical records of assets, supporting supply chain networks, dynamic time-response to manufacturing demands, and optimization of all of these tasks in real time. Control and management of these operations is within the purview of IoT systems used in the manufacturing industry. Digital control systems able to automate a myriad of process controls and successfully optimize manufacturing, industrial safety, and so forth through real-time analysis of critical parameters make IoT an indespensible aspect of future manufacturing industries.

The range of applications for IoT is so large that these IoT solutions and the systems delivering them have been dubbed the Industrial Internet of Things (IIoT). Basically, IIoT is the use of IoT technologies in the manufacturing sector. The IIoT holds immense potential for quality control, green practices, and profound improvements in supply chain dynamics. By some estimates, the potential for growth through IIoT is worth $12 trillion globally by 2030.

The IIoT combines a variety of recent trends including AI, big data, machine to machine (M2M) communication, and automation technologies, which have already existed in industrial settings for a long time (Perera et al., 2015). The replacement of humans by machines not only increases the efficiency of the process by limiting the scope of error, but also attempts to quantify the ineffective points in the system. The quantified information pertaining to the drawbacks of the systems in use allows companies to improve upon them easily and save valuable time and resources. In such a situation, IIoT is posed to become a key driver in growth and competitiveness of industries around the globe.

Although newer areas of application of IIoT are being discovered routinely, there is a common consensus about the main aspects of benefit from IIoT, namely:

1. Improving efficiency—IIoT has enabled enterprises to capture more data about industrial processes that provide critical information, which can help increase the efficiency of their business practices.
2. Innovative business models—due to the added capabilities of machines through improved connectivity and faster processing, new and innovative products have emerged. These new business models open up previously unexplored sources of revenue for corporations. As the industry has previously worked with the counterparts of IIoT, such as ML and AI, it can easily monetize its experience to create new streams of IoT services for their customers, requiring little external support.
3. Increasing safety—constant surveillance and making decisions on the fly based on the collected data is one of the key attributes of IoT. IIoT can thus help corporations to monitor manufacturing and related procedures, as well as produce alerts when sensing irregularities.

In today's economy where industry drives the governments' and countries' prosperities, it is essential to introduce smart machines to empower our markets to gather and analyze data in real time in order to make informed decisions.

14.4.4 Agriculture

Agriculture is a labor-intensive process relying largely on human intervention for successful production. The reason for this dependence is because the best farming techniques are decided based upon multiple environmental variables including temperature, humidity, humus content in soil, nutrient requirements, pests, etc. Now, with the use of various sensors linked to an IoT-based system, most of these parameters can be very accurately monitored for making informed decisions. Once decisions are made, various automation techniques can then be used to improve yields and reduce waste.

14.4.5 Energy Management

Integration of sensing and power-generating stations can potentially optimize power distribution and management through centralized control hubs. Experts predict IoT would enable the integration of all kinds of energy consuming devices

including lights, fans, and A/Cs among others to enable remote control of these devices. Apart from this, the data pertaining to the energy usage patterns can prove extremely valuable for smart grids. This information can be exploited to improve efficiency and reliability of power grids while ensuring sustainability in the production of energy.

14.4.6 Environmental Monitoring

Environmental monitoring is one the toughest and exhaustive fields of work, and it has benefited greatly from the advent of IoT. Although the lack of standardized wireless protocols and interoperability standards have stunted development in this area, nevertheless, IoT systems have been deployed on a major scale. They are being used mainly for environmental protection by monitoring air and water quality, atmospheric and soil conditions, etc. (Li et al., 2011). IoT has also assisted in monitoring the movements of wildlife and their habitats. Active research is currently underway to leverage IoT for developing early warning systems against natural calamities such as tsunamis.

14.4.7 Medical and Health Care

The most fundamental contribution of IoT in the area of medical care has been to improve the accuracy and efficiency of current instruments by providing real-time analytics and more precise information. It has drastically enhanced the collection of critical patient information, which had been impossible to assemble earlier.

IoT devices have made remote health monitoring and emergency notification systems a reality for blood pressure and heart rate monitors among others. Some additional innovative uses of IoT in the health care industry are:

- Smart beds detecting occupancy and movement in order to adjust themselves for ensuring appropriate pressure and support to the patient.
- Specialized sensors installed within living areas for the surveillance and evaluation of the general well being of senior citizens and infants.
- Consumer devices to encourage healthy living, including wearable heart monitors (Istepanian et al., 2011) to help in managing health vitals and recurring medical needs (Swan, 2012).
- Battery-powered prosthetic arms, powered by myoelectricity (converting muscle group sensations into motor control).

14.4.8 Transportation

Simplifying transportation has been one of the most extensive and successful applications of IoT. The integration of communication systems, information processing units, and control mechanisms across various transportation systems has enabled a dynamic interaction between transportation components such as the vehicle, driver, and IoT-based system for smart traffic and parking control, safety assistance, automated toll collection, etc.

14.5 CONCLUSION

With the rapid pace of developments in IoT and the ever-increasing interest in this field, IoT as undoubtedly become the star child of the century. However, despite sincere dedication challenges such as standardization, security, and so on remain constant. Apart from these, perhaps the biggest challenge today is the overhype regarding IoT. We must understand the abilities and limitations of any technology, and focus instead on viable solutions, which are not a "source of confusion for the end user." As Mike Farley quotes, "Instead of convincing consumers that they need complex systems to serve needs they don't have, we should fix real problems people struggle with every day."

Identifying the problem is the first step towards any solution (Yarmoluk, 2012), especially with IoT. We must first dedicate ourselves to understanding this technology before wielding it in order to create an efficient and sustainable ecosystem reaping benefits for all of humankind.

REFERENCES

Ardiri, A. 8 July 2014. "Will fragmentation of standards only hinder the true potential of the IoT industry?" evothings.com.

Ashton, K. 22 June 2009. "That 'Internet of Things' Thing."

Basenese, L. 2012. "The Best Play on the Internet of Things Trend." *Wall Street Daily.*

Bauer, H., M. Patel, and J. Veira October 2015. "*Internet of Things: Opportunities and Challenges for Semiconductor Companies*." McKinsey & Co.

Brown, E. 13 September 2016. "Who Needs the Internet of Things?" Linux.com.

Franceschi-Bicchierai, L. 2 August 2015. "Goodbye, Android." *Motherboard. Vice*.

Hendricks, D. 2015. "*The Trouble with the Internet of Things*." London Datastore. Greater London Authority.

Howard, P. N. 2015. Pax Technica: "How the internet of things May Set Us Free, Or Lock Us."

Istepanian, R. S. H., S. Hu, N. Y. Philip, and A. Sungoor. 2011. "The potential of Internet of m-health Things 'm-IoT' for non-invasive glucose level sensing." In *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, pp. 5264–5266. IEEE.

Kang, W. M., S. Y. Moon, and J. H. Park. 2017. "An enhanced security framework for home appliances in smart home." *Human-centric Computing and Information Sciences* 7, no. 1: 6.

Kingsley-Hughes, A. 2015. "The toxic hellstew survival guide." ZDnet.

Kovach, S. 30 July 2013. "Android Fragmentation Report." Business Insider.

Li, S., H. Wang, T. Xu, and G. Zhou. 2011. "Application Study on Internet of Things in Environment Protection Field." *Lecture Notes in Electrical Engineering* Volume 133: 99–106. ISBN 978-3-642-25991-3.

Magrassi, P., and T. Berg. 12 August 2002. "A World of Smart Objects." Gartner research report R-17-2243.

Mattern, F., and C. Floerkemeier. 2010. "From the Internet of Computers to the Internet of Things." *From Active Data Management to Event-based Systems and More*: 242–259.

Perera, C., C. H. Liu, and S. Jayawardena. 2015. "The emerging internet of things marketplace from an industrial perspective: A survey." *IEEE Transactions on Emerging Topics in Computing* 3, no. 4: 585–598.

Piedad, F. N. 2015. "Will Android fragmentation spoil its IoT appeal?" *TechBeacon*.

Raggett, D. 27 April 2016. "Countering Fragmentation with the Web of Things: Interoperability across IoT Platforms." W3C.

Santucci, G. 2015. "The internet of things: Between the revolution of the internet and the metamorphosis of objects." *Vision and Challenges for Realising the Internet of Things*: 11–24.

Swan, M. 2012. "Sensor mania! the internet of things, wearable computing, objective metrics, and the quantified self 2.0." *Journal of Sensor and Actuator Networks* 1, no. 3: 217–253.

Tung, L. 13 October 2015. "Android security a 'market for lemons' that leaves 87 percent vulnerable." zdnet.com.

Vermesan, O., and P. Friess, eds. 2013 "Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems." River Publishers.

Wallace, M. 19 February 2016. "Fragmentation is the enemy of the Internet of Things." Qualcomm.com.

Walsh, K. 5 April 2016. "Nest Reminds Customers That Ownership Isn't What It Used to Be." Electronic Frontier Foundation.

Webb, G. 5 February 2015. "Say Goodbye to Privacy." WIRED.

Weiser, M. 1991. "The computer for the 21st century." *Scientific American* 265, no. 3: 94–104.

Westerlund, M., S. Leminen, and M. Rajahonka 2014. "Designing business models for the internet of things." *Technology Innovation Management Review* 4, no. 7: 5.

Wieland, K. 25 February 2016. "IoT experts fret over fragmentation." *Mobile World*.

Wigmore, I. June 2014. "Internet of Things (IoT)." TechTarget.

Wood, A. 31 March 2015. "The internet of things is revolutionizing our lives, but standards are a must." The Guardian.

Yarmoluk, D. 2012 "5 Barriers to IIoT Adoption & How to Overcome Them." ATEK Access Technologies.

15 Healthcare IoT (H-IoT) *Applications and Ethical Concerns*

Srijita Banerjee, Adrish Bhattacharya, and Shampa Sen

CONTENTS

15.1 Introduction 247
15.2 Role in Health Care 248
15.2.1 Real-Time Location Services 248
15.2.2 Hygiene Monitoring 251
15.2.3 Remote Monitoring 253
15.2.4 Organ Donation and Transplantation 254
15.3 Ethical Obstacles 256
15.3.1 Privacy 256
15.3.2 Sensitivity 257
15.3.3 Ensuring Data Safety 258
15.3.4 Monitoring Consent 258
15.4 Future Prospects 259
References 259

15.1 INTRODUCTION

Internet connects various government-private-public and academic-business networks of local to global scope using optical networking technologies, via a broad array of electronic and wireless communication gadgets. It is mainly considered to be the storehouse of various information interlinked hypertext documents, applications of the World Wide Web (WWW), electronic mail, telephony, file sharing and many more. The evolution of the web and Internet has been vast, and can be categorized into five stages:

- *Advance Research Project Agency Network* (APRANET) was primarily designed for researchers and academicians.
- *The Gold Rush for domain names* had the objective of sharing product or service-related information.

- *The boom and bust of the dot com bubble*: An access was obtained for purchasing goods and services through Internet where companies like Amazon and eBay came into the picture.
- *The social and experience web*: The stage which customized Internet into a platform for social interaction (Facebook, Twitter, Instagram, WhatsApp).
- *IoT*: This concept deals with a hyperconnected society whereby an object is usually connected to various interconnected set-ups.

IoT, which usually incorporates traditional fields like embedded systems, control systems and automation, and wireless sensors to facilitate device-to-device communications through Internet was first coined in 1999 by Kevin Ashton (Kulkarni and Sathe 2014). The IoT can be chronicled as connecting everyday objects like smartphones, Internet, television sensors, and actuators to the Internet. Here, the devices are intelligently linked together enabling new forms of communication between objects and people, and between objects themselves. IoT can be applied to various areas, a prominent example being the health care sector. IoT has the potential to enhance varieties of medical applications such as remote monitoring, chronic diseases, organ donation, hygiene monitoring, real-time location services, and many more (Riazul Islam et al. 2015). With the advent of IoT, new data streams can be gathered, recorded, and analyzed faster and more accurately by enabling devices to collect and share information directly with the cloud and with each other, which impacts several application domains. These domains are classified based on the type of network availability, heterogeneity, scale, coverage, repeatability, and user involvement (Gluhak et al. 2011).

15.2 ROLE IN HEALTH CARE

Applying IoT to the health care sector has a number of advantages, such as reduced cost, device downtime, enhanced quality of services, and enriched experience for the user. This IoT-driven health care area is expected to help in the early diagnosis of various diseases and various medical emergencies (Riazul Islam et al. 2015). At the same time, medical servers are seen to have a cardinal importance in conceiving health records and delivering on demand health services to authorized stakeholders (Riazul Islam et al. 2015). Such systems not only allow faster delivery of health care solutions but also ensure accurate treatment for the patient.

The IoT-based health care network (H-IoT) supports access to the IoT backbone, which in turn facilitates the transmission and reception of medical data, thus enabling the use of health care-tailored communications. The H-IoT topology mainly deals with the arrangement of the various elements as described in (Figure 15.1).

H-IoT is developed in such a way it can contribute to profiling users as "health impaired" or "at risk" when needed (Rigby 2007).

15.2.1 Real-Time Location Services

Real-time location systems (RTLS) are tracking devices are used for identifying the location of an asset or person in real time, or near real time. Small ID badges or tags

FIGURE 15.1 Working of the H-IoT.

are attached to the object or the person of interest, which in turn remain connected through a wireless signal receiver to determine the location of the tagged entity. (Malik 2009) (Figure 15.2).

RTLS can be used to find and alert health care staff quickly in large facilities when any staff member or patient in the hospital is found in danger or during medical emergencies. For instance, patients suffering from Alzheimer's and dementia usually have a tendency to wander away, and this can be monitored using RTLS. It pinpoints the location of the patients and alerts the staff members. Newer technologies have also come up in which the doors are automatically locked when patiens come close. Some outdoor tracking devices related to this purpose also come up make use of a GPS (Boulos et al. 2011; Chang et al. 2005).

Devices have been developed in which RTLS analyses the overall well being of an individual by recording his/her mobility inside the house. This method includes calculating the daily distance walked by the individual based on the distance between each sensor he/she passes during a lapse of time (Graham 2012). Patient flow nowadays is also being tracked for throughput management, which is simplifying problems such as extended waiting time, overcrowding, boarding in outpatient clinics and emergency departments/rooms (ED/ER), postanesthesia care units (PACUs), and bumped and late surgeries (Drazen and Rhoads 2011; Stahl et al. 2011). Accidental

FIGURE 15.2 Real-time location device.

loss or theft can also be prevented by tracking expensive equipments, which can potentially lead to massive savings of time and money (Malik 2009).

Chances of nosocomial infections can be significantly decreased by making hand hygiene protocols smarter. Usage of RTLS in a low-cost method to record the time of hand-sanitation of the patients and the staff, after they enter or leave the room, can be an effective solution for this. The hand-wash areas can contain an electromagnetic field emitter, which stimulates the personal badge worn by the staff member, to transmit a "hand-washing event" notification identifying the staff member and the time during which a specific dispenser has been used (Malik 2009).

RTLS also has the ability to enhance the productivity of nurses and other caregivers, thus improving their job satisfaction by reducing any repetitive tasks normally experienced by staff on a daily basis. Such examples would be where RTLS can automatically detect the presence of instruments needed in a wardroom, instead of having the nurse do so manually. RTLS can also cut the time a staff spends checking the status of rooms and beds, or assist nurses by giving them a means to request emergency assistance during a crisis situation (Malik 2009).

Such systems can and also improve a patient's family/visitors' satisfaction by increasing their awareness on the patient's location and condition. Other than hospitals, RTLS was found to be beneficial in other applications as well. Fingerprint techniques have been developed (Lin et al. 2014), which deal with the matching of a fingerprint to some location dependent characteristic signal. This technique was mainly developed to provide safety services for the workers who work in the tunnels of high arch dam sites. It involved the determination of the position of a person with the help of a fingerprint map, which was developed using a K-nearest neighbor method. The fingerprint of the workers remains in the database while the position of the worker is determined (Lin et al. 2014) (Figure 15.3).

On conducting various surveys in US hospitals in 2008, 15% of administrators indicated their hospitals already had RTLS in place, and another 43% expressed their intent to purchase a system within the next two years (HIMSS [Health Care

FIGURE 15.3 Fingerprint matching system.

Information and Management Systems Society], 19th Annual—2008 HIMSS Leadership Survey 2008).

Among the numerous advantages of using RTLS in hospitals, certain drawbacks still prevail, such as the financial concerns and the lack of understanding of how RTLS can be integrated within the existing information technological investments (electronic medical record systems, for example) (Bouet and Pujolle 2010; Chung et al. 2010). Moreover, in a study conducted by Fisher and Monahan (2012), it was stated that:

- There was substandard functionality of most RTLS in hospitals.
- Several serious obstacles are faced by the hospitals during the effective deployment of the systems due to material and organizational constraints.

Various hospitals tend to consist of old buildings often arranged in complexes and unintuitive configurations. Considering these factors, the implementation of RTLS must go beyond the simple deployment of technology to be effective and useful (Dash 2009; Lorenzi et al. 2008). Before selecting a tracking technology, plans need to be designed detailing what goals the RTLS will meet, who will manage and operate the system, and how buy-in will be obtained from personnel(s) (Murphy 2006). RTLS should be chosen in such a way that specific goals are matched in the best way considering their own facilities, given their unique material and organizational constraints (Fisher and Monahan 2012). If we prioritize from the *hospital and health care context*, RTLS can perhaps be influenced to better serve the health care industry.

15.2.2 Hygiene Monitoring

On a statistical basis, there are 1.7 million cases of public health care-acquired illnesses each year in the United States. These cases result in 100,000 deaths each year, which ranks as the fourth-leading cause of U.S. deaths and cost hospitals approximately $30 billion annually. Studies show that poor hand hygiene practices are the most likely cause of the spread of various bacterial diseases (Centers for Disease Control and Prevention 2015). Hand hygiene, when maintained properly, is the single biggest defense against the spread of numerous diseases, especially in public health care institutions such as hospitals and nursing homes (World Health Organization (WHO) 2017). For all these reasons it is important for hospitals, business authorities, and public health agencies to consistently monitor hand hygiene among workers to control the spread of various bacterial diseases.

The methods involved in IoT hygiene monitoring range from simple hygiene dispensers and sprays to more sophisticated systems using advanced technologies such as *wireless communication* and *machine learning* in electronic monitoring of hand hygiene.

One of the simplest IoT-based systems included a battery-powered recording device incorporating a force-sensitive resistor and a microcontroller. The microcontroller in this unit has the function of recording the reduction in the amount of wall-mounted soaps and alcohol dispensers in hospitals. This device calculated the number of times the soap dispensers were used. However, this system was found to be disadvantageous,

FIGURE 15.4 Components of hygiene dispenser.

as it was not able to assess hand-cleaning episodes delivered from other nonfixed dispensers (i.e., personal gel dispensers), thus giving only a partial picture of the total hygiene (Kinsella et al. 2007).

Another system presented in 2013 had a function for monitoring the hand-hygiene compliance of caregivers, or individuals in hospitals, by using a passive RFID wearable wristband and wall-mounted dispensers. The sanitizer dispensers are equipped with RFID readers and ZigBee wireless transceivers. On activation of the dispenser the action gets registered by a centralized software system, which is then used for monitoring the hand-hygiene compliance (Meydanci et al. 2013) (Figure 15.4).

"*Smart hospital room*" is another technique using the RFID approach to monitor hand hygiene in hospital rooms. RFID tags are used in this system to record the movements of caregivers as they enter the patient rooms. It works by activation or deactivation based on the caregivers' location, and thus behaves as a hand hygiene reminder. Sometimes, it also logs the time used for the hand hygiene activity (Luen and Ellen 2009).

In the study presented by Rhodes (2014), the hand hygiene compliance was developed in such a way that duration and techniques of hand hygiene activities were recorded on a particular hand hygiene station using wrist-worn accelerometers (using machine learning techniques).

Nowadays, in addition to monitoring, systems have been developed giving importance to enforcement of the hand hygiene compliance. Biovigil Hygiene Technologies has developed a system involving changes in color of a badge (used by the staff members) based on whether they have washed their hands or not. The badge electronically measures the alcohol and turns bright green if alcohol is detected (and red if not) (Rhodes 2014) (Figure 15.5).

FIGURE 15.5 Hygiene Monitoring system.

A different product from Pure Hold Ltd has an automated sanitizer gel dispenser installed within a standard pull door handle with a specific design where the gel is emitted through a valve upon a pull grip. After operating a patient or opening any operating room, the user clean their hands with gel thus preventing staff, other patients, and visitors from transmitting and acquiring infections through touching door handles (Emery 2015).

A system was constructed by Bal and Abrishambaf (2017) consisting of a node in a wireless network sensor (WNS). The node represents the hand wash station and the data concerning the use of hand washing or hand-sanitizing gets transferred to the data center, which serves as a cloud to monitor the usage statistics of the hand washing stations and generate reports in real time. Health care facilities usually have an access to these reports. The basic form of the soap dispenser is automated and wall mounted. The system uses an infrared sensor to detect the presence of a hand and dispenses a controlled amount of hand soap or gel hand sanitizer. The soap dispenser module contains an RFID reader and an antenna encapsulated in the housing. These readers scan personalized passive RFID name tags, which are carried by the user associated with the hospital (Bal and Abrishambaf 2017).

15.2.3 Remote Monitoring

Remote monitoring in IoT mainly deals with the patients who fail to visit doctors regularly. In this case, the collected data is taken from the patients' home and sent to the cloud for doctors to review. For example, we can consider the presence of an ECG monitoring application downloaded on a mobile device collecting the biosignal data using a microcontroller, and then uploading it to the cloud for keeping a record of the unstructured data. Benefits are a reduction in waiting time for triage at the hospital, minimizing visits, reduction in the cost of the personnel and administrative operations, etc. Another word for this can be *telemedicine*, which can be explained as usage of telecommunication technologies as means to provide medical diagnosis, treatment, education, and health care for patients (Istepanian 1999).

A device able to check the health parameters of a patient at any place or time (Pont 2002) is actually a reality with the smart application of IoT solutions to health care. The only thing the person needs to do is place his finger on the device, which will sense the necessary parameters and notify the doctor through the wireless network;

the doctor then can prescribe the patient with the appropriate medicines. Sometimes, if the medical conditions are severe then the doctor gets to decide how to treat the patient. (Shelar et al. 2013; Gayakwad n.d.).

Another example includes a remote monitoring system developed in 2008 specifically developed for ECG monitoring. The system provides remote monitoring for patients wearing portable equipment (which is equipped with ZigBee module connectivity based on WSN). ZigBee is a low-power, low data rate, and close proximity (i.e., personal area) wireless ad hoc network, which can be efficiently employed to implement such solutions. The ECG signals are analyzed and recorded in an online database. On detection of any serious aberrations in heartbeat, an alarm is sent to the authorized medical staff through a telecommunication network (Khanja et al. 2008).

A.M. Cheriyan developed an H-IoT design embedded within a system capable of tracking biosignals from a person in real time. This system facilitated a dependable decision-making process in real time by incorporating the use of electroencephalogram (EEG) and oxygen saturation (SpO_2) signals, which provided alerts for possible changes in brain activity (Cheriyan et al. 2009).

Another group of scientists developed an original ECG measurement system based on a web service-oriented architecture capable of monitoring the heart conditions of patients. The device was developed in such a way to provide personalized diagnosis by using the personal data and clinical history of the patients being monitored (De Capua et al. 2010).

Although these models are efficient, poor bandwidth have limited the number of channels and signals obtained, which were degraded due to noise.

Yuan and Herbert (2011) presented a new setup, CARA, a health care architecture essentially focusing on web-based health monitoring systems. It is constituted of four parts—*the wireless monitoring device (WMD), the home monitoring system, the remote clinical monitoring, and the health care reasoning system*. This whole system included the real-time analysis of vital signs of patients resulting in automated responses. It was developed in such a way as to avoid inherent problems of data errors in wearable sensors (Yuan and Herbert 2011).

15.2.4 Organ Donation and Transplantation

Organ transplantation, or organ donation, is a very complex process requiring automation in different services. IoT-related technologies like RFID, RTLS, Real Time communication technologies, etc. are nowadays being heavily implemented to improve patient safety and enhance clinical operations. In the case of organ transplantation, the demand for organs far outweighs supply (Talmale and Shrawankar 2017) (Figure 15.6).

Previously, communications related to human organ transplantation processes relied on phone or fax. This system suffered from several drawbacks, for instance when difficulties arose in keeping track of patients in need of vital organs, where it led to a gap between supply of organs and their demand. Other than that, the increase in time to allocate organs due communication gaps between different stockholders of the organ transplantation processes were prevalent (Asberg et al. 2011).

FIGURE 15.6 Role of IoT in organ transplantation.

Therefore, it's clear that the human organ transplantation process comprises many services, which can be implemented using IoT. Some of these services are mentioned below:

- *Blood, organ and tissue cell banking*: An online database system can be maintained for blood, organ, tissue, and cell banks, which is accessible from any part of the world. It should have the capability to manage a national transplant waiting list, and match the donor details to those of the recipient (Bastoni et al. 2010). It should hold all kinds of data related to every transplantation event. To ensure the effective utilization of the limited supply of organs, this should be a common platform for all stakeholders (Hahm et al. 2015). Constant monitoring of the organ match process should be used to ensure proper working of the system. Nowadays, even sensors have been infused in such systems able to monitor the different parameters simultaneously (Talmale and Shrawankar 2017).
- *Real-Time Information to Physician*: This is also an important aspect of organ donation, where the physician is constantly kept updated about the status of the recipient and the organ. This is usually maintained by real-time monitoring of ambulatory patients (Talmale and Shrawankar 2017).
- *Drug and Medical devices monitoring*: It entails the monitoring of medical devices, biological and prescription drugs, and the patients' food. It involves identifying the risks of various drugs and medication for patients and notifying them, which enhances patient surveillance and produces a reduction in errors caused by medical instruments. It also deals with the increase in efficiency of product management with product validation and tracking. In this regard, H-IoT can help in managing medical sponges used during organ transplants, surgical processes, etc. and alerts can be given if no sponges are issued or matches are not found (Talmale and Shrawankar 2017).

An IoT transplantation system consists of a number of subsystems like pretransplant, inpatient, clinicians, organ provider organization, donor care, recipient care etc. The

FIGURE 15.7 Aspects of IoT-based organ transplantation.

inpatient transplant subsystem mainly deals with constant monitoring and caring of inpatients to prevent the organ rejection or failure.

It also includes tasks performed to manage dosages of medicines required in organ transplantation. Moving on, the posttransplantation subsystem consists of care and monitoring units including systems for managing the long-term medication after transplantation (Rosenkranz et al. 2015). The posttransplantation subsystem also accommodates tasks where real-time patient data is evaluated. These systems notify the doctor whenever changes in the lab settings are required and help to remotely monitor medications such as antiviral drugs, etc (Adjih et al. 2015) (Figure 15.7).

15.3 ETHICAL OBSTACLES

Despite the numerous groundbreaking applications of IoT in the health care sector, it raises a host of ethical concerns pertaining to the sensitive nature of any health-related data generated and analyzed. Furthermore, it becomes impossible for the users to retain control over the data produced and analyzed owing to the scale, scope, and complexity of such systems (Edgar 2005; Pellegrino 1993).

15.3.1 Privacy

The majority of H-IoT applications empower users to control collection and storage of personal data thereby limiting the opportunities for unwanted profiteering from sensitive information (Wel and Royakkers 2004).

This sensitivity of the health care data has been prominently recognized in international data protection and privacy laws (Terry 2014). Due to continued pressure from human rights' groups, protection of the user's privacy has become a critical concern for the design of H-IoT (Shilton 2012). For all these reasons, techniques such as anonymization and aggregation are being included, which are thought to reduce the risk of reidentification, and thus guarantee the privacy of a user (Joly et al. 2012). Policies allow access to identifiable data, only when the purpose(s) is (are) acceptable (Beaudin et al. 2006; Chakraborty et al. 2011; Giannotti and Saygin 2010; Massacci et al. 2009). Other than privacy, another essential feature of H-IoT infrastructure is security. Security in health care systems is mainly concerned with preventing hacks or breaches, thus also helping to protect privacy (Peppet 2014). Although reasonable exceptions are allowed by the system manufacturer to the user or to any third party, with regard to storage

and distribution of information, the key factor required for such provisions is the trust between users and the manufacturers (Liyanage et al. 2014). After taking all these factors into account, several privacy principles have been developed.

- *Proactive not reactive, preventative not remedial*: This is aimed at anticipation and prevention of security hacks and privacy infractions. These principles discourage spyware infiltrations from materializing by placing robust security infrastructure.
- *Privacy as the default setting*: Building IT systems and business practices such that privacy protection becomes a "by default" operation and not an additional feature by embedding privacy into the design, such systems integrate privacy safeguards into the design of IT systems, and make them a core component of business practices.
- *Full functionality*: Accommodating all the legitimate interests and the objectives without compromising efficiency.
- *End-to-end security—full lifestyle protection*: This principle talks about embedding the privacy protection into the system prior to the first element of information being collected in order to avoid leaks of even the most seemingly trivial information.
- *Visibility and* t*ransparency—keeping it open*: Assuring all the stakeholders through independent verification that the relevant business practices or technologies are "operating according to the stated promises and objectives."
- *Respect for user privacy—keeping it user-centric*: This ensures that the operators are keeping the interests of the individuals at the center by offering measures such as strong security measures (Cavoukian 2011).

15.3.2 Sensitivity

When it comes to the generation of data and their transfer, a lot of ethical concerns arise. The users ought to be included in the H-IoT architecture design wherever possible (Nissenbaum 2004). H-IoT designing should be done in such a manner benefiting the interests of specific groups of people, while simultaneously allowing individual control of features and privacy policies. For avoiding marginalization or exclusion of certain groups using H-IoT, enhancement in the accessibility scheme of the device and protocols can be put in place. These devices are generally so developed that an equity of access is maintained for the sake of commercial interests, and the interests of affluent users are ignored to prevent marginalization (Mittelstadt and Floridi 2016).

H-IoT can raise a host of ethical problems arising from the inherent risks of Internet-enabled devices, the sensitivity of health-related data and their impact on the delivery of health care. Being scientifically and technologically reliable are two of the most important criteria, or rather challenges, of H-IoT. Along with this, users also expect it to be ethically responsible and trustworthy. Other than this, the device needs to be made in such a way that it strikes a balance between preserving the autonomy of the individuals on their life (Mittelstadt 2017). Ultimately, access to information should be under the control of the patient or their appointed guardian to prevent misuse by any third party.

15.3.3 Ensuring Data Safety

Data safety can be brought out in several different ways.

- *Maintaining proper trust and confidentiality between the users and providers*: To contribute to the trustworthiness of the users and providers, proper design considering the key interest of the individuals, groups, and society is the priority for developing the product. Privacy, confidentiality, safety, and efficacy are closely linked to trust (Kosta et al. 2010). A user can always expect distress and anguish if their providers are untrustworthy, even if the system enhances privacy, confidentiality, safety, and effectiveness (Bagüés et al. 2010). Confidentiality is one of the critical elements for establishing a trustworthy relationship between the H-IoT users and providers. To handle the data responsibly and to ensure it's safety, users commonly place their trust in the devices and service providers (Little and Briggs 2009). Establishing trust is so important that a lack of trust has been linked to the reluctance to adopt H-IoT (Hildebrandt and Koops 2010).
- *Transparency and accountability of the data processing protocols*: H-IoT has got the capability to collect large amounts of data and process it in a complex and opaque manner. If the data is being processed and exchanged without properly letting the data subject (I.e., the user) know, a critical explanation is required about the handling. The social impact of decisions made based on the data is increasing, as well as the need for transparency when the rights of the data subjects are being kept constant (Hildebrandt and Koops 2010). When IoT is brought into picture, usually the guidelines proposed by TechUK, is followed. One school of thought argues that obtaining data using IoT without the involvement of users is unethical. Sometimes, in order to earn proper rights to derive value from IoT data, data processors are required to demonstrate the transparency, integrity, and security in processing to the general public. This in turn empowers the users to believe the impact of H-IoT on their medical care and quality of life. This impact can be assessed by checking the degree to which the providers adhere to the preceding eight principles in designing and deploying H-IoT devices and data protocols. Transparency is sometimes required to explain how the data produced might influence the medical care of the user (Wachter et al. 2017). These explanations can be directed to both the user and the care team, who may be better informed with technical knowledge to understand and modify the user's treatment (Giannotti and Saygin 2010).

15.3.4 Monitoring Consent

Manufacturers should always consider the potential value of the data generated even if there is uncertainty about the future value of the data in academic research and commercial analytics. Sometimes participants are de-identified (Ioannidis 2013), or data generated are repurposed for medical applications (Taddeo 2016), owing

to lack of consent between H-IoT manufacturers and users. For this reason, device manufacturers are advised to design user agreements to represent fairly the uncertain value of the data generated and the amount of aggregation and linkages to any third party for research and commercial purposes. Consent is usually considered when there is the participation for a single study and which does not include unrelated investigation resulting in sharing or aggregation or even repurposing of data within the wider research community (Choudhury et al. 2014). Ethical problems occur when H-IoT leads to psychological disappearance, for instance, when used in personal spaces or residential care surroundings (Ebersold and Glass 2016). Some H-IoTs' also involve the use of sensors inside the private sphere. For these reasons, consent is required in case the user forgets that monitoring is occurring inside their homes. Nowadays, occasional renewal of consents has been developed, which ensures monitoring has not been forgotten. This kind of system is mainly required for the cognitive impaired users who are unable to grant consent. Concerns are extended to the guests, which usually suggests the possibility of careless monitoring (Kenner 2008).

15.4 FUTURE PROSPECTS

By actuating the potential of existing services of IoT, various technological solutions have been designed to enhance health care provisions in multiple ways. Till now abundant research and development efforts have been carried out in IoT-driven health care appositeness and applicability (Riazul Islam et al. 2015). From all these we can safely conclude that IoT will change our society and bring coherent personalized health care and monitoring over fast, reliable networks. This in all likelihood indicates that the world is approaching towards the end of the present divide between digital, virtual, and physical worlds. Standard web service technology is one of the most widely adapted technologies for the Internet right now (Kulkarni and Sathe 2014). Similar kinds of efforts are going on for the development of networks with wireless identifiable embedded health care systems. Various plans have been made for the development of WSNs and ubiquitous networks, where the sensors will be connected to and controlled by embedded systems, and where services encapsulate the functionality and provide unified access to the system. These components produce, consume, and process information in different health care environments such as hospitals, households, and nursing homes. This is also the case in the work and everyday lives of people (Sundmaeker et al. 2010), thus making round the clock, affordable health care a reality for the masses.

REFERENCES

Adjih, C., Baccelli, E., Fleury, E., Harter, G., Mitton, N., Noel, T., Pissard-Gibollet, R. et al. 2015. "FIT IoT-LAB: A Large Scale Open Experimental IoT Testbed." *The 2nd IEEE World Forum on Internet of Things (WF-IoT).*

Asberg, M., Nolte, T., and Kato, S. 2011. "A Loadable Task Execution Recorder for Hierarchical Scheduling in Linux." *17th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications* 380–387.

Bagüés, S. A., Zeidler, A., Klein, C., Valdivielso, F., and Matias R. 2010. "Enabling personal privacy for pervasive computing environments." *Journal of Universal Computer Science* 16, 341–371.

Bal, M., and Abrishambaf, R. 2017. "*A System for Monitoring Hand Hygiene Compliance based-on Internet-of-Things*." DOI: 10.1109/ICIT.2017.7915560.

Bastoni, A., Brandenburg, B. B., and Anderson, J. H. 2010. "An empirical comparison of global, partitioned, and clustered multiprocessor edf schedulers." *31st IEEE Real-Time Systems Symposium, RTSS '10*. USA: IEEE. 14–24.

Beaudin, J., Intille, S., and Morris, M. 2006. "To track or not to track: User reactions to concepts in longitudinal health monitoring." *Journal of Medical Internet Research* 8, e29.

Bouet, M., and Pujolle, G. 2010. "RFID in ehealth systems: Applications challenges, and perspectives." *Annals of Telecommunications* 65, 497–500.

Boulos, M. N. K., Anastasiou, A., Bekiaris, E., and Panou, M. 2011. "Geo-enabled technologies for independent living: Examples from four European projects." *Technol Disabil* 23, 7–17.

Cavoukian, A. 2011. *The Privacy by Design: 7 Foundational Principles*.

Centers for Disease Control and Prevention. 2015. *Healthcare associated Infections (HAIs): Data and Statistics*. https://www.cdc.gov/hai/surveillance/index.html.

Chakraborty, S., Choi, H., and Srivastava, M. B. 2011. "Demystifying privacy in sensory data: A QoI based approach." *9th IEEE International Conference on Pervasive Computing and Communications Workshops, PERCOM Workshops*. Seattle. 38–43.

Chang, I. C., Cheng-Yaw, L., Yu-Chuan, L., Chia-Cheng, C., Chien-Tsai, L., and Chieh-Feng, C. et al. 2005. "Pervasive observation medicine: The application of RFID to improve patient safety in observation unit of hospital emergency department." *Connecting Medical Informatics and Bio-informatics Proceedings of MIE* 2005, 311–320.

Cheriyan, A. M. et al. 2009. "Pervasive embedded real time monitoring of EEG & SpO2." *3rd International Conference on PervasiveHealth 2009*. Pervasive Computing Technologies for Healthcare. 1–4.

Choudhury, S., Fishman, J. R., McGowan, M. L., and Juengst, E. T. 2014. "Big data, open science and the brain: Lessons learned from genomics." *Frontiers in Human Neuroscience* 8, 239.

Chung, Y.-C., Shih, P.-C., Li, K.-C., Yang, C.-T., Hsu, C.-H., and Hsu, F.-R. et al. 2010. "Medicare-grid: New trends on the development of E-health system based on grid technology." *E-health*, Springer, Boston. 335, 148–150.

Dash, A. 2009. "Lost + found: Making the right choice in equipment location systems." *Health Facilities Management* 22, 19–20.

De Capua, C., Meduri, A., and Morello, R. 2010. "A smart ECG measurement system based on web-service-oriented architecture for telemedicine applications." *Instrumentation and Measurement, IEEE Transactions* 59, 2530–2538.

Drazen, E., and Rhoads, J. 2011. "*Using tracking tools to improve patient flow in hospitals*." California Health Care Foundation.

Ebersold, K., and Glass, R. 2016. "The internet of things: A cause for ethical concern." *Issues in Information Systems*.

Edgar, A. 2005. "The expert patient: Illness as practice." *Medicine, Health Care, and Philosophy* 8, 165–171.

Emery, G. 2015. *Enforcing Rather than Encouraging Hand Hygiene in Healthcare*.

Fisher, J. A., and Monahan, T. 2012. "Evaluation of real-time location systems in their hospital context." *International Journal of Medical Informatics* 81, 705–712.

Gayakwad, R. A. n.d. In *Op-Amps and Linear Integrated Circuits*, 4th Edition, 342, 417, 455. New York: Prentice-Hall.

Giannotti, F., and Saygin, Y. 2010. "Privacy and security in ubiquitous knowledge discovery." *Lecture Notes in Computer Science* 6202, 75–89. *Berlin*.

Gluhak, A., Krco, S., Nati, M., Pfisterer, D., Mitton, N., and Razafindralambo, T. 2011. "A survey on facilities for experimental Internet of Things Research." *IEEE Communications Magazine* 58–67.

Graham, E. 2012. "My Amego relationship centred telecare: Using location-based technology to build relationships and manage risk in care homes." *Online Proceedings of 'Location-Based Technologies in Health & Care Services'*.

Hahm, O., Baccelli, E., Petersen, H., and Tsiftes, N. 2015. "Operating system for low-end device in internet of things: A servey." *IEEE Internet of Things Journal* 3, 720–734.

Hildebrandt, M., and Koops, B. J. 2010. "The challenges of ambient law and legal protection in the profiling era." *The Modern Law Review* 73, 428–460.

HIMSS. 2008. "HIMSS (Healthcare Information and Management Systems Society), 19th Annual – 2008 HIMSS Leadership Survey." *Healthcare CIO*.

Ioannidis, J. P. A. 2013. "Informed consent, big data, and the oxymoron of research that is not research." *American Journal of Bioethics* 13, 40–42.

Istepanian, R. S. H. 1999. "Telemedicine in the United Kingdom: Current status and future prospects." *IEEE Transactions on Information Technology in Biomedicine* 3, 158–159.

Joly, Y., Dove, E. S., Knoppers, B. M., Bobrow, M., and Chalmers, D. 2012. "Data sharing in the post-genomic world: The experience of the International Cancer Genome Consortium (ICGC) Data Access Compliance Office." *PLoS Computational Biology* 8, e1002549.

Kenner, A. M. 2008. "Securing the elderly body: Dementia, surveillance, and the politics of 'aging in place'." *Surveillance & Society* 5, 252–269.

Khanja, P., Wattanasirichaigoon, S., Natwichai, J., Ramingwong, L., and Noimanee, S. 2008. "A web base system for ecg data transferred using zigbee/IEEE technology." *The 3rd International Symposium on Biomedical Engineering*.

Kinsella, G., Thomas, A. N., and Taylor, R. J. 2007. "Electronic surveillance of wall mounted soap and alcohol gel dispensers in an intensive care unit." *Journal of Hospital Infection* 66, 34–39.

Kosta, E., Pitkänen, O., Niemelä, M., and Kaasinen, E. 2010. "Mobile-centric ambient intelligence in Health- and Homecare-anticipating ethical and legal challenges." *Science and Engineering Ethics* 16, 303–323.

Kulkarni, A., and Sathe, S. 2014. "Healthcare applications of the Internet of Things: A Review." *International Journal of Computer Science and Information Technologies (IJCSIT)* 5, 6229–6232.

Lin, P., Li, Q., Fan, Q., Gao, X., and Hu, S. 2014. "*A Real-Time Location-Based Services System Using WiFi Fingerprinting Algorithm for Safety Risk Assessment of Workers in Tunnels*." Hindawi.

Little, L., and Briggs, P. 2009. "Pervasive healthcare: The elderly perspectives." In *Proceedings of the 2nd International Conference on PErvasive Technologies Related to Assistive Environments*. Corfu.

Liyanage, H., de Lusignan, S., Liaw, S. T., Kuziemsky, C. E., Mold, F., Krause, P., Fleming, D., and Jones, S. 2014. "Big data usage patterns in the health care domain: A use case driven approach applied to the assessment of vaccination benefits and risks." *Yearbook of Medical Informatics* 9, 27–35.

Lorenzi, N. M., Novak, L. L., Weiss, J. B., Gadd, C. S., and Unertl, K. M. 2008. "Crossing the implementation chasm: A proposal for bold action." *Journal of the American Medical Informatics Association* 15, 290–300.

Luen, D., and Ellen, Y. 2009. "Technological interventions for hand hygiene adherence: Research and intervention for smart patient room." *In joining Languages, Cultures and Visions: Proceedings of the 13th International CAAD Futures Conference*. Canada: Les Presses de lUniversité de Montréal. 303–313.

Malik, A. 2009. "*RTLS For Dummies*." Hoboken: Wiley Publishing.

Massacci, F., Nguyen, V. H., and Saidane, A. 2009. "No purpose, no data: Goal-oriented access control forambient assisted living." *1st ACM Workshop on Security and Privacy in Medical and Home-Care Systems, SPIMACS'09, Co-Located with the 16th ACM Computer and Communications Security Conference*. Chicago. 53–57.

Meydanci, M. A., Adali, C., Ertas, M., Dizbay, M., and Akan, A. 2013. "RFID based hand hygiene compliance monitoring station." *2013 IEE International Conference on Control System, Computing and Engineering*. Mindeb. 573–576.

Mittelstadt, B. 2017. "Designing the health-related internet of things: Ethical principles and guidelines." *Information* 8, 22–25.

Mittelstadt, B., and Floridi, L. 2016. "The ethics of big data: Current and foreseeable issues in biomedical contexts." *Science and Engineering Ethics* 22, 303–341.

Murphy, D. 2006. "Is RFID right for your organization?" *Materials Management in Health Care* 15, 28–30.

Nissenbaum, H. 2004. "Privacy as Contextual Integrity." *Social Science Research Network: Rochester,* NY, USA, 22–25.

Pellegrino, T. 1993. *The Virtues in Medical Practice*. New York: Oxford University Press.

Peppet, S. R. 2014. "Regulating the internet of things: First steps toward managing discrimination, privacy, security and consent." *Texas Law Review* 93, 85–178.

Pont, M. J. 2002. In *Embedded C, Edition 2002*, 57–87, 217. New York: Addison-Wesley.

Rhodes, M. 2014. "*A Gadget Designed to Finally Make Doctors Wash Their Hands Enough*."

Riazul Islam, S. M., Kwak, D., Humaun Kabir, M. D., Hossain, M., and Kwak, K.-S. 2015. "The internet of things for health care: A comprehansive survey." *IEEE Access* 3, 678–708.

Rigby, M. 2007. "Applying emergent ubiquitous technologies in health: The need to respond to new challenges of opportunity, expectation, and responsibility." *International Journal of Medical Informatics* 76, 349–352.

Rosenkranz, P., Wählisch, M., Baccelli, E., and Ortmann, L. 2015. "A distributed test system architecture for open-source IoT software." *ACM MobiSys Workshop on IoT Challenges in Mobile and Industrial Systems (IoT-Sys)*. DOI: 10.1145/2753476.2753481.

Shelar, M., Singh, J., and Tiwari, M. 2013. "Wireless patient health monitoring system." *International Journal of Computer Application* 62, 975–8887.

Shilton, K. 2012. "Participatory personal data: An emerging research challenge for the information sciences." *Journal of the American Society for Information Science and Technology* 63, 1905–1915.

Stahl, J. E., Drew, M. A., Leone, D., Crowley, R. S. 2011. "Measuring process change in primary care using real-time location systems: Feasibility and the results of a natural experiment." *Technol Health Care* 19, 415–421.

Sundmaeker, H., Guillemin, P., Friess, P., Woelffle, S. 2010. "Outlook on Future IoT applications," *Vision and Challenges for Realising the Internet of Things: European Commission*. Brussels: *Information Society and Media* 189.

Taddeo, M. 2016. "Data philanthropy and the design of the infraethics for information societies." *Philosophical Transactions of the Royal Society*. DOI: 10.1098/rsta.2016.0113

Talmale, G., and Shrawankar, U. 2017. "Dynamic clustered hierarchical real time scheduling for IoT based human organ transplantation." *International Journal of Control Theory and Applications* 10, 239–249.

Terry, N. 2014. "Health privacy is difficult but not impossible in a post-hipaa data-driven world." *Chest* 146, 835–840.

Wachter, S., Mittelstadt, B., and Floridi, L. 2017. "Why a right to explanation of automated decision-making does not exist in the general data protection regulation." *International Data Privacy Law* 7, 76–99.

Wel, L., and Royakkers, L. 2004. "Ethical issues in web data mining." *Ethics and Information Technology* 6, 129–140.

World Health Organization (WHO) 2017. "World Health Organization (WHO) guidelines on hand hygiene in health care."

Yuan, B., and Herbert, J. 2011. "Web-based Real-time Remote Monitoring for Pervasive Healthcare." *Smart Environments to Enhance Health Care* 625–629.

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

16 Brain–Computer Interface

Abhishek Mukherjee, Madhurima Gupta, and Shampa Sen

CONTENTS

16.1 Introduction 265
16.2 Basic Brain Anatomy 266
16.2.1 Cortex 267
16.2.2 Functioning of the Brain 268
16.2.3 Signal Reception 269
16.3 EEG 270
16.3.1 Functioning 270
16.3.2 Data Acquisition from a Subject 271
16.3.3 The 10–20 System 272
16.3.4 Benefits 274
16.3.5 Drawback 275
16.3.6 Waveforms and Common EEG Patterns 275
16.4 Classification of EEG Signals 277
16.4.1 Support Vector Machine 277
16.4.2 KNN (K-Nearest Neighbors) 278
16.4.3 MLP-ANN (Multilayer Perceptron-Artificial Neural Networks) 279
16.4.4 Logistic Regression 279
16.5 Discussion 280
16.6 Applications 281
16.6.1 Aiding Invalids 282
16.6.2 IoT-Based Applications 282
16.7 Conclusion 284
References 284

16.1 INTRODUCTION

In all primates, the human brain is considered to be most advanced and complex. It has a fascinating structure with a variety of functions. It allows for remembering past events, processing all the present sensory inputs, and integrating the information, thinking, analyzing situations, creating memories, displaying emotions, and making estimations into the future. For a while now, scientists from various fields seek to understand the fine intricate workings of the brain and channel it in a way as to enhance the overall

productivity and efficiency of the task at hand. BCI or brain–computer interface is a buzzing technology in the world, which aims to do this. It can be understood as a mechanism of commanding various devices just by the use of signals generated by the brain, and also using those to comprehend the intention of the brain to perform a certain activity. The microelectrical signals generated by the neurons are assembled as the concept of EEG (electroencelagraphy), which can be translated as a pattern of data generated on by a brain stimulus (Nicolas-alonso and Gomez-gil 2012). A variety of algorithms can be used to understand and comprehend these patterns, and they can be used to build generic models to interpret signals of similar nature obtained in future readings. This can be further used to trigger computer functionalities as per requirement. A plethora of applications have the potential to find their roots with the advent of BCI and its subsidiary technologies. Starting from motor-related activities of the body to IoT applications and automation, everything can have a widespread use in the domain of BCI. This chapter discusses the concepts of inter-connection of the knowledge domain corresponding to the biological brain and the computer & computational related technical aspects. This will involve the conjugation of various aspects of brain anatomy, understanding of various perceived signals, algorithmic approaches, and finally some of its popular applications.

The primary focus is to understand the key elements of this technology and its workings so as to pave the way for researchers in this domain.

16.2 BASIC BRAIN ANATOMY

Before discussing the brain we need to understand the nervous system. The nervous system is essentially the electrical wiring of the body. It is made of trillions of nerve cells (neurons and glial cells). Neurons conduct impulses and cause sensations throughout the body, while glial cells take part in protective functions. In general terms, the nervous system takes in signals from your senses and processes the information to provide an appropriate response. On the whole this system is divided into (1) the PNS, or peripheral nervous system, which has nerves spanning and connecting all the body parts to the spinal cord; and (2) the CNS, or the central nervous system, consisting of the spinal cord and the brain.

The brain is the command center of the nervous system in all vertebrates and most invertebrates. It can be viewed as the biological computer; because just like an electronic computer it is involved in taking inputs from the environment in the form of sensory signals, storing the information (memory), processing and integrating it (learning), and producing an output (motor and hormonal signals) to exert control over the rest of the body. It occupies a space of up to 2% of the total body volume. The size of the brain differs slightly among the sexes and also from one individual to the other. However, this difference does not reflect on the cognitive ability of a person. The brain is constituted of mainly of three components: nerve cells, vascular system, and interstitial system, while it is divided into three functionally distinct but connected regions: cerebrum, brainstem, and cerebellum. The cerebrum is the largest part of the brain occupying 80% of the total brain volume. This region is involved in motor control, cognitive thinking, and behavior and processing of sensory inputs; the brainstem connects the cerebrum to the cerebellum and consists of three regions:

FIGURE 16.1 Schematic representation of the human brain.

mid brain, pons, and medulla (not shown in Figure 16.1), and continues after that into the spinal cord. The brainstem is the control center for processes such as sleep cycles, respiration rate, heart rate, temperature, blood pressure and volume, etc. while the small bulb-like structure of the cerebellum is involved in maintaining posture and balance of the body, as well as finely coordinating muscle movements. The basic structure can be understood from Figure 16.1; it is situated in the head region, and protected by the skull, or brain box, and the meninges.

In the course of this chapter our main focus shall be on the functioning of the cerebral cortex.

16.2.1 Cortex

As mentioned before, the largest part of the brain is the cerebrum. It is divided into two symmetrical hemispheres—right and left hemispheres—connected together by the calossum, as observed in Figure 16.1. The cortex is composed of grey and white matter, and these are nothing but bundles of neurons having special and distinctive orientations. The white matter is just the axons with myelin sheaths, while the grey matter is composed of everything else pertaining to the neurons. The cerebral cortex is the region of cortex associated with the cerebrum. It is composed of a greater number of layers compared to the cortex associated to the cerebellum. The cerebral cortex is a highly convoluted structure, composed of ridges known as gyri and fissures called sulci. Functional demarcation results in dividing each hemisphere into parietal, temporal, occipital, and frontal lobes. They all have distinct functions and yet are integrated which each other. The schematic structure and position of the lobes can be observed in Figure 16.2. As we can see, the frontal lobe is the largest of the four; it is our emotional control center and home to our personality. Thus, as expected, it is involved in a multitude of activities such as cognition (memory, problem solving, information processing, etc.), behavioral responses, and motor functioning and control. The occipital lobe is involved in visual processing while the temporal lobe takes care of sound and language. This lobe also harbors the amygdala and hippocampus, regions involved in emotions and memory, respectively. Parietal lobes function as an

FIGURE 16.2 Cortical lobes.

integration center for sensory inputs from different senses, and are also involved in spatial orientation, awareness, and sensation of self, as well as navigation. Between the cerebrum and brainstem lie the thalamus and hypothalamus. The hypothalamus acts as a relay center between the nervous system and the endocrine system. The thalamus communicates motor and sensory data to the cortex; also, it comprehends wakefulness and sleeping patterns.

16.2.2 Functioning of the Brain

Our brain is never asleep. It works 24/7. Some activities are voluntary like walking, writing, talking, etc. whereas others like breathing, heartbeat rate, balancing, etc. are involuntary. Both of these are monitored and controlled by different regions of the brain. However, in the course of this chapter we shall focus on the activities and functioning of the cortex.

Typically, the functioning domain of the brain can be divided into active (conscious state) and the passive (subconscious) stages. The conscious state is when we are awake and knowingly making decisions, thinking and involved in motor movements and assessing our environment, whereas the subconscious mind is a background phenomenon, in which we are always involved although we are not actively aware of it. The workings of the subconscious mind is not yet understood, however, it is accepted that all of our conscious decisions and our individual personalities and responses are greatly affected by it.

Further in this chapter we shall come across methods and technologies to study brain activity, and in turn the applications for their results. These studies are based on cortical activity.

The majority of our understanding of the functioning of the cortex and the task assignment to various regions comes from knock out studies and observing subjects with mental deformities and diseases (Dikmen, Machamer, and Temkin 2017). Take for instance the frontal cortex, which given its size and location is the most vulnerable to injury. An injury to the frontal lobe can cause drastic behavioral and personality changes in the patient, as well as it can permanently completely impair the fine-tuning of motor skills and may cause a loss of function in the limbs (Kraus et al. 2007).

The brain typically functions by means of electrical impulse transmission. The chemical signals (hormones) act in response to the internal environment; while electrical signals (transmitted by nerves) act based on both internal and external stimuli. For BCI applications, our focus is more on the electric stimuli, although understanding the fine control mechanisms hormones have over the motor responses and thought processes is also important. For this purpose, data from techniques such as PET, can be combined with EEG data to draw conclusions and identify unique patterns concerning the particular stimulus. As will be explained in the coming units, the EEG is applicable only for picking up voltage fluctuations from synchronized cortical pyramidal neurons of similar spatial orientation.

16.2.3 Signal Reception

So far, we have understood that the neuron is the functional unit of sensation and brain activity; in this context we shall introduce the pyramidal neurons, which are a special type of neurons found in all areas of the cortex, hippocampus, and the amygdala. Their orientation is perpendicular to the cortical surface; the cell body is oriented towards the grey matter while dendrites extend towards the white matter. The reason why their electrical fields are this strong and stable lies in their peculiar anatomy with multiple branches and unique orientation.

In the cortex, the synaptic activity of these neurons is dynamic in nature and keeps on changing in strength during the course of development and learning process. The dendrites conduct the input signal while the axons are involved in conducting output signals. Both dendrites and axons are highly branched in this case. The specialty of these cells, and one of the possible reasons for their abundance in the cortex, lies in the existence of dendritic domains with distinct excitabilities, modulation abilities, synaptic inputs, and plasticities. These distinctions allows the synapses in the dendritic tree to be dynamic while maintaining structural and functional connectivity. The synapse is the functional element related to changes in electric flux and the generation of signals. Synapses are present all over the body; however, in the cortex an extra complexity is added, where the synaptic strength is different in strength for different synapses (think about long-term and short-term memories).

The action potentials are generated when these cells fire, that is, when there is synapse formation. The synapse formation is initiated by an external stimuli, for example, a picture flickers in front of your eyes and then the neurons in the visual cortex are stimulated and start firing. However, how long the region will fire depends on factors such as the duration for which the image is shown, the attention span of the respondent, and the purpose for which the picture in question is being studied. It must be understood that in the cortex, when a particular type of stimuli predominates and is more focused than the others, the neurons pertaining to that stimuli fire more strongly and actively; however, that doesn't mean that the other regions are not active or are not firing simultaneously as well. The EEG, which shall be discussed later, also gets focused with a typical pattern when a particular stimulus is being forced upon the respondent. The encephalogram will detect more pulses over the region dedicated to that stimulus, yet small signals will be captured from other parts as well during the experiment.

FIGURE 16.3 Pyramidal neuron.

For a detailed account on the signal transduction in pyramidal cells, the reader is referred to Spruston (2008) (Figure 16.3).

16.3 EEG

16.3.1 Functioning

An EEG is a noninvasive method commonly used for recording the electrical activity of the brain. The EEG data is collected by making a subject wear an electrode cap. There are several electrodes placed at strategic locations around the scalp. Each electrode is individually connected to a separate wire, which in turn is connected to the monitor. The EEG records the spontaneous electrical field of the brain over a period of time. It does so by measuring the voltage fluctuation resulting from ionic current within the neurons. However, at this point it is critical to mention that the voltage fluctuation thus measured is not of a single neuron, rather it is a cumulative effect of activity of millions of neurons located in that area. Let's visualize it in this way, when an impulse is generated in one neuron, the disturbance causes changes in the resting potential and ion distribution. Tremors near one group of neurons causes tremors in the neighboring neurons, which in turn disturbs the next group of neighboring cells and so on. This creates a wave, similar to creating ripples on the surface of a body of water. And when the wave of ions reaches the scalp, the electrodes capture the disturbances and are displayed on the screen. The graph generated can either be voltage versus time or voltage versus frequency. In case of the voltage versus time graph the readings are represented in the form of pulses.

As mentioned above, a single wave is the result of the synchronous activity of neurons of similar spatial orientations. The firing patterns of the cortical neurons are exceptionally complex. Hence, to make sense and draw conclusions from the EEG data it is necessary to characterize the frequency ranges and the cognitive activity they characterize in turn. Based on individual factors, stimulus properties, and internal states, scientists classify the band frequencies into five ranges: delta (1–4 Hz), theta (4–8 Hz), alpha (8–12 Hz), beta (13–25 Hz), and gamma (>25 Hz).

16.3.2 Data Acquisition from a Subject

The experiment is designed keeping in mind the specific cortex region we need to examine, and the cognitive activity to be studied. For example, if we wish to differentiate between patients with epilepsy and normal brain graphs, we plan different awareness states; sleep, awake, while reading, meditation, etc. The final results for the two different types of subjects are then summarized in order to draw common patterns and compare and contrast them with each other.

The different frequency bands, elaborated in Section 16.3.6, are detected under different stimuli cases and cognitive awareness states. Hence, we see that based on our target stimuli and activity, the specific band frequencies can be concentrated on effectively to draw patterns. Some of the studies and their specific band frequencies are mentioned here:

Delta bands, which are present only during sleep state. More prominent in the right hemisphere, these are observed for studies concerning sleep and sleep disorders, alcoholism, and its effects on sleeping patterns. These frequencies play an important role in characterizing memory consolidation and rewiring activities of the brain, typical of the asleep state.

Theta bands are observed when the person is awake and involved in task solving. These are recorded from all over the cortex, hence pointing towards the fact that to solve a particular problem all the cortical regions are active simultaneously, groping for information which may help to solve the task at hand. This can be used to understand the task-learning pattern of the brain, and hence design the circuits to be used for implantation.

Alpha bands are generated in the posterior cortical regions and are involved in correlation activities between sensory inputs, motor function, and memory; in other words, it is the brain activity when one is in a meditating state, awake with eyes closed, and no other voluntary physical activity. The alpha activities are effectively monitored for meditation studies and understanding the attention levels of respondents during different tasks; in case of latter high alpha power is a feature of distraction that is to say, that in case of complex and attention involving tasks, high alpha power corresponds to active blocking of distractions by the subject (Bonnefond and Jensen 2013).

Beta bands are observed in both the posterior and frontal cortex. These are generated when the subject is involved in active, busy, anxious thinking, that is when the person is involved in planning and decision-making activities. These frequencies can also be associated to learning and perfecting the fine nuances of a task, motor movement coordination, and fine-tuning; for example, learning a particularly difficult dance step or sketching a detailed picture, etc.

Gamma bands, the exact regions of generation or function are not known; however, these high frequency bands are suspected to be involved in events of long-term and short-term memory formation, and segregation of the information gathered from the external environment, for grouping under these two memory pathways. One of the studies involving these are

microsaccade studies involving coupling of the EEG to eye movements, and thus drawing inferences for attention and visual information processing based on tracking subtle eye movements.

As pointed earlier, to collect EEG data, the subjects wear electrode caps. While generating the data and tuning the electrodes, care must be taken to generate sensitive and noise free data. Collecting clean data is of prime importance, as there is no algorithm that can "correct" the EEG readings once generated, that is, if your initial data is of bad quality and unclear you cannot improve upon it and then reanalyze it better. Each electrode should be shown to be active on the monitor and must be fine-tuned. The positioning of the electrodes around the scalp must be precise, and hence the size of the cap and the number of electrodes incorporated must be carefully chosen (Samy 2016).

The EEG electrodes are cleaned with 70% alcohol before and after use. The subject's scalp must be grease free and hair should be washed well before the experiment. There should not be any hair accessories. Generally, we have the option for both dry and wet electrode-based systems, however wet electrodes are preferred as the sharpness and sensitivity of the data is greater in this case. This type of electrodes forms a capacitor-like arrangement with the skin; here, the electrode comes in contact with the skin via a saline-based conductive solution (gel, cream, or paste). This capacitor analogy system attenuates any background low-frequency signals (Herron et al. 2017).

As discussed above, the respondent is subjected to various stimuli based on the subject matter of the study. Each test includes several reruns to gain confidence on the patterns detected.

However, the electrical data we gain needs to be assessed for miscellaneous nonrelated signals (contaminants) arising due to factors such as muscle movements (clenching of jaws, facial muscles, neck movement, etc.), eye movements, blinking, movement of the electrodes, or headset movements. These contaminants can mislead the interpretations, for example, if during a visual stimuli the respondent blinks too much the EEG may fail to record any activity pertaining to that stimuli in the visual cortex, giving rise to a blind spot. For each of the factors mentioned above, the researcher should try to minimize these variations as much as possible. As for the eye movements and blinking, the number of trials is kept high (at least hundred or above) and these movements are separately tracked, and their effect is attenuated using statistical tools from the raw EEG data (Singh, Singh, and Sandel 2014).

16.3.3 The 10–20 System

As previously mentioned, in order to extract the EEG data from the subject data acquisition should be done properly and the steps, guidelines, and rules should be strictly adhered to. It is understood that for the signals to be transmitted the cap studded with electrodes must be worn by the subject. Hence, it is crucial to have a fixed system, which will ensure there is a standardized way of getting the electrodes placed inside so that there is a uniform way of collecting data irrespective of the size, structure, or shape of the subject's head (Soutar 2013).

FIGURE 16.4 (a) Nasion, inion, and preauricular points. (b) Nasion to inion 10–20 division. (c) Left-right pre-auricular 10–20 division. (d) 10–20 circumferential division. (e) 10–20 parasagittal and transverse. (f) Electrode locations.

For this reason, there is a system popularly known as the *International 10–20 System,* which provides a standard arrangement of placing the electrodes on the cap. The procedure is explained in the following steps.

1. *Location*: First the head is marked by four primary locations known as nasion, inion, left preauricular point, and right preauricular point. These are the points, which are the primary locations for the 10–20 system. The nasion is the point just between the eyebrows or a little below that. The inion is the little bony protrusion at the center back of the head where the neck joins the skull. The left and right preauricular zones are the points just in front of the left and right ears, or near the regions of sideburns (Figure 16.4).

2. *Measurements*: The total length from the inion to the nasion is measured with a straight line. This line is divided into 10 and 20 percentages such that starting from the nasion tending towards the inion, 10% of length of the line is marked (Yang, Wang, and Member 2017). In a similar way, the next two intervals are marked at 20% increments each (Fz). Similarly starting from the inion tending towards the nasion the first interval is marked at 10%, and the next two intervals marked at 20% each (Pz). Following the same rule the line connecting the left and right preauricular points are also connected with starting as the 10% (T7 and T8), and the next two subsequent divisions at 20% intervals (C3 and C4). The place where both the lines (inion to nasion and left/right preauricular lines) intersect each other is the point marked as the center and is labeled as Cz. After that a circular path is taken assuming the circumference of the head running through the points of the first 10% incremental points of the inion, nasion, and preauricular points. The circumference drawn is also divided into 10% increments or intervals. Two parasagittal lines are virtually assumed to be segmented into intervals of 25%, which are running on both sides of the nasion–inion line. In the same way two transverse lines are also assumed segmented into 25% increments as well as running along both sides of the preauricular line (Soutar 2013).
3. *Labels*: The labeling convention is pretty simple. As we know there are four lobes of the brain. The frontal (F), temporal (T), parietal (P), and the occipital (O) lobe. The electrode placing points are coded with the initials of these lobes along with some number and letter conventions. It is such that the points on the left are always odd numbered whereas the ones on the right are always even numbered. The points nearer to the centerline have a lower number compared to the farther points. The three center points (C3, Cz, C4) of the auricular line are given the initial (C) defining the center. The three center points (Fz, Cz, Pz) of the inion–nasion line are given the values (z) defining zero. Using this convention the other points are also named as per their respective locations.

16.3.4 Benefits

With advancement in technology a number of technologies are now available to study brain activity, such as functional magnetic resonance imaging (fMRI), positron emission tomography (PET), magnetoencephlaography (MEG), to name a few. However EEG remains the primary choice in both clinical and research-based applications as hardware costs are significantly lower than for the other techniques, as well as its light weight, mobile, robust, and flexible nature (in contrast to techniques such as MRI, which need to be conducted in a shielded room and the respondent has to be strapped to the observation table or chair). Hence, EEG can be used in conveniently for real-time experimentation. It is a noninvasive technique with very high temporal resolution in the order of milliseconds. EEG is safe and nonhazardous and does not cause any sort of discomfort to the respondents; the experimentation is carried in a relatively open and comfortable environment, thus the subject does not feel claustrophobic. Also, the machine is silent and hence gives better results for auditory stimuli-based studies. Another very important real-time advantage is it can be used for respondents who are

incapable of making motor responses. Also, some of the event related potential (ERP) components can be studied when the subject is not actively attending to stimuli.

16.3.5 Drawback

Despite its many advantages, one of the major challenges faced with EEG is its poor spatial resolution. It cannot precisely tell which part of the brain is primarily active in response to a particular stimuli; such inferences can be drawn only on the basis of an hypothesis. Another major drawback is that EEG can be used only to study the cortex; for deeper structures and subcortical layers, assistance from other techniques such as fMRI is required. Additionally, EEG cannot detect the specific location of neurotransmitters or drugs, which is possible with PET.

Though the experimental setup is comparatively simple, it is time consuming, as sample size has to be kept very large and the number of trials for each stage of the experiment has to be kept sufficiently large in order to extract reliable information.

16.3.6 Waveforms and Common EEG Patterns

EEG is all about detecting patterns and signals, which are interpreted into waveforms from the EEG signals. For a clearer understanding of these concepts, it is important to have some fundamental knowledge of the waveforms and common EEG patterns, as well as what they imply.

The EEG signals are categorized as EEG rhythms, or frequencies, in technical words. These frequencies are nothing but oscillations of the microelectrical signals obtained from the postsynaptic potentials. There are a couple of frequency types distributed into bands, or in simple words, which can be understood as a range of frequencies.

- *Delta frequency band*: The frequencies in the delta band range between 1 and 4 Hz. This band of signals is primarily encountered during sleep. These frequencies are low and their amplitude is high. These are mainly used by sleep specialists and physicians to assess the depth and quality of sleep, and are also useful in detecting symptoms of sleep apnea. The right part of the brain shows a higher occurrence of such frequencies in comparison to its left counterpart.
- *Theta frequency band*: These frequencies range from 4 to 8 Hz. These frequencies are detected when the brain is performing deep memory-related activities, such as remembering or trying to recall some information, learning something, and storing information. Primarily it is associated with the frontal and prefrontal cores of the brain. Although these kinds of frequencies are found all over the cortices, its intensity has been observed mainly when the brain is processing difficult tasks, as well as mental and memory intensive operations.
- *Alpha frequency band*: These range from 8 Hz to 12 Hz. They are associated with relaxed wakefulness, when the brain is awake and during focused attentiveness. Also, they are sometimes noticeable when the mind and body are relaxed and awake, while the alpha wave diminishes when there is high mental or physical activity. They typically signify motor, sensory, and other related brain activities. They mainly originate from the posterior

cortical regions such as the parietal, occipital, and temporal zones. These frequencies seem to fire up when the brain is involved in activities requiring the senses to acquire attention to a particular thing.

- *Beta frequency band*: Ranging from 12 to 25 Hz, these frequencies depict subtle body motion such as micro limb movements, which require undivided focus, as well as excitement or being busy in some activity. These are ideally found in the posterior and frontal zones. They also show activity when other people initiate limb movements.
- *Gamma frequency band*: Frequencies ranging above 20 Hz are known as gamma bands. It still is a highly debated theory among the researchers regarding the location of the origin of the gamma signals. Some believe these are close to the theta frequencies, whereas others feel they originate as a residue of other neural functions.

The brainwaves generated are often a combination of all these kinds of waves where there can be hybrid frequencies, such as one of the frequencies (alpha, beta, gamma, and theta) superimposed on the other. These represent different cortical activities and can be assimilated as combinations of brain functions as well. However, over the years researchers have figured out some common EEG patterns, which are related to meaningful information.

The waves are acquired from the electrodes as a differential values of montages running through the adjacent electrodes. There are various montages of travelling patterns which are known as bipolar montages. A differential amplifier shows the differences between two identical looking waves and only portrays the difference between the two signals. There are sequences observed how their montages travel from brain cortex like anterior to posterior, temporal wise, etc. For example a sequence obtained from a difference of Fp2–F8, F8–T8, T8–P8, and P8–O2, which depicts a chain moving posteriorly picking up brain activities from the right frontal, temporal, parietal, and occipital regions (right temporal chain).

Let us now discuss some common EEG patterns useful in detecting common brain activities and used in medical sciences.

One of the most commonly studied EEG patterns is an eye blink. The two main parts of the eye are the retina and the cornea. The retina is situated at the back of the eye and is negatively charged, whereas the cornea is at the front containing a positive polarity. When an eye blink occurs, according to Bell's phenomenon, the eyeball rolls upwards to the forehead. Now, since regions like Fp1 and Fp2 are nearest to the eye, the positively charged cornea induces some charge on Fp1 and Fp2, which causes a significant deflection in the signals of the frontal polar electrodes.

Alpha rhythm is another popular signal, as discussed earlier, and is generally obtained during wakefulness with eyes closed.

A common pattern, which is largely used in medical sciences, is the focal epileptiform discharge. This occurs when there is a simultaneous unusual activity by the neurons in the cortex of the brain. These can originate in any region be it temporal, frontal, occipital, or parietal and range anywhere from anterior to posterior.

Another form of this is the generalized epileptiform discharges. These forms of frequencies are found in patients with epilepsy syndromes. As per some studies, when

they occur all the regions oscillate in a 3 Hz frequency generally in adults in a sudden time frame. Studying the results of EEG recordings in epileptic patients reveal that records vary from the time to time when attacks take place. If the recording is done within 24 hours of the occurrence there are 50 percent chances of finding abnormalities in the readings, If the recording done routinely or even non routinely in a particular epilepsy attack period there is a high chance of observing epileptiform discharges, whereas if the recording is done after a 24-hour time gap the chances of such discharges is reduced significantly.

16.4 CLASSIFICATION OF EEG SIGNALS

So far we have seen how the electroencephalogram signals are processed as to extract meaningful brainwave information from the subjects. We have also seen the supporting systems and a few standard extraction techniques to accomplish this. Common patterns can be systematically rendered so as to associate them with a specific brain–body activity. Now, we will discuss how the EEG signals are interpreted and the algorithms with which they are classified through the process of extraction (Poorna, Sai Baba, and Ramya 2016).

16.4.1 Support Vector Machine

The support vector machine algorithm is a classification algorithm initially proposed by Vapnik. This classification algorithm is just like any other classification algorithm which labels samples and categorizes them into classes. In the initial version of this algorithm it performed only linear classification. The linear classification process is such that, suppose there are two "attributes" of a sample using which the sample has to be "categorized" into either of the classes, there is a straight line that is termed as a hyperplane which is generated in such a way that it acts as a segregation border for the classification or so called categorization. There is a concept of maximum margin, which means that the distance of the nearest occurrence (technically called a "datapoint") of both the classes from the hyperplane should be maximum in order to get the most accurate classification and to be considered into one of the categories (Mehmood and Lee 2015) (Figure 16.5).

In Figure 16.5 we can see that there are two hyperplanes, the shaded one is "A" and the non-shaded one "B." Both hyperplanes A and B segregate the classes accurately but the maximum margin is left by the hyperplane "A," as it is evident that it has the maximum gap from the nearest elements of both classes compared to "B." Hence, it proves to be a better classifier. This algorithm focuses first on the accuracy by taking care of 2 aspects: (i) whether is able to classify all the data points accurately and (ii) margin maximization. In some cases there may be outliers of each class, which fall into other classes and are very difficult for linear SVM classification (Rahimi 2016). For that reason, Vapnik, Guyor, and Boser proposed a method to create a nonlinear classifier for SVM using the kernel trick for high dimensional planes. SVM is a training-based algorithm, which uses a training set for producing accurate results. In BCI, this is useful when there is a deviation of a signal received from the electrodes and distinguished by a differential amplifier (Ilyas et al. 2016).

FIGURE 16.5 SVM hyperplane.

16.4.2 KNN (K-Nearest Neighbors)

K-nearest neighbors is considered one of the simplest machine learning algorithms and is a nonparametric algorithm widely used for classification and regression. The algorithm falls under the supervised learning category and makes use of training data for classification of data. The working of the algorithm is such that a new sample can be classified into either classes based on the majority of the k-nearest attribute values. The k value is a number, which tells how many nearest data points of various attributes or features have to be considered so as to allot a class to the new sample. For the nearest neighbors, the Euclidian distance formula is used to evaluate the distances from the neighbors (Fayzrakhmanov and Bakunov 2016).

$$\text{Euclidian Distance}(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots (x_n - y_n)^2}$$

The above formula for Euclidian distance is used to calculate the distance between two points x and y. The nearest neighbors are weighted or ranked in ascending order, and the first k values of that order are selected for analysis. Further among those selected, the class which shows the majority is considered as the class label for the feature vector (Figure 16.6).

Example: [Class {Sun: Moon}]: For k value = 3 the shape triangle will be classified as a shape "sun," as among the nearest three neighbors the majority of up-votes are obtained by the shape "sun."

It is conceptually described as a lazy learner algorithm, which does not have any generalized model, whereas it uses the training values of the various instances in the training set to generate its own model. K-NN does not have a very high success rate for high-dimensional data, and hence it is considered efficient for low-dimensional scenarios (Manjusha and Harikumar 2016). The k value is appropriately chosen for cross validation and multiclass environments (Subasi and Erçelebi 2005). K-NN thereby produces a confusion matrix, which depicts the accuracy for a given data, while the accuracy of K-NN varies from subject to subject in BCI (Awan 2016; Ilyas et al. 2016).

FIGURE 16.6 K nearest neighbor.

16.4.3 MLP-ANN (Multilayer Perceptron-Artificial Neural Networks)

The multilayer perceptron is an algorithm that makes use of the artificial neural network (ANN) method of classification. ANN is a multidimensional and nonlinear model for classification, which is used in multiple disciplines for computational and classification purposes. The single layer perceptron may be used for linear classification, but often linear classification cannot accurately classify all the feature vectors of complex data sets. The EEG signals can be one such example. In those cases, the multilayer perceptron is required. In ANN, the working principle in single layer is such that each of the neurons transmit the signals to the output neuron, which computes the results based on the inputs of its preceding neurons. In the multilayer perceptron method there are multiple hidden layers present between the input and the output neurons. Inputs received by the neurons are assigned weights in each iteration of data flow, which are summed up while obtaining the bias as well. Now, the final output is measured against the desired output and the errors are recorded. Thereafter, for subsequent iteration of the algorithm the weights of the neurons are adjusted so as to reach the highly accurate and desired results. This feature of MLP-ANN is known as back-propagation (Figure 16.7).

It is used widely for training of neural networks and can be efficiently used for classification of motor signal images. The advantage lies in its flexibility and apparent multidimensionality (Ilyas et al. 2016).

16.4.4 Logistic Regression

Logistic regression helps in building models. These models make for a classification tool. The models are linear logistic regression models, which can be considered as

FIGURE 16.7 MLP-ANN.

learner algorithms giving a probabilistic output upon attribute selection for independent, as well as dependent variables, by evaluating the relationship between them.

16.5 DISCUSSION

Before we start the comparison and analysis of these algorithms we need to understand how these algorithms capture the data and initiate a procedure for classification. In the previous sections, we have seen how the brain waves generated are captured by the electrodes during the data acquisition process. The waves generated are passed on from the electrodes to the computer. Any raw data received from the subject is highly contaminated in the sense that the data acquired may pick-up signals from other external sources (other than the microvoltages generated from the combined postsynapses of the different zones of the lobes of the brain). These unwanted signals are known as artifacts. Artifacts are signals from electrodes other than the electrode we are focusing on to receive data for a specific functionality (Šťastný, Sovka, and Stančák 2003). They can also be other signals such as from electronic equipment in the environment, ambient noise, etc. After the raw data is received it is cleaned and preprocessed with methods such as manual cleansing of artifacts such as blinks of the eye, movements of the eye, muscle twitches, and bad channels as well. Also, methods like independent component analysis can be used to clean the data for each subject. Now, when the data is cleaned the uncontaminated EEG signals are prominent. Each channel connected to the array of electrodes supplies continuous analogous data where the EEG receptors take readings of that data at regular intervals. The efficiency of the receptors is measured by the number of samples collected per unit time, also known as *resolution*. The process of classification involves categorization of deviation of the frequencies from their usual readings, as received from the array of electrodes into two possible cases (i) Considerable deviation has occurred and (ii) Deviation has not occurred. The deviation may occur when a stimulus is given to the subject. These algorithms tend to build a learning model, which trains how a signal is

classified as a "deviation" or a "nondeviation" with respect to a given stimulus. While the model is being built, the various combinations of the patterns from the various channels are considered in consolidation for a particular stimulus. This then becomes a reference pattern associated to a stimulus. Once the algorithms are done building their classification models they can predict the actions for new samples to be classified so as to conclude if they mean any deviation, which in turn may refer to some activity based upon the previous pattern of signals. There is a considerable amount of research in this field to determine how data is classified and the various steps involved in it in granular steps (Samanta 2017).

These algorithms perform differently for different scenarios when classification of EEG signals is involved. As discussed by M.Z. Ilyas et al. (Ilyas et al. 2016), information obtained from the "Berlin BCI group" where they had collected BCI data for motor imagery related to left and right limb movements upon controlled visual stimulus. It was obtained from various subjects. A stimulus provided resulted in some motor imagery tasks which were recorded. Conversion of those signals from time domain to a frequency domain was performed using Fourier transformations. It was further converted into an attribute relation file format using WEKA, which is capable of performing classification algorithms.

The results obtained from each of them indicate that no particular algorithm can be considered as the "perfect one" for a particular dataset of EEG specific to a subject. Each of them show varying results for different subjects. Logistic regression and support vector machine performed well if looked at from a broad perspective posing as highly efficient classifiers, on the contrary k-Nearest Neighbor and MLP- Artificial Neural network performed better for some instances where the feature vector was highly correlated but also deteriorated for other instances. The performance of k-NN seems to be poor statistically, as it apparently fails to work efficiently if the data contains high dimensionality (Zhang et al. 2015).

In Section 16.6, we will see how EEG and BCI can be used for various application-oriented utilities and practical usage in a variety of fields.

16.6 APPLICATIONS

Humans love their machines and a technician is as good as his tools. The software developers have claimed the virtual world with their algorithms; the genetic engineers' job is eased with gene sequencing tools and CRISPER (clustered regularly interspaced short palindromic repeats). Doctors and clinicians have too many electronic helpers at their disposal to be named here.

These days, science has advanced enough to enable us to print functional organs (3D bioprinting); develop computers able to detect and record the brain activities on paper (EEG and related technologies); create robots that can do our manual work (robotic arms and process belts); and the list goes on (Hasan et al. 2016).

However, all these remarkable discoveries and inventions are employed separately, hence we are now faced with the opportunity to build a platform to integrate it all in one. AI (artificial intelligence) is one such project. Yet, it merely seeks to mimic the workings of the mind and does not connect with it. Hence, human intelligence still reigns supreme outside of narrowly defined tasks. To go beyond these limitations, it

has been sought to integrate the human brain and its synapses with the Internet in the ever-expanding domain of IoT. In separate endeavors, Bryan Johnson, founder and CEO of Kernel, Elon Musk, and the US military's research arm, DARPA, among others have begun to look for ways to link mind and machine.

This quest has a number of applications, as discussed below.

16.6.1 Aiding Invalids

Diseases such as lower body paralysis, muscular dystrophy, Parkinson's, limb amputation, etc. can render the patient incapable of making appropriate motor responses (Zafar et al. 2017). A person with severe motor disabilities can feel helpless and go into a depression as they become imprisoned in their own bodies. As the disease progresses, the person loses the ability to interact with their environment, or lose the ability to communicate—which is terrifying. Take for instance ALS (amyotrophic lateral sclerosis), a condition in which the neuromuscular connections progressively degrade to the extent that the only way to stay alive is to be connected to a life support systems. However, the fact is that the patient, even though physically impaired beyond resurrection, is not brain dead and can still hope for some normalcy, if a technology is introduced, which can work on brain synapses rather than neuromuscular connections. This was demonstrated by Kübler et al. where they used EEG based BCIs to help four ALS patients (Kübler et al. 2005).

The basic idea is that the brain activity can be transformed into a digital form, which can then act as an input command for the computer. The BCI tools can be used in a multitude of areas, such as education, gaming, industry, astrophysics, medical fields, etc. (Hadi, Sholahuddin, and Rahmawati 2016).

It has been confirmed by neuroscientists that, even when the limb is no longer functional due to some accidental damage, the area of the brain associated with control of that organ still fires signals. In extreme cases, we have cases of phantom limbs (Nikolajsen and Jensen 2001); these findings show that there is an innate neural circuit associated with a particular motor response. Hence, to control the robot to perform the corresponding physical task, it can be made to recognize the specific firing pattern with the help of EEG readings and data from healthy individuals. This way it forms an algorithm for the specific command (Obeidat, Campbell, and Kong 2017).

The machine can be trained to perform a specific task by tuning in to the thought process (in other words conscious decision to do the work) associated with the task (Arjestan and Vali 2016). This means that BCIs can not only be used as a means of communication but also to develop implantable chips with clean and safe surgical procedures. Also remote controls can be designed which can be operated with least amount of muscular input and be employed to control an external exoskeleton or a robotic limb. As had been reviewed by Carmen in her 2012 review paper under the section for IOTs for physical disabilities (Carmen 2012).

16.6.2 IoT-Based Applications

IoT involves connecting various sensors, receivers, transmitters connected to computing devices so as to process, analyze, visualize, and provide necessary responses to input information.

Brainwaves controlling physical and electronic devices was still science fiction in the recent past. But with the evolution of various BCI techniques and their enhancements using brainwaves to control various devices, it may soon be a reality finding uses in medical and home-automation fields.

IoT involves three basic elements—the sensors, the processors, and the actuators. These three devices work in unison to fulfill the activities of IoT-based environments. The IoT has several layers in its interface. The first is the perception layer, which involves the sensors or the actors. Next, is the network layer or the layer involving the connectivity. Then comes the middle layer which acts as a facilitator for various types of different devices to communicate without any hindrances. The top two layers consist of the application layer and the business layer, which consist of the user interaction and deep analysis of data involving machine learning for data analytics and inferring meaningful information (Mathe and Spyrou 2016).

One of the popular devices in the market is the NeuroSky Mindware Mobile device. These are used as we use normal headsets and it mainly perceives brainwaves generated at the center of the forehead. It also has an auxiliary sensor near the ear. This product has its own e-sense algorithm, which can be used to translate the brainwaves into values such as calm or attentive depending upon the frequency readings of alpha, beta, gamma, theta, or delta frequencies (Jiang et al. 2016). It can infer important functions such as eye closing/opening, etc. These can be used for a variety of purposes (Arora, Choudhury, and Kumar 2016; Arora et al. 2016).

One of the best uses of BCI in IoT is home automation and control. Although it is still at a developmental stage, there are a variety of functionalities which can be performed. For example, the Mindwave device can be linked to a computer, which can be in turn linked to the controller unit. A popular controller unit is the Arduino Uno, which can be coupled with a Bluetooth module. Upon receiving the signals the Bluetooth transceiver passes on the signals to the Arduino controller to enable real automation. The Arduino controller is linked with various electronic devices or appliances (Nanditha and Persya 2017).

It can be used to turn off/on home lighting, or help prepare a mood lighting setting for each individual user. It can be used in setting the air-conditioning temperature of the home. It is found that beta frequencies become activated when the subject is brought in contact with high levels of sound or light stimulus. These can be used to set toggle lighting or the music player in the house. Since the alpha waves depict concentration, such sensors can support applications, which performs analytics using various algorithms that may be used to track effective meditation patterns as well as sleep disorders. This brings us to medical utilities. In the medical field, it can be used to alert paramedics in case a patient undergoes some kind of epileptic attack, or to monitor postepileptic symptoms of a patient. Alongside aiding invalids with sensory motor utilities, it has been observed that such frequencies can used a good deal in virtual reality. Virtual reality being the next big thing, this has huge applications in multimedia, interactive media, andgaming control.

There may be a plethora of applications developed further, as and when the BCI and its supporting technologies are enhanced and become even more advanced.

16.7 CONCLUSION

The brain and its interface with the computer is a field with immense potential. We aimed to provide the various nuances of the concept of Brain-Computer-Interface using a holistic approach. In the course of this chapter we explored briefly the brain anatomy and how its functional aspects are exploited and integrated with the machines. The EEG is our most powerful and vital tool for achieving this integration. As we have seen, electrical synapse is generated by the neurons and then transmitted to one another, thus creating a field of electrical discharge succeeding the postsynaptic activity. These are then detected and recorded by the EEG. The international 10–20 system is the most commonly followed system for collection of this data. EEG will always be the first choice for the applications of BCI and IoT because of its ease of application and robustness. However, it is still not perfect, nor an all-around approach, as there are many gaps. Nevertheless, scientists are now coming up with supporting technologies, algorithms, and sister techniques (FMRI, PET etc.), which can attenuate these gaps to a great extent and open more arenas for us to explore. We hope there may be a day when we can finally understand the incredible functioning of the human brain and tap into its many potentials that still hidden from us.

REFERENCES

Arjestan, M. A., and M. Vali. 2016. "*Brain Computer Interface Design and Implementation to Identify Overt and Covert Speech*." no. November: 23–25.

Arora, B., T. Choudhury, and P. Kumar. 2016. "*An Intelligent Way to Play Music By Brain Activity Using Brain Computer Interface*." no. October: 223–28.

Arora, B., T. Choudhury, S. Sabitha, and P. Kumar. 2016. *"An Exhaustive and Comparative Analysis of Brain Computer Interface Algorithms*." no. October: 202–7.

Awan, U. I. 2016. "*Effective Classification of EEG Signals Using K-Nearest Neighbor Algorithm*." 120–24. https://doi.org/10.1109/FIT.2016.28.

Bonnefond, M. and O. Jensen. 2013. "The Role of Gamma and Alpha Oscillations for Blocking out Distraction," 20–22.

Carmen, M. 2012. "Journal of Network and Computer Applications An Overview of the Internet of Things for People with Disabilities." *Journal of Network and Computer Applications* 35 (2). Elsevie: 584–96. https://doi.org/10.1016/j.jnca.2011.10.015.

Dikmen, S., J. Machamer, and N. Temkin. 2017. "Mild Traumatic Brain Injury: Longitudinal Study of Cognition, Functional Status, and Post-Traumatic Symptoms." *Journal of Neurotrauma* 34 (8): 1524–30. https://doi.org/10.1089/neu.2016.4618.

Fayzrakhmanov, R. A., and R. R. Bakunov. 2016. "*The Reduction of Learning Sample in Information- Measuring and Control Systems Based on Brain- Computer Interface Technology*." 0–3.

Hadi, S., A. Sholahuddin, and L. Rahmawati. 2016. "*The Design and Preliminary Implementation of Low-Cost Brain-Computer Interface for Enable Moving of Rolling Robot*." no. Icic: 8–12.

Hasan, K., S. M. Hasnat Ullah, S. S. Gupta, and M. Ahmad. 2016. "*Drowsiness Detection for the Perfection of Brain Computer Interface Using Viola-Jones Algorithm*."

Herron, J. A., M. C. Thompson, T. Brown, H. J. Chizeck, J. G. Ojemann, and A. L. Ko. 2017. "Cortical Brain Computer Interface for Closed-Loop Deep Brain Stimulation." 4320 (c): 1–7. https://doi.org/10.1109/TNSRE.2017.2705661.

Ilyas, M. Z., P. Saad, M. I. Ahmad, and A. R. I. Ghani. 2016. "Classification of EEG Signals for Brain-Computer Interface Applications: Performance Comparison." *2016 International Conference on Robotics, Automation and Sciences (ICORAS)*, 1–4. https://doi.org/10.1109/ICORAS.2016.7872610.

Jiang, M., T. N. Gia, A. Anzanpour, A. M. Rahmani, T. Westerlund, S. Salantera, P. Liljeberg, and H. Tenhunen. 2016. "IoT-Based Remote Facial Expression Monitoring System with sEMG Signal." *SAS 2016 - Sensors Applications Symposium, Proceedings*, 211–16. https://doi.org/10.1109/SAS.2016.7479847.

Kraus, M. F., T. Susmaras, B. P. Caughlin, C. J. Walker, J. A. Sweeney, and D. M. Little. 2007. "White Matter Integrity and Cognition in Chronic Traumatic Brain Injury: A Diffusion Tensor Imaging Study." *Brain* 130 (10): 2508–19. https://doi.org/10.1093/brain/awm216.

Kübler, A., F. Nijboer, J. Mellinger, T. M. Vaughan, H. Pawelzik, G. Schalk, D. J. McFarland, N. Birbaumer, and J. R. Wolpaw. 2005. "Patients with ALS Can Use Sensorimotor Rhythms to Operate a Brain-Computer Interface." *Neurology* 64 (10): 1775–77. https://doi.org/10.1212/01.WNL.0000158616.43002.6D.

Manjusha, M., and R. Harikumar. 2016. "Performance Analysis of KNN Classifier and K-Means Clustering for Robust Classification of Epilepsy from EEG Signals." *Proceedings of the 2016 IEEE International Conference on Wireless Communications, Signal Processing and Networking, WiSPNET 2016*, 2412–16. https://doi.org/10.1109/WiSPNET.2016.7566575.

Mathe, E., and E. Spyrou. 2016. "Connecting a Consumer Brain-Computer Interface to an Internet-of-Things Ecosystem." *Proceedings of the 9th ACM International Conference on PErvasive Technologies Related to Assistive Environments - PETRA '16*, 1–2. https://doi.org/10.1145/2910674.2935844.

Mehmood, R. M., and H. J. Lee. 2015. "Emotion Classification of EEG Brain Signal Using SVM and KNN." *International Journal of Bio-Science and Bio-Technology* 7: 23–32. https://doi.org/10.1109/ICMEW.2015.7169786.

Nanditha, M., and S. C. Persya A. 2017. "EEG-Based Brain Controlled Robo and Home Appliances." *International Journal of Engineering Trends and Technology (IJETT)* 47 (3): 161–69.

Nicolas-alonso, L. F., and J. Gomez-gil. 2012. "*Brain Computer Interfaces, a Review*." 1211–79. https://doi.org/10.3390/s120201211.

Nikolajsen, L., and T. S. Jensen. 2001. "Phantom Limb Pain." *British Journal of Anaesthesia* 87 (1): 107–16.

Obeidat, Q., T. Campbell, and J. Kong. 2017. "Spelling with a Small Mobile Brain-Computer Interface in a Moving Wheelchair." 4320 (c): 1–11. https://doi.org/10.1109/TNSRE.2017.2700025.

Poorna, S. S., P. M. V. D. Sai Baba, and L. Ramya. 2016. "*Classification of EEG Based Control Using ANN and KNN- A Comparison*." 0–5.

Rahimi, M. 2016. "*Ensemble Methods Combination for Motor Imagery Tasks in Brain Computer Interface*." no. November: 23–25.

Samanta, B. 2017. "*Empirical Mode Decomposition of EEG Signals for Brain Computer Interface*." 4–9.

Samy, A. A. 2016. "*Extracting and Discriminating Selective Brain Signals in Non-Invasive Manner and Using Them for Controlling a Device : A Cost-Efficient Approach to Brain Computer Interface (BCI)*."

Singh, M., M. Singh, and A. Sandel. 2014. "Data Acquisition Technique for EEG Based Emotion Classification." *IJITKM* 7 (2): 133–42.

Soutar, R. 2013. "An Introductory Perspective on the Emerging Application of qEEG in Neurofeedback." *Clinical Neurotherapy: Application of Techniques for Treatment* 19–54. https://doi.org/10.1016/B978-0-12-396988-0.00002-7.

Spruston, N. 2008. "Pyramidal Neurons: Dendritic Structure and Synaptic Integration." *Nature Reviews Neuroscience* 9 (3): 206–21. https://doi.org/10.1038/nrn2286.

Šťastný, J., P. Sovka, and A. Stančák. 2003. "EEG Signal Classification: Introduction to the Problem." *Radioengineering* 12 (3): 51–55. https://doi.org/10.1109/IEMBS.2001.1020628.

Subasi, A., and E. Erçelebi. 2005. "Classification of EEG Signals Using Neural Network and Logistic Regression." *Computer Methods and Programs in Biomedicine* 78 (2): 87–99. https://doi.org/10.1016/j.cmpb.2004.10.009.

Yang, Y., H. Wang, and S. Member. 2017. "*A Brain-Computer Interface Based on a Few-Channel EEG-fNIRS Bimodal System*." 5.

Zafar, M. B., K. A. Shah, H. A. Malik, and A. Principles. 2017. "*Prospects of Sustainable ADHD Treatment Through Brain-Computer Interface Systems*." 1–6.

Zhang, R., P. Xu, R. Chen, T. Ma, X. Lv, F. Li, P. Li, T. Liu, and D. Yao. 2015. "An Adaptive Motion-Onset VEP-Based Brain-Computer Interface." 604 (c). https://doi.org/10.1109/TAMD.2015.2426176.

17 IoT-Based Wearable Medical Devices

Avitaj Mitra, Abanish Roy, Harshit Nanda, Riddhi Srivastava, and Gayathri M.

CONTENTS

17.1 Introduction 287
17.1.1 The History of Medical Devices 288
17.1.2 The Advent of IoT 288
17.1.3 The Age of WMDs 289
17.2 Mechanism of WMDs 289
17.3 Detection, Diagnosis, and Treatment 290
17.3.1 Hypertension 290
17.3.2 Epilepsy 291
17.3.3 Cancer 292
17.3.4 Migraines 293
17.4 Computational Testing Methods 294
17.4.1 Epilepsy 294
17.4.1.1 Computerized Cognitive Testing in Epilepsy (CCTE) 294
17.4.1.2 Cognitive Drug Research Computerized Assessment System (CDR) 294
17.4.1.3 Cognitive Neurophysiological Test (CNT) 294
17.4.1.4 FePsy 295
17.4.1.5 Neurocog FX 295
17.4.2 Cancer 295
17.5 Conclusion 296
References 296

17.1 INTRODUCTION

In the current day and age, with increasing usage of technology, the majority of people in urban areas work white-collar jobs. These jobs, although well paid, involve working in front of a screen for extended periods of time. The lifestyle of these people is becoming increasingly sedentary. This, combined with a propensity for consuming large quantities of junk food, leaves them susceptible to various diseases like diabetes, hypertension, cholesterol, heart disorders, and many more.

The Internet of things (IoT) is a concept, which in simple terms is the interconnectivity of sensors, actuators, and microcontrollers, etc. along with smart devices and Internet connectivity to exchange and modify large sets of data. A subset

of IoT is wearable medical devices (WMDs). As the name suggests, these are small contraptions like a band or a small microchip, which can be worn on any part of the body and can help to monitor certain parameters and send warning signals if necessary. The basic components of a WMD are the chip/band, a sensor, a detector, a Bluetooth system for data transmission, and an alarm detector system. Since the simplest method to treat diseases is to detect the symptoms as early as possible, WMD's can prove to be hugely beneficial in this aspect.

Apart from the usual problems faced in the initial concept design formulation, obtaining funds for components, etc. possibly the biggest challenge preventing the large-scale commercialization of medical devices is the time taken for FDA approval. A probable reason for this is that these devices target the core physiological functions of the body. Any kind of operational failure can, in the worst possible scenario, lead to the death of the patient.

There is no doubt about the fact that WMDs, though currently at a nascent stage in their development, is the technology for the future. With the advent of smartphone technology and with increasing and widespread access to Internet services for the common people, health care professionals will come to depend upon wearable devices as a means to improve health care and reduce patient mortality rates.

17.1.1 The History of Medical Devices

WMDs are a classification of devices generally attached to the surface of the skin of an individual. They usually assist in monitoring certain parameters such as heartbeat, blood pressure, and oxygenation levels. An example of this is a heartbeat monitor or an SpO2 sensor. The other major function of a WMD is to improve the efficiency of any kind of cognitive function, which has degenerated in the human body. An example of this kind of medical device is a hearing aid/implant. The earliest examples of medical devices can be considered to be stethoscopes, sphygmomanometers, earlier forms of CT and MRI scanners, etc. However, the shift in medical device manufacturing received a big boost with the advent of the IC, semiconductors, and BioMEMS or biological microelectromechanical systems technology in the 1970s and 1980s. This led to devices small in size, increasing their portability and overall utility. The major impact has been on cardiovascular devices with a number of heart valves and stents coming up in the market. All this has increased not only the lifespan of patients but also provided comfort and improved their daily lives.

17.1.2 The Advent of IoT

The IoT is a global system facilitating the exchange of information in a secure manner. The primary architecture is based on data communication, facilitated by the use of RFID (Radio Frequency Identification) tags (Weber and Weber 2010). In the current industrial scenario, the most popular method is that of an EPC (Electronic Product Code). In simple terms, this means that a unique code is given to any object with an RFID tag, which helps to track the object remotely, without causing any discomfort to the individual carrying the object. The entire data is not saved within the RFID

tag, but the information is made available to servers on the Internet by cross-linking with the help of an ONS (Object Naming System).

The ONS is a centralized system linking both metadata and services and is operated by a private company, VeriSign. The ONS architecture is designed to be multifunctional and can be used for computation, recognition and identification of objects, and retrieval of datasets from the Internet.

The ONS is based on the more commonly used and well-known domain name system (DNS). In order to retrieve information about an object, the EPC of the object should be converted into a form the DNS can understand. This is done by using the usual format of a domain name ("dot," delimited, "domain name from left to right"). The EPC is usually encoded within the domain name and is used within the existing architecture of the DNS. ONS can be considered to be a subset of the DNS.

Thus, using IoT, the data collected by the WMDs can be transmitted to a storage device and can be accessed by a medical practitioner in real time. This has enabled the development of new age WMDs, which are aimed at improving health care delivery, with minimal effect on patient comfort and mobility.

17.1.3 The Age of WMDs

The recent advances in wireless sensor technology and information technology have resulted in an unprecedented number of technological advancements in the field of WMDs based on IoT (IoT). The use of WMDs for both monitoring and diagnostic purposes is rapidly becoming popular. With an increasing number of people owning smartphones enabled with Bluetooth, global positioning system (GPS), and Internet service it has become much easier to link WMDs, which measure physiological parameters for a software. This has opened up new avenues in medical devices currently being explored.

Round-the-clock monitoring of certain physiological parameters are of paramount importance in the detection and control of cardiovascular diseases. With the development of advanced wireless sensors, we no longer need to confine a patient in a hospital bed to monitor his/her vitals; it can be done remotely as the patients carry on with their daily lives. The power of remotely monitoring the vitals of a patient can go a long way in exponentially increasing the reach of proper health care facilities to semirural and rural locations. With the advent of IoT, these physiological data can be stored and transferred to the desktop of a physician for monitoring in real time.

17.2 MECHANISM OF WMDs

The heart of any WMD is data. In this case, data refers to patient history of the vital signs such as core body temperature, heart rate, and blood pressure. The functioning of any WMD can be broken down into the following: collection of raw data, transmitting the raw data using Ethernet, Bluetooth, or wireless technology, uploading the data into a secure server linked to a research facility and/or a hospital, postprocessing of the data using mathematical algorithms and statistical analysis, and finally displaying and storing the processed data using encrypted files, which can then be sent to individual patients.

The detection, diagnosis, and treatment of diseases using WMDs depends upon various factors such as the type of bio signal extracted, previous patient history, any genetic aspects which may interfere with electronic signals from the wearable device, the age of the person, work history (for a working professional, round-the-clock remote monitoring may not be a feasible option), and any kind of lifestyle changes, which need to be incorporated, etc. In short, it can be said there is no perfect wearable device, or no ideal form of treatment existing for an individual, and the diagnosis and treatment need to be carried out on a case-by-case basis.

17.3 DETECTION, DIAGNOSIS, AND TREATMENT

17.3.1 Hypertension

With the rise in the percentage of the population suffering from obesity, the number of people suffering from hypertension is on the rise. High blood pressure is often asymptomatic, and thus hypertension goes on unnoticed in many patients before it is too late. Recent studies have shown that blood pressure variability, or BPV, is an important factor for determining the cardiovascular health of an individual. Traditionally, blood pressure was measured in a clinic by a trained technician, nurse, or a doctor. In that scenario, constant monitoring of blood pressure was not possible. To tackle this problem, medical researchers and engineers have come up with a few types of WMD (Yilmaz et al. 2010).

The first type of medical device designed for home use by nontrained personnel was based on the oscillometric method. It consists of an armband, much like the one used in clinics attached to a monitoring device with a display. This device could measure the blood pressure with respect to the brachial artery. A similar type of device was later designed able to measure the blood pressure using the radial artery (Teng et al. 2008).

This new device used a wrist cuff instead of an armband. This enabled the development of smart wristwatches for blood pressure monitoring and display. The smartwatch could be connected to a smartphone to store and relay the data as per needed. The wrist cuff or smartwatch system made it easier to use the device outdoors with ease. It reduced the bulkiness of the previous devices (Yilmaz et al. 2010).

Although these devices enabled people to monitor their blood pressure without the aid of a doctor or a trained clinician, there were certain drawbacks. One of the major drawbacks was the sensitivity and data reliability. Second, these methods do not allow for a continuous monitoring of blood pressure; at best these devices could allow the monitoring of blood pressure at a regular interval of around 15–30 minutes. These methods can also lead to wrist or arm pain, sleep disturbance, and can even cause skin allergy (Bobade and Walli 2015).

Vasotrac (Medwave Inc., Arden Hills, MN) is a popular wristwatch-based ambulatory noninvasive blood pressure sensor. This medical device consists of a circular sensor placed on top of the radial artery (marked using a disposable adhesive tape) and a display unit. For reliable measurement, an external pressure needs to be applied. The data is collected from the sensor, processed by a control unit, and then displayed on the screen. Vasotrac is a good and reliable device for periodic measurement of blood pressure, but not for continuous blood pressure monitoring (Findlay et al. 2006).

17.3.2 Epilepsy

Epilepsy is a functional pathophysiology present in the cerebrum part of the brain, which arises in almost all mammalian species. It's a medical condition classified with a broader range of different types of seizures and activation from one individual to another. Different types of brain seizures in epilepsy lead to the development of brain tumors.

In epilepsy, people face different types of brain seizures. This marks the development of brain tumors. A person suffering with epilepsy has higher chances of suffering from a depression. This is due to higher exposure to chronic stress (Duyn et al. 2007). During sleep nocturnal seizures occur, which worsen the condition of the patient. These seizures mostly occur at specific sleep stages or during sleep instability (Bernhardt et al. 2015). The recent advance in the pathogenesis of epilepsy is the involvement of genetic testing. Next generation sequencing has proven to be effective for revealing epilepsy causing gene mutations (Witt et al. 2013).

Epilepsy is initially treated with medications where approximately 70% of patients get complete seizure control (French 2007). However, this is not possible for patients who have poor seizure control, and for them several other methods are used such as medical management, surgery, vagus nerve stimulation (VNS), ketogenic diets, and complementary therapies.

To detect brain abnormalities, several techniques are used such as electroencephalogram (EEG), computerized tomography (CT) scan (Duyn et al. 2007), magnetic resonance imaging (MRI) (Jewells and Shin 2014), functional MRI (Fang et al. 2017), positron emission tomography (PET) (You et al. 2012), and single photon emission computerized tomography (SPECT) (Błaszczyk and Czuczwar 2016).

There are various WMDs being developed for the timely detection of epileptic seizures. One of the latest devices is SeizeIT, which is being developed in KU Leuven (Vandecasteele et al. 2017). It is based on designing a WMD that detects the electrocardiography (ECG) and photoplethysmography (PPG) in real time. The wearable ECG device proved to be as efficient as the wired hospital ECG systems, and thus allows for greater patient mobility and comfort. The sensitivity of the wearable ECG device was 70%, whereas that of the hospital system was 57%, and it had a slightly greater false alarm rate (2.11 per hour as compared to 1.92 per hour rate of the hospital system) making it an ideal choice for people suffering from epilepsy.

The Brain Sentinel is an interventional device for the detection of tonic clonic seizures. It is based on measurement of electromyography (EMG) signals from the concerned subject and then analyzing and transmitting the data to look for identifiable patterns. The subjects were monitored for seizure-based activity for a period of 4.5 years and the test results are currently being analyzed for any favorable outcomes. However, one disadvantage is that the study excludes pregnant females from participation.

The Embrace Watch is an IoT-based device providing regular monitoring of vital signs for epilepsy-afflicted patients. It is considered to be a therapeutic rather than a diagnostic device, which transmits the dataset to a smartphone using Bluetooth technology. The data is then uploaded to Empatica servers where further analysis and processing is performed.

The neurospace responsive neurostimulator (RNS) is a recently FDA-approved device for epilepsy treatment. The principle behind this device is monitoring the brain activity and thus delivering an electrical stimulus when there is a detection of abnormal brain activity in different regions of brain (Thomas and Jobst 2015).

The RNS system is approved in the United States as a medical therapy for treatment of partial onset seizures of epilepsy. The different parts of this system include a stimulator, implanted leads, and wireless programming wands connected to a complex computer hardware and software system. The stimulator works on a design where wires connected to electrodes are implanted in the hippocampus area of the brain. The machine targets the resistant partial seizure onset zone.

The neurotransmitter monitors the ECoG (electrocorticography) activity and utilizes various methods for detection of abnormal brain activity. Measurement of changes in electrical activity and frequencies is done with this machine. The four proposed theories for mechanism of action includes:

- Depolarization blockage, which refers to changes in voltage-gated channels leading to process of inhibition in excitability (Beurrier et al. 2001).
- Synaptic inhibition refers to depolarization effects of distal axon.
- Synaptic depression, which causes the decrease in release of neurotransmitters (McIntyre et al. 2004).
- Electrical stimulation by modulating the action of pathologic networks (Durand 1986).

The RNS system does not include the allowance for external recordings and includes only limited sampling for selective ECoG. A comprehensive file of the data obtained through this system is very expensive and is difficult to track the efficacy of seizures. Further research and experiments are currently carried out for provision of better understanding and effects of this system on the human body.

17.3.3 Cancer

Cancer is among the deadliest diseases in the world, for which there is no comprehensive cure till now. Early detection of cancer can enable its treatment is most cases. However, a large percentage of the cancer cases are detected in the late stages, where nothing much can be done except to ease the pain and accept the inevitable outcome. Countless research groups across the globe are working on developing methods to enable early detection of cancers to give patients a fighting chance, and to hopefully defeat this deadly disease.

The advent of IoT and WMDs has brought about great improvements in the cancer care delivery system. Real-time data collection and analysis has undergone massive improvements in the last few years (Sledge et al. 2013). CancerLinQ, an initiative of the American Society of Clinical Oncology focuses on collecting and analyzing patient data from electronic health data to drive the process of developing better cancer care technologies.

Patient generated health data (PGHD) is an important tool in the cancer care delivery system. It includes the data collected from the patient's regular environment

in form of WMD data, other local sensor data, and outcomes reported by the patients. PGHD enables us to have a holistic view on the real-time health of the patient (Chung et al. 2016). IoT technologies can be used to collect and transmit these PGHD to oncologists and other researchers in real time. Because of IoT and WMDs, it is possible to collect such health data outside the clinic in real time.

There are a few sensor-based devices having been developed to detect various types of cancer in their early stages. A majority of these devices have focused on skin cancer and breast cancer detection. One such device was developed by S. K. Attili et al., which is a lightweight, organic light-emitting diode using photo-dynamic therapy (PDT) to detect skin cancer (Attili et al. 2009). Some of the other devices for skin cancer detection include a wearable designed by D. D. Godoy et al. which has a series of smart sensors able to detect the ambient temperature, humidity, light, and UV and notifies user through bluetooth about possible UV-harm using a smartphone-based application (Ray et al. 2017). My UV Patch and Violet Plus are two more WMDs to be worn around the wrist, which detects the UV radiation of the sun on the skin—one of the primary causes of skin cancer. The former is designed as an adhesive patch, whereas the latter is designed as a clip attached module.

In case of breast cancer, A. Rahman et al. designed a compact and ultrawide band antenna placed on a flexible substrate, which was able to measure breast cancer using microwave imaging (Rahman et al. 2016). Teng et al. 2017, developed a diffusion-based optical probe working on the principle of using continuous waves to supply the hemodynamic response during neoadjuvant chemotherapy infusions (Teng et al. 2017). H. Bahrami et al. developed a lightweight, flexible, and cheap microwave array of antennas able to detect breast cancer (Bahramiabarghouei et al. 2015).

Beyond these few devices, researchers and engineers around the world are developing better wearable devices integrated with IoT to detect cancer at an early stage. Although most of the devices discussed focused on skin and breast cancer, some research is being done to develop devices to detect prostate cancer (Ray et al. 2017).

17.3.4 Migraines

Migraine is a disorder of the neural as well as the vascular systems of the body (Silberstein and Silberstein 2004; Lipton et al. 2001). It is typically considered as a form of headache, though certain migraines can occur even without any kind of definitive pain.

In the general population, migraine is more prevalent among women, and nearly three times as many women suffer from migraine compared to adult men (Durand 1986). However, the onset of migraines is earlier among boys, which implies that until puberty, boys are likely to suffer from migraines more than girls (Abel 2009). Currently, three genes have been found for migraine and it is anticipated that further genes will be discovered, providing a genetic basis for explaining the disorder.

Migraines are likely to be a major distraction from work and familial responsibilities, since they have a debilitating effect on the energy and activity levels of an individual. Roughly one-third of migraineurs say they have missed school and/or work due to migraines, while 52% to 73% of the people feel their lives are being adversely affected by migraine attacks.

There has been extensive research carried out on the pathophysiology of migraines, however a definite cause or pathway has not yet been identified. The older theory, as propounded by Wolff, was that migraines were caused by constriction and dilation of cranial blood vessels.

However, current research points to the trigeminovascular system, or more specifically, the interactions between the ophthalmic area of the trigeminal nerve with the dura mater and the cranial blood vessels. A more recent theory postulates that the sensitization of neurons, specifically peripheral sensitization, may be responsible for the throbbing pain usually associated with migraines.

Treatment for migraine mostly involves either pharmacological drugs or sustained care under a neurologist and/or optometrist. Ensuring regular sleep and having meals on time may prevent migraines. Specific techniques, which have proven to be effective, include relaxation training, biofeedback training, and cognitive behavioral therapy.

Cefaly is an FDA-approved noninvasive method of treatment for individuals affected by frequent migraines. It is an external trigeminal nerve stimulation device, wherein a self-adhesive electrode is attached to the forehead and the Cefaly device is connected with it. Microimpulses are sent to the two branches of the trigeminal nerve to either relieve the pain in the head region or to prevent future migraine attacks.

17.4 COMPUTATIONAL TESTING METHODS

17.4.1 Epilepsy

Computational analysis provides a time- and cost-effective approach for the assessment of cognitive functions in patients suffering from epilepsy. The computerized testing is advantageous for the evaluation process in attenuation and they are also easily administrable.

17.4.1.1 Computerized Cognitive Testing in Epilepsy (CCTE)

This technique comprises mainly of performing eight tasks including digit span forward and backward, focused attention, incidental recollection issues, verbal learning, and figular memory (Kurzbuch et al. 2013). The performance in various subsets of CCTE shows a match with the results obtained in paper-pencil tests and thus, validates epilepsy (Orsini et al. 2016).

17.4.1.2 Cognitive Drug Research Computerized Assessment System (CDR)

The different assessment techniques of core aspects of cognitive function sustain the ability to conduct various activities in day-to-day life. The major mechanism underlying this technique is change over time in cognitive functions. This technique is simple to administer and is applicable in the studies on psychomotor effects, which are produced by remacenide and carbamazepine in the diagnostic treatment of newly developed epilepsy (Jan 2007).

17.4.1.3 Cognitive Neurophysiological Test (CNT)

The major advantage of this technique is it incorporates the speed and precision of task performance, as well as type of neural resources and the efforts to provide levels

of performance and alertness are increased by this test. The additional knowledge is therefore increased to the specific effects of drugs and their in-depth pharmacokinetics and pharmacodynamics in the human body (Tufenkjian and Lüders 2012; Packer et al. 2015). Validation of this test is in the intellectual effects of antiepileptic drugs (Stufflebeam 2011; Hasegawa 2016).

17.4.1.4 FePsy

It comprises the eleven computerized tests for neurophysiological functions and is a relative database system for storing all the obtained results. It includes various subsets of tests for the measurement of different side-effects produced by the drugs. The simultaneous EEG recording is used to assess the effects of frequency and short and epileptiform EEG discharges in the absence of seizure (Menicucci et al. 2015; Stufflebeam 2011).

17.4.1.5 Neurocog FX

The key feature of this computerized software is the assessment of patients suffering from epilepsy and neurological diseases with respective experimental setups. This specific test was designed to be used in detection of CNS diseases and monitoring neuro-oncology and neurosurgery (Helmstaedter et al. 1996). This software is available only in Germany and the reaction of the patient is monitored in real time.

17.4.2 Cancer

In recent years, there has been several developments in the field of detecting cancer in its early stages using computer programs and applications. There have been some major breakthroughs in this regard.

In 2015, a team of scientists and engineers from the Technical University of Denmark developed a set of algorithms, dubbed as TumorTracer, able to detect the exact location of cancer with a nearly 85% success rate. It enabled the proper identification of the tissue of origin of the cancer, which is crucial for designing a proper treatment plan. This was a game changer for the people who suffered from cancer, whose primary origin was difficult to locate using conventional methods (Marquard et al. 2015).

In March 2017, a team of researchers from the University of California, Los Angeles designed a computer program dubbed CancerLocator, which could detect the tumor's DNA and from where it was coming from using blood samples of the patients. This program works by looking for specific molecular patterns in cancer DNA. It exploits the theory that cell free DNA from cancer cells are often found in the blood stream. By comparing their sequence with a known database of tumor cell sequences it was possible to both detect the cancer and locate its origin (Kang et al. 2017).

Beyond these two, there are two important algorithms used by several research groups to diagnose cancer. One of them was random forest. Cuong Nguyen et al., in 2013, designed a system based on this algorithm able to differentiate benign breast cancer tumors from malignant ones using a random forest algorithm. It was able to do so with an accuracy rate greater than 99% (Nguyen et al. 2013). This method was obtained to solve diagnosis problems by classifying the Wisconsin Breast Cancer Diagnosis Dataset (WBCDD) and to solve prognosis problems by classifying it.

Another algorithm popularly used in cancer detection is support vector machine (Sweilam et al. 2010).

17.5 CONCLUSION

To summarize, it is clear that WMDs have certainly proven to be hugely beneficial for the health care community. Not only have they significantly improved the lives of people suffering from various disorders, they have also made the job of medical professionals easier by introducing the concept of remote monitoring. The potential drawbacks, however minor, cannot be entirely ignored. Security and confidentiality of patient data is absolutely crucial and forms the framework of wearable devices. The unreasonably high cost of certain devices can sometimes act as a deterrent, however with increasingly newer technologies being developed, it can be hoped that wearable devices will become affordable for a majority of the population in the near future.

REFERENCES

Abel, H. 2009. "Migraine Headaches: Diagnosis and Management." *Optometry* 80 (3). Mosby, Inc:138–48. https://doi.org/10.1016/j.optm.2008.06.008.

Attili, S. K., A. Lesar, A. McNeill, M. Camacho-Lopez, H. Moseley, S. Ibbotson, I. D. W. Samuel, and J. Ferguson. 2009. "An Open Pilot Study of Ambulatory Photodynamic Therapy Using a Wearable Low-Irradiance Organic Light-Emitting Diode Light Source in the Treatment of Nonmelanoma Skin Cancer." *British Journal of Dermatology* 161 (1):170–73. https://doi.org/10.1111/j.1365-2133.2009.09096.x.

Bahramiabarghouei, H., E. Porter, A. Santorelli, B. Gosselin, M. Popović, and L. A. Rusch. 2015. "Flexible 16 Antenna Array for Microwave Breast Cancer Detection." *IEEE Transactions on Biomedical Engineering* 62 (10):2516–25. https://doi.org/10.1109/TBME.2015.2434956.

Bernhardt, B. C., S.-J. Hong, A. Bernasconi, and N. Bernasconi. 2015. "Magnetic Resonance Imaging Pattern Learning in Temporal Lobe Epilepsy: Classification and Prognostics." *Annals of Neurology* 77 (3):436–46. https://doi.org/10.1002/ana.24341.

Beurrier, C., B. Bioulac, J. Audin, and C. Hammond. 2001. "High-Frequency Stimulation Produces a Transient Blockade of Voltage-Gated Currents in Subthalamic Neurons." *J Neurophysiol* 85 (4):1351–56. http://www.ncbi.nlm.nih.gov/pubmed/11287459%5Cnhttp://jn.physiology.org/content/jn/85/4/1351.full.pdf.

Błaszczyk, B., and S. J. Czuczwar. 2016. "Epilepsy Coexisting with Depression." *Pharmacological Reports* 68 (5):1084–92. https://doi.org/10.1016/j.pharep.2016.06.011.

Bobade, Y., and R. M. Walli. 2015. "A Review of Wearable Health Monitoring Systems." *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering* 4 (10):8386–90. https://doi.org/10.15662/IJAREEIE.2015.0410076.

Chung, A. E., R. E. Jensen, and E. M. Basch. 2016. "Leveraging Emerging Technologies and the 'Internet of Things' to Improve the Quality of Cancer Care." *Journal of Oncology Practice* 12 (10):863–66. https://doi.org/10.1200/JOP.2016.015784.

Durand, D. 1986. "Electrical Stimulation Can Inhibit Synchronized Neuronal Activity." *Brain Research* 382 (1):139–44. https://doi.org/10.1016/0006-8993(86)90121-6.

Duyn, J. H., P. van Gelderen, T.-Q. Li, J. A. de Zwart, A. P. Koretsky, and M. Fukunaga. 2007. "High-Field MRI of Brain Cortical Substructure Based on Signal Phase." *Proceedings of the National Academy of Sciences* 104 (28):11796–801. https://doi.org/10.1073/pnas.0610821104.

Fang, Z., Y. Yang, X. Chen, W. Zhang, Y. Xie, Y. Chen, Z. Liu, and W. Yuan. 2017. "Advances in Autoimmune Epilepsy Associated with Antibodies, Their Potential Pathogenic Molecular Mechanisms, and Current Recommended Immunotherapies." *Frontiers in Immunology* 8 (APR). https://doi.org/10.3389/fimmu.2017.00395.

Findlay, J. Y., B. Gali, M. T. Keegan, C. M. Burkle, and D. J. Plevak. 2006. "Vasotrac® Arterial Blood Pressure and Direct Arterial Blood Pressure Monitoring during Liver Transplantation." *Anesthesia and Analgesia* 102 (3):690–93. https://doi.org/10.1213/01.ane.0000196512.96019.e4.

French, J. A. 2007. "Refractory Epilepsy: Clinical Overview." *Epilepsia* 48 (SUPPL. 1):3–7. https://doi.org/10.1111/j.1528-1167.2007.00992.x.

Hasegawa, D. 2016. "Diagnostic Techniques to Detect the Epileptogenic Zone: Pathophysiological and Presurgical Analysis of Epilepsy in Dogs and Cats." *Veterinary Journal* 215. Elsevier Ltd:64–75. https://doi.org/10.1016/j.tvjl.2016.03.005.

Helmstaedter, C., B. Kemper, and C. E. Elger. 1996. "Neuropsychological Aspects of Frontal Lobe Epilepsy." *Neuropsychologia* 34 (5):399–406. https://doi.org/10.1016/0028-3932(95)00121-2.

Jan, M. M. 2007. "The Value of Seizure Semiology in Lateralizing and Localizing Partially Originating Seizures." *Neurosciences (Riyadh, Saudi Arabia)* 12 (3):185–90. http://www.ncbi.nlm.nih.gov/pubmed/21857567.

Jewells, V., H. W. Shin. 2014. "Review of Epilepsy—Etiology, Diagnostic Evaluation and Treatment." *International Journal of Neurorehabilitation* 1 (3):1–8. https://doi.org/10.4172/2376-0281.1000130.

Kang, S., Q. Li, Q. Chen, Y. Zhou, S. Park, G. Lee, B. Grimes et al. 2017. "CancerLocator: Non-Invasive Cancer Diagnosis and Tissue-of-Origin Prediction Using Methylation Profiles of Cell-Free DNA." *Genome Biology* 18 (1):53. https://doi.org/10.1186/s13059-017-1191-5.

Kurzbuch, K., E. Pauli, L. Gaál, F. Kerling, B. S. Kasper, H. Stefan, H. Hamer, and W. Graf. 2013. "Computerized Cognitive Testing in Epilepsy (CCTE): A New Method for Cognitive Screening." *Seizure* 22 (6):424–32. https://doi.org/10.1016/j.seizure.2012.08.011.

Lipton, R. B., F. Stewart, S. Diamond, M. L. Diamond, and M. Reed. 2001. "Prevalence and Burden of Migraine in the United States: Data from the American Migraine Study II." *Headache* 41 (7):646–57. https://doi.org/10.1046/j.1526-4610.2001.041007646.x.

Marquard, A. M., N. J. Birkbak, C. E. Thomas, F. Favero, M. Krzystanek, C. Lefebvre, C. Ferté et al. 2015. "TumorTracer: A Method to Identify the Tissue of Origin from the Somatic Mutations of a Tumor Specimen." *BMC Medical Genomics* 8 (1):58. https://doi.org/10.1186/s12920-015-0130-0.

McIntyre, C. C., M. Savasta, L. K.-L. Goff, and J. L. Vitek. 2004. "Uncovering the Mechanism(s) of Action of Deep Brain Stimulation: Activation, Inhibition, or Both." *Clinical Neurophysiology* 115 (6):1239–48. https://doi.org/10.1016/j.clinph.2003.12.024.

Stufflebeam, S. M. 2011. "Clinical Magnetoencephalography for Neurosurgery." *Neurosurgery Clinics of North America* 22 (2):153–67. https://doi.org/http://dx.doi.org/10.1016/j.nec.2010.11.006.

Menicucci, D., A. Piarulli, P. Allegrini, R. Bedini, M. Bergamasco, M. Laurino, L. Sebastiani, and A. Gemignani. 2015. "Looking for a Precursor of Spontaneous Sleep Slow Oscillations in Human Sleep: The Role of the Sigma Activity." *International Journal of Psychophysiology* 97 (2). Elsevier B.V.:99–107. https://doi.org/10.1016/j.ijpsycho.2015.05.006.

Nguyen, C., Y. Wang, and H. N. Nguyen. 2013. "Random Forest Classifier Combined with Feature Selection for Breast Cancer Diagnosis and Prognostic." *Journal of Biomedical Science and Engineering* 6 (5):551–60. https://doi.org/10.4236/jbise.2013.65070.

Orsini, A., F. Zara, and P. Striano. 2016. "Recent Advances in Epilepsy Genetics." *Neuroscience Letters*. Elsevier Ireland Ltd. https://doi.org/10.1016/j.neulet.2017.05.014.

Packer, R., M. Berendt, S. Bhatti, M. Charalambous, S. Cizinauskas, L. D. Risio, R. Farquhar et al. 2015. "Inter-Observer Agreement of Canine and Feline Paroxysmal Event Semiology and Classification by Veterinary Neurology Specialists and Non-Specialists." *BMC Veterinary Research* 11 (1):39. https://doi.org/10.1186/s12917-015-0356-2.

Rahman, A., M. T. Islam, M. J. Singh, S. Kibria, and M. Akhtaruzzaman. 2016. "Electromagnetic Performances Analysis of an Ultra-Wideband and Flexible Material Antenna in Microwave Breast Imaging: To Implement A Wearable Medical Bra." *Scientific Reports* 6 (November). Nature Publishing Group:1–11. https://doi.org/10.1038/srep38906.

Ray, P. P., D. Dash, and D. De. 2017. "A Systematic Review of Wearable Systems for Cancer Detection: Current State and Challenges." *Journal of Medical Systems* 41 (11). https://doi.org/10.1007/s10916-017-0828-y.

Silberstein, S., and S. D. Silberstein. 2004. "Migraine Pathophysiology and Its Clinical Implications." *Blackwell Science* 24:2–7. https://doi.org/10.1111/j.1468-2982.2004.00892.x.

Sledge, G. W., C. A. Hudis, S. M. Swain, P. M. Yu, J. T. Mann, R. S. Hauser, and A. S. Lichter. 2013. "ASCO's Approach to a Learning Health Care System in Oncology." *Journal of Oncology Practice / American Society of Clinical Oncology* 9 (3):145–48. https://doi.org/10.1200/JOP.2013.000957.

Sweilam, N. H., A. A. Tharwat, and N. K. Abdel Moniem. 2010. "Support Vector Machine for Diagnosis Cancer Disease: A Comparative Study." *Egyptian Informatics Journal* 11 (2): 81–92. https://doi.org/10.1016/j.eij.2010.10.005.

Teng, F., T. Cormier, A. Sauer-Budge, R. Chaudhury, V. Pera, R. Istfan, D. Chargin, S. Brookfield, N. Y. Ko, and D M. Roblyer. 2017. "Wearable Near-Infrared Optical Probe for Continuous Monitoring During Breast Cancer Neoadjuvant Chemotherapy Infusions." *Journal of Biomedical Optics* 22 (1):14001. https://doi.org/10.1117/1.JBO.22.1.014001.

Teng, X.-F., Y.-T. Zhang, C. C. Y. Poon, and P. Bonato. 2008. "Wearable Medical Systems for P-Health." *IEEE Reviews in Biomedical Engineering* 1:62–74. https://doi.org/10.1109/RBME.2008.2008248.

Thomas, G. P., and B C. Jobst. 2015. "Critical Review of the Responsive Neurostimulator System for Epilepsy." *Medical Devices: Evidence and Research* 8:405–11. https://doi.org/10.2147/MDER.S62853.

Tufenkjian, Kr., and H. O. Lüders. 2012. "Seizure Semiology: Its Value and Limitations in Localizing the Epileptogenic Zone." *Journal of Clinical Neurology (Korea)* 8 (4): 243–50. https://doi.org/10.3988/jcn.2012.8.4.243.

Vandecasteele, K., T. D. Cooman, Y. Gu, E. Cleeren, K. Claes, W. V. Paesschen, S. V. Huffel, and B. Hunyadi. 2017. "Automated Epileptic Seizure Detection Based on Wearable ECG and PPG in a Hospital Environment." *Sensors (Switzerland)* 17 (10):1–12. https://doi.org/10.3390/s17102338.

Weber, R. H., and R. Weber. 2010. "Internet of Things." *Development*, 1–8. https://doi.org/10.1007/978-3-642-11710-7.

Witt, J. A., W. Alpherts, and C. Helmstaedter. 2013. "Computerized Neuropsychological Testing in Epilepsy: Overview of Available Tools." *Seizure* 22 (6). BEA Trading Ltd:416–23. https://doi.org/10.1016/j.seizure.2013.04.004.

Yilmaz, T., R. Foster, and Y. Hao. 2010. "Detecting Vital Signs with Wearablewireless Sensors." *Sensors* 10 (12):10837–62. https://doi.org/10.3390/s101210837.

You, G., Z. Sha, and T. Jiang. 2012. "The Pathogenesis of Tumor-Related Epilepsy and Its Implications for Clinical Treatment." *Seizure* 21 (3). BEA Trading Ltd:153–59. https://doi.org/10.1016/j.seizure.2011.12.016.

18 People with Disabilities
The Helping Hand of IoT

Ashmita Das, Sayak Mitra, and Shampa Sen

CONTENTS

18.1 Surviving with a Disability 299
18.2 Existing Technologies to Help Disabled People 300
18.3 Disabled to IoT-Abled 301
18.3.1 Physical and Dexterity 303
18.3.2 Visual 303
18.3.3 Hearing 305
18.3.4 Cognitive 306
18.4 The Path Ahead 307
18.4.1 Accessibility 307
18.4.2 Broadband Access 308
18.4.3 Standardization 308
18.4.4 Privacy 308
18.4.5 Security 308
References 308

18.1 SURVIVING WITH A DISABILITY

The World Health Organization (WHO) describes disability as an inability to lead a normal lifestyle according to their age, gender, social, and cultural state (Sen and Yurtsever 2007). It is an umbrella term covering impairments, boundary of participation, and restriction of activities. Currently, almost 15% of the world's population is estimated to be living with one form of disability or another.

We generally classify disabilities into four categories (World Health Organization (WHO 2003). They are:

1. Physical disability
2. Visual disability
3. Hearing disability
4. Cognitive disability

In today's world, the invention of various aids such as special glasses, walking sticks, and many more provide people support to lead an independent lifestyle under any circumstances (Stucki 2005). Hearing aids help deaf people to become connected. An affordable and effective alternative for deaf people is the American sign language (ASL) (Keating and Mirus 2004), which has evolved several means

of communication among deaf people. The innovation of the wheelchair was as miraculous as the invention of the wheel itself, as it saved physically disabled people from being confined to a single place for a long time. But where a wheelchair causes problem, the crutches offer an alternative. Similarly, many other technologies have evolved to help people with disabilities and make their lives a bit easier.

The rapid development of technology has, however, brought more help to disabled people. One such boon is Internet of things (IoT). In essence, IoT promises that everyone and everything could be connected to each other. This makes it easier for people with disabilities to perform tasks they are usually unable to perform, simply by issuing a command to a smart device. Thanks to IoT, disabled people can change the thermostat settings of their home when they feel cold, summon a glass of water when they feel thirsty, open the door by means of a voice command, and even receive proper medications at the precise moment. Perhaps the most revolutionary application of IoT has been the automated smart home, which ensures, above all, the security of people with disabilities. IoT also allows concerned caregivers to monitor the well-being of disabled people and be notified immediately in case of emergencies.

This chapter focuses on the many ways IoT has helped people with disabilities gain confidence and become more sociable and communicable, while ensuring their proper health and well-being. It ends on an optimistic note, urging authorities to make these technologies more easily accessible to everybody, and not for profit, thus ensuring nobody is left out of society simply because they are unable to live like the majority.

18.2 EXISTING TECHNOLOGIES TO HELP DISABLED PEOPLE

Assistive technology includes all technologies designed to assist disabled people, starting from the wheelchair to hearing aids (Figure 18.1).

FIGURE 18.1 Existing technologies to help disabled people.

For the deaf, assistive devices such as hearing aids play a significant role. They are divided into two categories, external and internal devices. External assistive devices are worn by the patients, while internal ones are implanted inside the ear.

Among external devices, hearing aids (a battery operated electronic device) (Cox and Alexander 1983), assistive listening devices (ALDs) (Zanin and Rance 2016) amplify the incoming sounds to make them audible for the patients suffering from superficial or middle hearing loss problems. Specific telephone equipment helps the patients to communicate over the telephone. Other than that, several automated circuits are externally or internally implanted in different parts for producing profound sound (Garin et al. 2004). RetroX, Vibrant Sound Bridge (VSB) (Vincent et al. 2004), and cochlear implant devices (Minoda et al. 2004) are also helpful for severe hearing loss problems. An online sign language translator was invented converting English news or related articles to sign language by using a Natural Language Processor (Saleem and Alagha 2016).

Several technologies have been developed to help blind people. Screen readers plays a significant role to help the low vision or visually impaired patients (Neto and Fonseca 2014). Braille watches and braille printers are some of those latest inventions offering patients an opportunity to lead a normal life (Kurze 1996).

Wheelchairs, crutches, and artificial legs help the physically challenged people and make them more confident and independent. Above all, speech recognition, special keyboards such as "SmartNav," "Intellikeys," and "Sesame enable apps" are considered to be most effective and innovative solution.

Visually impaired people benefit from devices allowing them to use computers. They also present an alternative way to access the contents of the screen, process data, and command the computer. It is basically either a device or equipment which is able to read Braille language, or a software service (e.g., screen reading software) (Arrigo 2005).

18.3 DISABLED TO IoT-ABLED

The IoT provides the most effective and reliable solution for all problems related to disability. Nowadays, all things in the physical world achieve interconnection with each other through data exchange via the Internet. IoT highlights a complex paradigm, which also builds strong communication between things, things and humans, and even between humans (Coetzee and Olivrin 2012). It also has potential to solve problems related to remote health monitoring, fitness programs, etc. Various medical devices, sensors, and imaging devices are considered to be a core part of the IoT. IoT-based medical devices are expected to provide several advantages such as reduced costs, and increased quality of life and user experience (Ngu et al. 2017).

IoT health care networks are an essential part of IoT in the health care domain. They support the access to the IoT backbone and also facilitates the transmission of medical data (Figure 18.2).

IoT provides real-time access to device information and also allows for the remote management of devices. A combined system of IoT and cloud computing features a specialized scalable and globally accessible web application, which also provides an interface for effective and enhanced communication amongst the devices and also between the device and its user.

FIGURE 18.2 (a) The IoT health care network and its components, (b) The architecture of IoT equipments.

IoT-applied smart house or shopping scenarios helps the disabled person to carry out their daily activities. This also increases their self-confidence and autonomy. Smart houses enable the disabled person to move independently, handle objects, and also communicate effectively. Monitoring systems reduce or eliminate the dependency on caregivers by improving the autonomy of the individuals.

On the other side, disabled children can experience improved linguistic skills, emotional skills, and also social skills by using interactive learning and IoT (Hengeveld et al. 2009).

18.3.1 Physical and Dexterity

IoT plays a significant role in making the lives of physically challenged people easier, facilitating communications and helping them integrate into society.

Two equipments are known to play a major role, (1) body sensors, acutors, and neurochips, and (2) body sensors with RFID technology.

- Body sensors, acutors, and neurochips

Acutors are able to functionally reanimate paralyzed limbs. The sensors connected with the nerves are able to detect the user's urge to move a particular muscle, whereas acutors stimulate the movement of those muscles.

BIOnic Neurons (BIONs) (Domingo 2012) are wireless capsules injected in various parts of the body near the motor nerve. They detect voluntary command and stimulate a related appropriate response to the neuromuscular cells.

Electrocorticography (ECoG) (Yanagisawa et al. 2012) signals seem to control prosthetic arm controls and provide clinical motor restoration for patients.

Facial expression-based computer cursor control systems (Vasanthan et al. 2012) are also used commonly as an assistive device for the physically disabled persons. Various indicative expressions such as left and right cheek movement, eyebrow rising and lowering, and mouth open or shut controls various computer-related applications. Four illuminator stickers are placed in proper places to provide instructions, which are communicated to the computer through the BASIC STAMP microcontroller. The x-y coordinate changes are detected on the video image through movement marker and facial expression. The result is also represented by a binary number which is useful for controlling the cursor.

m-IoT (Istepanian et al. 2011) is a new concept introduced for helping the disabled elderly. It is a combination of IoT and m-health (mobile health i.e., wireless medical care). This technology synchronizes the functionalities of m-health and IoT for future and innovative applications. It connects IP-based communication technologies 6LoWPAN with 4G networks by introducing a new paradigm.

18.3.2 Visual

Visually impaired people also include people who suffer from partial blindness and even color blindness. People with visual disabilities face two main problems. First, they are not able to understand the pictorial or graphical information displayed on the screen. Second, the pointing devices (e.g., a mouse), which require eye-hand coordination, also create difficulties for them (Kavcic 2005).

The components introduced for visually impaired patients are different types of sensors such as body micro- and nanosensors, etc., as well as sensor assistive devices like RFID-based assistive devices. Schwiebert et al. discovered an artificial retina to restore a measure of vision to patients suffering from degenerative blindness due to retinitis pigmentosa and age-related diseases. In these two diseases, the photoreceptor cells of the outer retina start degrading but they do not affect the optic nerve, which is made of inner retinal ganglion nerve cells. Similarly, a camera attached to a pair

of glasses is used to send data to the implanted device (body sensors) placed on the retina. This sensor made retina (Schwiebert et al. 2001) stimulates specified ganglion cells using electric impulses. The ganglion cells convert them into neurological signals and transfer the signals to the brain via the optical nerve.

Another important implant developed to solve the problem of visual impairment is the Bio-Retina (Domingo 2012). The Bio-Retina incorporates several nanosized components into the retinal implants and is used as a replacement for damaged photoreceptor cells. It converts the received natural light into electrical impulses, which stimulate neurons and transfers the image received by the Bio-Retina to the brain. The nanosized components are charged by a unique pair of activation eyeglasses.

RFID-based assistive devices are very useful for visually impaired people to navigate in unfamiliar envoronments. RFID tags are placed at the center of the sidewalks to prevent accidental falls. The Sesamonet System (Faggion and Azzalin 2011) provides safe and secure paths to allow visually challenged people to move and navigate independently. It is based on a path marked by passive low frequency RFID transponders. A unique identification number (UID) is assigned to every transponder and is stored in a database. The electronic stick reads the UID present on the road and transfers the data to a portable electronic device or smartphone where the Sesamonet software is opened. The software constantly maintains communication with saved local databases and relates the user's position to the important information of the surrounding environment (Figure 18.3).

Virtual leading blocks (Amemiya et al. 2004) is an effective tool for deaf-blind people. The wearable computer and ubiquitous environment play a role of virtual navigator. The navigator guides the user on the basis of position data detected by the RFID tags and the orientation of motion sensors in real time. Verbal instruction is translated by finger braille method, while the nonverbal information is translated into spatial, accentual, and numerical change of tactile stimuli.

Obstacle Recognition System (Nassih et al. 2012) is another sophisticated system proposed regarding this purpose. It consists of a can, which also has an RFID tag reader on it and a braille interface. It also possesses a mechanical battery, which provides full-time energy during the operation of the device.

FIGURE 18.3 Electronic stick for visually impaired people.

Another important prototype "SmartVision" (Fernandes et al. 2013) has also been invented for this purpose. It uses an electronic walking stick enabling people to activate sensors on the floor, which are connected to form connected lines and clusters. These are able to guide the person by building a safe route to common locations of interest, such as a grocery store, book store, restaurant, and others. The software is able to run offline on a portable device using a local representation of the data presented in a geographic information system (GIS) stored in a tag identifier, which is a remote server. This uses corresponding geographic coordinates and tag ownership for tracing the location. After sensing the RFID, the software handles the inputs in two ways: navigating and scrolling. For navigation purposes, the software calculates the routes between the point of interest and the destination point, and each input from the location module triggers the navigation module to direct as well as keep the record on track. But for scrolling, the system picks the current coordinates and is able to find out all the relevant features of the surroundings using the GIS module.

Almeighder proposed a navigation system (Alghamdi et al. 2013) to help guide blind people in indoor locations where GPSs are not able to work properly. Indoor features are identified using RFID tags and users carry a mobile reader. The system helps to determine the shortest path to the destination. After calculating the shortest path, the path points are indicated by activated tags, which are scanned by a QR code and navigate the users to their destination.

18.3.3 Hearing

Deaf people have difficulties in detecting sounds and differentiating auditory information from the background. They communicate mostly through sign language (World Health Organization 1999). The World Health Organization (WHO) classified hearing disabilities into five categories. They are listed in Table 18.1 (Duthey 2013).

HandTalk (Sarji 2008) helps deaf people break the communication barrier. It is basically a pair of normal cotton gloves in which several flexor sensors are incorporated. They are normally embedded along the length of the each finger as

TABLE 18.1
Classification of Hearing Disabilities

Impairment Type/Grade	Audiometric Range (dbHL)	Description
No Impairment (Grade:0)	>25	Able to hear whisper
Slight Impairment (Grade:1)	26–40	Able to hear repeated word in normal voice within 1 meter
Moderate Impairment (Grade:2)	41–60	Able to hear repeated word in raised voice within 1 meter
Severe Impairment (Grade:3)	61–80	Able to hear the shouted voice
Profound Impairment (Grade:4)	<80	Not able to hear the shouted voice

well as the thumb. Normal gestures are sensed by sensors and transmitted to the BlueSentry software. This software converts the data into digital language in a range of 0 to 65,500. These numbers provide proper reflexes by using commercial off-the-shelf (COTS) elements. The architecture of these hand gloves is illustrated in Figure 18.4.

Language Acquisition Manipulatives Blending Early Childhood Research and Technology (LAMBERT) (Parton et al. 2010) allows the children, especially the deaf preschoolers, to play with objects that help them to learn the ASL language of most commonly used words via Multimedia Presentation. A 15–20 second multimedia presentation is prepared for each object. Various clip arts, coloring objects, and written descriptions help the children learn ASL.

18.3.4 Cognitive

Cognitive impairments include the impairments of thinking, memory, language, and perception. Several learning disabilities such as attention deficit disorders, dyslexia, and dementia fall into this category (Castaneda et al. 2008).

Brain computer interface (BCI) (Fazel-Rezai et al. 2011), (Lee et al. 2013) technology provides an electronic interface for communicating messages and commands to the computer from the human brain. BCI technology has the potential to improve the quality of life and increase its productivity. A commercially available product using this technology is the Nano Chip. This technology is also used for stroke rehabilitation (Ang and Guan 2013).

The Thought Translational Device (TTD) (Birbaumer et al. 2000) is a very innovative approach for helping mentally impaired patients. It is a device through

FIGURE 18.4 (a) Schematic model of HandTalk, (b) Working of HandTalk.

which patients are able to self-regulate the slow cortical potentials (SCPs) of their electroencephalograms (EEG). After learning the self-regulation process, patients are asked to select letters, words, pictograms, etc. from a computerized language support system, which finally help them describe and demonstrate their thoughts to others. As a result, mentally challenged people are able to establish a communicative network in society.

The assistive and augmentative communication (AAC) (Ann 2012) tool utilizes human computer interfacing to assist the cerebral palsy patients with their daily communications. Real-time emotions are recognized by the display and also send alerts to the caretakers through an SMS (Figure 18.5).

18.4 THE PATH AHEAD

In this chapter, we discussed the uses and advantages of IoT in improving the quality of life of disabled persons. But there are several drawbacks needing to be addressed before its worldwide adoption.

18.4.1 Accessibility

There are numerous companies and organizations exploiting IoT in some way or another. Accessibility should be maintained at each stage of the developmental process. All the parties related to this sector should maintain a universal design pattern (Alkhalil and Ramadan 2017).

IoT-related devices bring a huge profit to their manufacturing companies, but their accessibility is limited to a small part of society due to their excessive price tags. Companies should start the process of making the devices more cost effective to help the people in need.

FIGURE 18.5 Working of assistive and augmentative communication.

18.4.2 Broadband Access

There are many patients who are not well accustomed with the Internet, as they are not able to access the web all the time. The broadband access must be increased by developing the infrastructure to realize the benefits of IoT.

18.4.3 Standardization

IoT has emerged with multidisciplinary applications in various fields, but there are problems with standardization of the product at present. Several organization have started working on it.

One such organization is the IEEE Standard Association (IEEESA), which has announced several parameters related to the standardization of IoT. These parameters are bluetooth connectivity, compatibility with Wi-Fi technology, and many more (Bandyopadhyay and Sen 2011).

18.4.4 Privacy

IoT collects data from the users' every single movement to understand their lifestyle patterns as well as their needs in a particular situation, which raises the question of privacy. Disabled people should be made aware of the data collection activities by providing proper disclaimer, and also providing an option for choosing the data types that help the user to decide which data they want to share (Zhang et al. 2014).

18.4.5 Security

IoT also creates a debate on security issues because large amounts of personal data are collected and stored by the software companies. This matter is discussed within companies and through private-public partnerships.

IoT is still in the developmental stages, and it will take years to become a part of our daily lives. IoT has to overcome several challenges, such as guidelines for standardization, extension of broadband Internet networks, private data protection, and many other before revolutionizing the lives of its users (van Kranenburg and Bassi 2012; Bekara 2014).

REFERENCES

Alghamdi, S., R. V. Schyndel, and A. Alahmadi. 2013. "Indoor Navigational Aid Using Active RFID and QR-Code for Sighted and Blind People." In *Proceedings of the 2013 IEEE 8th International Conference on Intelligent Sensors, Sensor Networks and Information Processing: Sensing the Future, ISSNIP 2013,* 1: 18–22. doi:10.1109/ISSNIP.2013.6529756.

Alkhalil, A., and R. A. Ramadan. 2017. "IoT Data Provenance Implementation Challenges." *Procedia Computer Science*, 109: 1134–39. doi:10.1016/j.procs.2017.05.436.

Amemiya, T., J. Yamashita, K. Hirota, and M. Hirose. 2004. "Virtual Leading Blocks for the Deaf-Blind: A Real-Time Way-Finder by Verbal-Nonverbal Hybrid Interface and High-Density RFID Tag Space." In *Proceedings—Virtual Reality Annual International Symposium*, 165–72. doi:10.1109/VR.2004.1310070.

Ang, K. K., and C. Guan. 2013. "Brain-Computer Interface in Stroke Rehabilitation." *Journal of Computing Science and Engineering* 7 (2): 139–46. doi:10.5626/JCSE.2013.7.2.139.

Ann, O. C. 2012. "Helping Disabled People: The Potentials of Biometric Information." *Interactive Technology and Smart Education* 9 (3): 153–70. doi:10.1108/17415651211258272.

Arrigo, M. 2005. "E-Learning Accessibility for Blind Students." *Recent Resaerch Development in Learning Technologies*, 1–5.

Bandyopadhyay, D., and J. Sen. 2011. "Internet of Things: Applications and Challenges in Technology and Standardization." *Wireless Personal Communications*, 58: 49–69. doi:10.1007/s11277-011-0288-5.

Bekara, C. 2014. "Security Issues and Challenges for the IoT-Based Smart Grid." *Procedia Computer Science*, 34: 532–37. doi:10.1016/j.procs.2014.07.064.

Birbaumer, N., A. Kübler, N. Ghanayim, T. Hinterberger, J. Perelmouter, J. Kaiser, I. Iversen, B. Kotchoubey, N. Neumann, and H. Flor. 2000. "The Thought Translation Device (TTD) for Completely Paralyzed Patients." *IEEE Transactions on Rehabilitation Engineering* 8 (2): 190–93. doi:10.1109/86.847812.

Castaneda, A. E., A. Tuulio-Henriksson, M. Marttunen, J. Suvisaari, and J. Lönnqvist. 2008. "A Review on Cognitive Impairments in Depressive and Anxiety Disorders with a Focus on Young Adults." *Journal of Affective Disorders*. 106(1–2): 1–27. doi:10.1016/j.jad.2007.06.006.

Coetzee, L., and G. Olivrin. 2012. "Inclusion Through the Internet of Things." *Assistive Technologies*. doi:10.5772/31929.

Cox, R. M., and G. C. Alexander. 1983. "The Abbreviated Profile of Hearing Aid Benefit." *British Journal of Audiology* 17 (1): 31–48. doi:10.1097/00003446-199504000-00005.

Domingo, M. C. 2012. "An Overview of the Internet of Things for People with Disabilities." *Journal of Network and Computer Applications* 35 (2). Elsevier: 584–96. doi:10.1016/j.jnca.2011.10.015.

Duthey, B. 2013. "Background Paper 6.21 Hearing Loss." *World Health Organization* 1 (February): 6.

Faggion, L., and G. Azzalin. 2011. "Low-Frequency RFID Based Mobility Network for Blind People: System Description, Evolutions and Future Developments." In *2011 IEEE International Conference on RFID-Technologies and Applications, RFID-TA 2011*, 364–69. doi:10.1109/RFID-TA.2011.6068663.

Fazel-Rezai, R., And Ahmad, W., C. Guger, G. Edlinger, and G. Krausz. 2011. *Recent Advances in Brain-Computer Interface Systems*. doi:10.5772/579.

Fernandes, H., V. Filipe, P. Costa, and J. Barroso. 2013. "Location Based Services for the Blind Supported by RFID Technology."*Procedia Computer Science*, 27: 2–8. doi:10.1016/j.procs.2014.02.002.

Garin, P., F. Genard, C. Galle, and J. Jamart. 2004. "The RetroX Auditory Implant for High-Frequency Hearing Loss." *Otology & Neurotology* 25 (4): 511–19. doi:10.1097/00129492-200407000-00019.

Hengeveld, B., C. Hummels, K. Overbeeke, R. Voort, H. van Balkom, and J. de Moor. 2009. "Tangibles for Toddlers Learning Language." In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction—TEI '09*, 161. doi:10.1145/1517664.1517702.

Istepanian, R. S. H., A. Sungoor, A. Faisal, and N. Philip. 2011. "Internet of M-Health Things 'M-IOT'." *IET Seminar on Assisted Living* 2011, 20–20. doi:10.1049/ic.2011.0036.

Kavcic, A. 2005. "Software Accessibility: Recommendations and Guidelines." In *EUROCON 2005—The International Conference on Computer as a Tool* 2: 1024–27. doi:10.1109/EURCON.2005.1630123.

Keating, E., and G. Mirus. 2004. "American Sign Language in Virtual Space: Interactions between Deaf Users of Computer-Mediated Video Communication and the Impact of Technology on Language Practices." *Language in Society* 32 (5): 693–714. doi:10.1017/S0047404503325047.

Kurze, M. 1996. "TDraw: A Computer-Based Tactile Drawing Tool for Blind People." In *Assets '96 Proceedings of the Second Annual ACM Conference on Assistive Technologies*, 131–38. doi:10.1145/228347.228368.

Lee, S., Y. Shin, S. Woo, K. Kim, and H.-N. Lee. 2013. "Review of Wireless Brain-Computer Interface Systems." In *Brain-Computer Interface Systems—Recent Progress and Future Prospects*, 215–37. doi:http://dx.doi.org/10.5772/56436.

Minoda, R., M. Izumikawa, K. Kawamoto, and Y. Raphael. 2004. "Strategies for Replacing Lost Cochlear Hair Cells." *Neuroreport* 15 (7): 1089–92. doi:10.1097/01.wnr.0000126216.79493.8d.

Nassih, M., I. Cherradi, Y. Maghous, B. Ouriaghli, and Y. Salih-Alj. 2012. "Obstacles Recognition System for the Blind People Using RFID." In *2012 Sixth International Conference on Next Generation Mobile Applications, Services and Technologies*, 60–63. doi:10.1109/NGMAST.2012.28.

Neto, R., and N. Fonseca. 2014. "Camera Reading for Blind People." *Procedia Technology* 16: 1200–1209. doi:10.1016/j.protcy.2014.10.135.

Ngu, A. H., M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng. 2017. "IoT Middleware: A Survey on Issues and Enabling Technologies." *IEEE Internet of Things Journal* 4 (1): 1–20. doi:10.1109/JIOT.2016.2615180.

Parton, B. S., R. Hancock, and A. D. DuBusde Valempré. 2010. "Tangible Manipulatives and Digital Content: The Transparent Link That Benefits Young Deaf Children." In *Proceedings of IDC2010 The 9th International Conference on Interaction Design and Children*, 300–303. doi:10.1145/1810543.1810597.

Saleem, K., and I. Alagha. 2016. "System for People with Hearing Impairment to Solve Their Social Integration." In *2015 5th International Conference on Information and Communication Technology and Accessibility, ICTA 2015*. doi:10.1109/ICTA.2015.7426913.

Sarji, D. K. 2008. "HandTalk: Assistive Technology for the Deaf." *Computer* 41 (7): 84–86. doi:10.1109/MC.2008.226.

Schwiebert, L., S. K. S. Gupta, and J. Weinmann. 2001. "Research Challenges in Wireless Networks of Biomedical Sensors." In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking—MobiCom '01*, 151–65. doi:10.1145/381677.381692.

Sen, E., and S. Yurtsever. 2007. "Difficulties Experienced by Families with Disabled Children." *Journal for Specialists in Pediatric Nursing* 12 (4): 238–52. doi:10.1111/j.1744-6155.2007.00119.x.

Stucki, G. 2005. "International Classification of Functioning, Disability, and Health (ICF)." *American Journal of Physical Medicine & Rehabilitation* 84 (10): 733–40. doi:10.1097/01.phm.0000179521.70639.83.

van Kranenburg, R., and A. Bassi. 2012. "IoT Challenges." *Communications in Mobile Computing* 1 (1): 9. doi:10.1186/2192-1121-1-9.

Vasanthan, M., M. Murugappan, R. Nagarajan, B. Ilias, and J. Letchumikanth. 2012. "Facial Expression Based Computer Cursor Control System for Assisting Physically Disabled Person." In *Proceeding—COMNETSAT 2012: 2012 IEEE International Conference on Communication, Networks and Satellite*, 172–76. doi:10.1109/ComNetSat.2012.6380800.

Vincent, C., B. Fraysse, J. P. Lavieille, E. Truy, O. Sterkers, and F. M. Vaneecloo. 2004. "A Longitudinal Study on Postoperative Hearing Thresholds with the Vibrant Soundbridge Device." *European Archives of Oto-Rhino-Laryngology* 261 (9): 493–96. doi:10.1007/s00405-003-0669-9.

World Health Organization (WHO). 2003. "International Classification of Funcitoning, Disability and Health." *World Health Organization*. http://www.who.int/classifications/icf/icfchecklist.pdf?ua=1.

World Health Organization. 1999. "Future Programme Development for Prevention of Deafness and Hearing Impairment: Report of Second Informal Consultation." http://apps.who.int/iris/bitstream/10665/64994/1/WHO_PBD_PDH_99.6.pdf.

Yanagisawa, T., M. Hirata, Y. Saitoh, H. Kishima, K. Matsushita, T. Goto, R. Fukuma, H. Yokoi, Y. Kamitani, and T. Yoshimine. 2012. "Electrocorticographic Control of a Prosthetic Arm in Paralyzed Patients." *Annals of Neurology* 71 (3): 353–61. doi:10.1002/ana.22613.

Zanin, J., and G. Rance. 2016. "Functional Hearing in the Classroom: Assistive Listening Devices for Students with Hearing Impairment in a Mainstream School Setting." *International Journal of Audiology* 55 (12): 723–29. doi:10.1080/14992027.2016.1225991.

Zhang, Z. K., M. C. Y. Cho, C. W. Wang, C. W. Hsu, C. K. Chen, and S. Shieh. 2014. "IoT Security: Ongoing Challenges and Research Opportunities." In *Proceedings—IEEE 7th International Conference on Service-Oriented Computing and Applications, SOCA 2014*, 230–34. doi:10.1109/SOCA.2014.58.

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

19 Smart Analytical Lab

Subhrodeep Saha, Sourish Sen, Bharti Singh, and Shampa Sen

CONTENTS

19.1 Introduction 313
19.2 Smart Analytics 315
19.2.1 Biosensors 315
19.2.2 Fluorescence Imaging 317
19.2.3 Colorimetric Analysis 318
19.2.3.1 Ultraviolet (UV) Method 318
19.2.4 Spectroscopy 320
19.3 Cloud Computing 321
19.4 IoT in Smart Analysis 322
References 323

19.1 INTRODUCTION

All analytical instruments, from a small pH meter to a large electron microscope, are integral parts of any biotechnological and chemical laboratory. Analytical labs play a major role in diagnostics, research, and development across all fields. Spectroscopy, colorimetric analysis, sensing and imaging techniques are a few analytical techniques having found widespread use in many fields. For example, in the field of medicine, magnetic resonance imaging (MRI) is used to take images of our anatomy without invasive procedures. In environmental sciences, or chemical and biotechnological industries these techniques are used to detect impurity, contamination, and quality and composition of any given sample.

However, the conventional analytical lab has certain challenges such as limited space and high cost of equipment, which limits the number of skills practiced in the lab. Also, handling most of the sophisticated diagnostic instruments requires a high level of expertise and experience, making the process of analysis very time consuming. This results in a long wait for the samples to be analyzed, which can result in the deterioration of sample quality, thus giving false results. Some of these problems can be tackled by a so-called "smart analytical lab."

Smart analytical lab is a concept based on Internet of things (IoT). IoT essentially puts the "smart" in the "smart analytical lab." It has the potential to change the way work is done in a lab by circumventing the challenges and limitations associated with conventional analytical labs (Alaa et al. 2017).

IoT is made up of three main parts:

1. The "things" (the instruments and smartphones)
2. The network connecting them, which allows data sharing
3. The computer systems (the "brain") controlling the flow of data from and to the "things" and interpret/process the data to give us coherent information (Stergiou et al. 2018).

As an important component of the IoT, analytical instruments can serve users effectively by communicating with various digital devices such as computers and smartphones (Alaa et al. 2017). This has become possible because automation in lab instruments has become increasingly sophisticated in recent years permitting them to gather and interchange data across different platforms. This can increase experimental data quality and efficiency, and reduces the time taken for performing experiments and processing data. It also allows experimentation, which generally would be impossible for reasons such as small sample sizes, lengthy procedures, etc. (Stergiou et al. 2018). Due to recent advancements in technology and the introduction of user-friendly operating systems and applications, smartphones have replaced laptops and desktop computers. Taking this fact into account, researchers have designed sensing systems more compatible with smartphones making them handy analytical devices (Kanchi et al. 2018). The processing power of microchips has doubled while their price has halved every few years for the last few decades (Moore's Law); modern smartphones are exponentially more powerful than the desktop computer of 1970s, thus it has become possible to utilize smartphones as handy biosensors and analytical instruments. Although, in most cases, smartphones don't work alone as lab devices, they can be augmented by peripherals, which give the smartphones theirs analytical features (Roda et al. 2016). The main reason behind this is that many sophisticated analytical instruments are not miniaturized enough to fit a smartphone.

The advancement of point-of-care (PoC) instruments, which can be used in nonlaboratory has enabled many tests to be performed at the point of need, outside the lab. A point-of-care instrument must have certain key features—portable, durable, cheap, and energy efficient (can be used on battery). These portable and cheap instruments could be utilized to run routine tests, which are right now performed by trained personnel utilizing research center instruments such as microscopes and spectrophotometers. This would offer huge potential for improving the detection and treatment of diseases, especially in remote regions (Erickson et al. 2014; Xu et al. 2015).

In the ideal version of a connected future, all appliances in a lab would communicate with one another seamlessly; this would require immense computing power and mass storage supported by mobile cloud computing (Alaa et al. 2017; Stergiou et al. 2018). It provides access to data and information from anywhere at any time. For example, integrating a blood glucose meter within a smartphone would help reduce the chances of a diabetic patient forgetting to carry their glucometer with them; it can also remind its users to take timely medication. In addition, with the use of IoT and cloud computing, the smartphone can autonomously store, process, and send blood glucose readings from the phone to a care provider, or cloud service, via the built-in wireless functionality (Bandodkar et al. 2018). Another example of this is

the compact digital lens-free microscopy smartphone. This lightweight and compact microscope could be integrated in a simple smartphone; using lens-free holographic on-chip imaging, the setup can visualize microparticles of various size along with biological samples such as body fluids and microbes (Zhang and Liu 2016).

Further in this chapter, the feasibility and potential applications of a smart analytic laboratory are discussed in detail.

19.2 SMART ANALYTICS

In academia, smart analytics are defined as the process of collecting and analyzing data to help business make better decisions about their products and services. With the introduction of smart analytics in analytical labs, collection and interpretation of data using large sophisticated instruments have become easier and less time consuming. Moreover, advanced mobile phones furnished with high-resolution cameras are exploited to design bioelectronics and biosensors for optical imaging. Currently, with innovations in imaging nanoscale determinations can be achieved by smartphones with conservative gadgets. Following in the footsteps of microscopy imaging, fluorescent and colorimetric imaging have been improved by integration with advanced cameras of smartphones. Mainly, colorimetric analysis with advanced cell phone has been greatly detailed due to its broad potential applications in point-of-care diagnostics. Although smartphones are easy to understand, numerous explanatory systems still require the utilization of substantial and modern supplies.

19.2.1 Biosensors

A biosensor is essentially a device to detect a biological signal (for example urea in urine) and convert it into a measurable electrical signal, thus allowing us to find the "strength" of the biological signal (for example urea concentration). It requires the use of biological substances like nucleic acids or more commonly enzymes and proteins associated with the electrochemical transducers. This setup converts the biological signal into the electrical signal, for example antibody–antigen interaction; here, the antigen acts as the biological detector and the antibody as the analyte. The antigen is associated with the transducer and this method is used for disease diagnosis. A schematic representation of the working of a biosensor has been shown in Figure 19.1. A wide variety of biosensors are being used in the fields of medicine, food and cosmetics industry, and environmental impact assessment to detect and remove adulterants, contaminants, pathogens, impurities, etc. (Ali et al. 2017; Eggenstein et al. 2017).

For example, the glucose biosensor (glucometer) utilizes glucose oxidase to oxidize glucose in the sample, and this glucose reaction generates an electrical signal. The strength of this current is then measured to detect the quantity of glucose in the sample.

Time consuming qualitative and quantitative analysis of biological components can be greatly eased by the advent of "smart" biosensors in modern analytical labs.

Biosensors integrated with a smartphone have been researched through two main approaches where either the smartphone is modified to be a biosensor or the smartphone is used as an interface for the biosensor. These smartphone-based

biosensors have tremendous potential as PoC devices considering they have most of the key features necessary for a PoC device (portable, cheap, energy efficient, etc.). Thus, they have use in health care, food safety, environmental monitoring, and biosecurity; especially in inaccessible and remote areas because a smartphone modified to function as biosensor would be far smaller and more portable than conventional biosensors (Roda et al. 2016). Smartphones can be useful even in cases where the biosensor cannot be integrated into a smartphone due to practical or technological limitations, as it can be used as an interface between the user and the biosensor. The biosensor could wirelessly transmit information to the smartphone and operated remotely, therefore the user does not need to go to the place where the biosensor is installed to take readings. This would be especially useful in inaccessible and/or hazardous zones. A smartphone-based biosensor would also be easy to operate, thus eliminating the need for expertise and laboratory conditions.

FIGURE 19.1 Biosensors.

19.2.2 Fluorescence Imaging

Fluorescence imaging is the process of visualizing molecular and cellular structures by labeling them with fluorescent dyes or markers. This technique has various applications in the fields of biotechnology, organic chemistry, and medicine because of easy implementation, innate sensitivity, inert nature, nonintrusiveness, and the commercial availability of many well-developed fluorescence dyes and labeling schemes for biological studies. A few cases of across the board use of fluorescence imaging are enzyme assays, capillary electrophoresis (CE), antibody detection and screening, bead-based immunoassays, DNA sequencing and fragment sizing or extraction, protein detection, food assay, cellular microscopy, and imaging detection.

The fluorescence microscope used in analytical labs is one of the most common imaging devices utilizing phosphorescence and fluorescence along with reflection and absorption to study properties of organic or inorganic substances. Here, the sample is illuminated with light of a particular wavelength (or wavelengths), which is absorbed by the fluorophores (a chemical compound that absorbs light and emits it as fluorescence), making them emit light of higher wavelengths (the emitted light has lower energy and different color). The light from the source is separated from the much weaker fluorescence of the sample using a spectral emission filter. Its major components are a light source (mercury-vapor lamp or xenon arc lamp are common; more advanced forms are high-power LEDs and lasers), the excitation filter, the dichroic beamsplitter, and the emission filter. The filters and the dichroic beamsplitter are selected to correspond to the spectral excitation and emission spectrum of the fluorophore used to label the sample. In this way, the fluorescence from the labels is imaged to find its distribution at a time. Multicolor images of numerous types of fluorophores are combinations of several single-color images. Epifluorescence microscopes are the most common type of fluorescence microscopes, here excitation of the fluorophore and fluorescence detection are done through the objective (i.e., straight path).

In the biological sciences fluorescence microscopy is a potent device for precise and sensitive staining of a sample to detect the distribution of biomolecules of interest. Numerous fluorescent stains have been produced for indirect labeling of our specimen, for instance nucleic acid stains such as Hoechst and DAPI bind to the minor groove of DNA, thus labeling the cell nuclei.

In smart analytical labs, smartphones can be used to detect fluorescence and obtain the image. Ochratoxin A (OTA) concentration can be measured using a smartphone as a fluorescence device. Here ultraviolet (UV) light is used to excite the sample. A lens is used to focus the fluorescence from the excited sample to the smartphone camera. Finally, the data is transmitted to a personal computer. The fluorescence image data from the smartphone camera is analyzed by the personal computer and represented in its red, green, and blue (RGB) components. OTA is naturally fluorescent and hence when the light is emitted through a solution with OTA present a blue fluorescence is emitted. Thus, the image is captured with the smartphone camera and no fluorescence is observed for a blank solution (Bueno et al. 2016). This technique can be used for various other bio molecules with appropriate excitation and emission spectrum.

19.2.3 Colorimetric Analysis

Colorimetric analysis is used to determine the concentration of the analyte in a colored solution. The colorless analyte can be transformed into a visible color by the addition of a shading reagent. It is significant to both inorganic and natural mixes and might be used by means of enzymatic or nonenzymatic steps. This approach is normally used as a part of therapeutic labs and for mechanical purposes like examinations related to water treatment. The types of apparatus required for colorimetric examination are a colorimeter, an appropriate shading reagent, and some cuvettes. The method might be automated, for example, by using an auto analyzer or by flow infusion examination.

Colorimetric examination can be classified into three types of techniques: enzymatic strategies, nonenzymatic techniques, and ultraviolet techniques.

1. Nonenzymatic techniques: In these techniques, the complex of the test substance with the shading reagent is formed without the use of biocatalysts/ enzymes. For example, calcium and copper are prepared for colorimetric analysis by the methods shown below.

$$\text{Calcium} + \text{o-cresolphthalein complexone} \rightarrow \text{Hued complex}$$

(Ray Sarkar and Chauhan 1967)

$$\text{Copper} + \text{Bathocuproin disulfonate} \rightarrow \text{Hued complex (Zak 1958)}$$

2. Enzymatic techniques: It require the use of an enzymatic catalyst to form a stable complex between the substance to be tested and the shading reagent. Some examples are illustrated below:
   a. *Glucose (GOD-Perid technique)* (Werner et al. 1970)

$$\text{Glucose} + \text{Oxygen} + \text{Water} \xrightarrow{\text{Glucose oxidase}} \text{Gluconate} + \text{Hydrogen Peroxide}$$

$$\text{Hydrogen peroxide} + \text{ABTS} \xrightarrow{\text{Peroxidase}} \text{Colored complex}$$

   b. *Urea* (Fawcett and Scott 1960)

$$\text{Urea} + \text{Water} \xrightarrow{\text{Urease}} \text{Ammonium carbonate}$$

$$\text{Ammonium carbonate} + \text{Phenol} + \text{Hypochlorite} \rightarrow \text{Coloured complex}$$

19.2.3.1 Ultraviolet (UV) Method

No visible change in color is observed with the UV method, but the mechanism is exactly the same, that is, the measurement of an alteration in the absorbance of the solution. UV methods generally measure the disparity in absorbance at the 340 nm wavelength between nicotinamide adenine dinucleotide (NAD) and its reduced form (NADH).

However, these strategies are less exact and tedious. A versatile location framework, named Smart Forensic Phone, has been created, has empowered us to quickly and definitely perform colorimetric investigations of a bloodstain for age estimation. In this strategy, blood is set on five distinct materials (backdrop, texture, glass, wood, and A4 paper), the RGB values per pixel of the bloodstain picture at 6-h interims are observed, and the inexact age of the bloodstain was evaluated utilizing a cell phone application. The RGB values are changed over into the V estimation of HSV (tint, immersion, and brilliance). A quick decrease in RGB and V values is seen over the initial 42 h and from that point remains moderately unaltered. The age of the bloodstain is determined from the plot of V versus time. This offers a novel technique to estimate the time since the victim's demise technique (Kanchi et al. 2018).

Additionally, an advanced cell phone-based optical platform is utilized for colorimetric investigations of blood utilizing a dispensable miniaturized scale fluidic gadget in an advanced scientific lab. Using an incorporated camera in a personal digital assistant (PDA), pictures of human blood are taken in the smaller scale channel and broken down by a versatile application. To avoid image burning and ambient light effect, a special light-diffusing system consisting of a microfluidic device enclosed inside a white acrylic box is used. With the picture software program on the PDA, the created gadget is effectively connected to decide different haematocrit levels of human blood from 10% to 65% (Kim et al. 2017).

Also, a quick and direct technique is being produced for colorimetric detection of smelling salts, which release ammonia, utilizing advanced mobile phones. This system depends on the control of surface plasmon bands of silver nanoparticles (AgNPs) by forming $Ag(NH_3)^{2+}$ complex. Hence, there is a reduction in AgNP concentration, and thus, the color of the suspension diminishes. The variation in color intensity can not only be measured by the standard UV-Vis spectrometer but it can also be done using a smartphone by RGB analysis. This was verified by detecting ammonia in the range of 10–1000 mg L^{-1}, by UV-Vis spectrometry and the developed smartphone colorimetric technique. It was seen that the cell phone-based colorimetric technique can be used as reliably as spectrophotometer for quantification of ammonia in the sample (Amirjani and Fatmehsari 2018).

With the specific goal of understanding the location and visual recognition of antimicrobial compounds, a PoC Testing technique is built by utilizing computerized picture colorimetry based on advanced smartphone. To test this, streptomycin aptamer was taken as the analyte model of antitoxins. The streptomycin aptamer interacted with the analyte, and the unreacted aptamer was hybridized with the correlative DNA to form dsDNA. SYBR Green I was joined with the dsDNA to emit clear green fluorescence. The fluorescence diminishes with the increase of streptomycin fixation. This was detected once again by using the smartphone-based technique. In this example, the phone camera takes the pictures of the fluorescence and the Touch Color application is used to perform the RGB estimations of the pictures. This strategy has the upsideof good selectivity, straightforward operation, and on location representation (Lin et al. 2018).

19.2.4 Spectroscopy

Spectroscopy is the study of the interaction between electromagnetic radiation and matter. Spectral measurement devices are otherwise called spectrometers, spectrophotometers, spectrographs, or spectral analyzers. To put it plainly, the succession of events in an advanced spectrophotometer is as follows:

1. The beam of light is passed through a monochromator, which diffracts it into its components and allows only the required monochromatic light to pass through it, and split into two beams. It is then incident over the sample and the reference arrangements.
2. Only a portion of the incident wavelengths reflects from, or transmits through the sample and the reference.
3. The resultant light strikes the photo detector, which extrapolates the relative power of the two beams.
4. Computerized systems convert the relative currents into absorbance values, which allow us to determine the sample concentration.

These days handheld mobile phone-based spectrometers have been created, which work in both assimilation and discharge modes. The device, named Spectrophone, includes modules made in medium-density fibreboard (MDF), an embedded light source intended for retention mode, a digital video disc (DVD) for the diffraction grinding, and a cell phone to process the picture information obtained. No significant difference was observed when compared to the values obtained from conventional measurement methods. The figures of merit of the Spectrophone are decent, with cutoff points of measurement of 70 g/L for absorption and 60 g/L for emission modes, while showing a high linearity ($R^2 > 0.9995$). High precision, simplicity, and an ease of use of the application, while requiring minimal efforts make this handheld device an attractive option for absorption/emission estimations (de Oliveira et al. 2017).

An advanced mobile phone-based spectrometer design, which is independent and held on a remote stage can be utilized for quick, nondamaging testing of organic products. This gadget is inherently low-cost and its power consumption is negligible, making it convenient to do a range of studies in the field. Every single basic component of the device like the light source, spectrometer, channels, microcontroller, and remote circuits are integrated into a device of small dimensions ($88 \times 37 \times 22$ mm) weighing only 48 g. The resolution of the spectrometer is 15 nm, conveying exact and repeatable estimations. The device has a dedicated application interface on the PDA to receive, communicate, analyze, and plot spectral data. The execution of the advanced cell spectrometer is practically identical to existing bench top spectrometers regarding soundness and wavelength determination. Validations of the device are carried out by exhibiting nondangerous readiness testing in organic product tests. During the ripening process, ultra-violet (UV) fluorescence from chlorophyll present in the skin are measured across various apple varieties and correlated with destructive firmness tests. An acceptable correlation is seen among ripeness and fluorescence signals (Das et al. 2016).

19.3 CLOUD COMPUTING

Cloud computing is a system which allows wireless access to a collective pool of electronic resources, as demonstrated in Figure 19.2. These computing services can be easily managed and released with minimal effort or human interaction. It has brought a paradigmatic shift in providing data storage as well as computing resources (Ravi et al. 2018).

In simple terms, cloud computing basically provides access to data and information from anywhere at any time, all the while reducing the need for bulky hardware. More specifically, mobile cloud computing can be defined as the incorporation of mobile devices (such as smartphones) with cloud computing technology in order to make these devices more resourceful by increasing their storage capacity, computational power, energy efficiency, and context awareness ("smartness") (Stergiou et al. 2018).

The modern smartphone and smart analytics necessitate the use of cloud computing to manage their data and resources. Cloud computing has become a versatile and powerful platform to achieve large-scale and complex computing. Some of the benefits of cloud computing can be listed as scalable data storage, virtualized resources, data/information security, and parallel processing. Virtualized resources saves space, parallel processing reduces time consumption, data security is essential in an easily accessible network, and scalable data storage makes sure that storage space is plentiful.

FIGURE 19.2 Interconnected devices in a smart lab.

FIGURE 19.3 Virtualization in cloud computing.

Apart from the previously mentioned advantages, cloud computing helps a smart lab meet major challenges faced by a conventional analytical lab by increasing efficiency and decreasing operational cost by restricting its owned software and hardware, as cloud services can be readily outsourced (Ravi et al. 2018). Virtualization is the key technology behind cloud computing, it makes cloud computing possible. It is a method of logically dividing the computer's processing power to allow for multiple operations or applications to run on a single system. Before the introduction of virtualization, computers could only perform a single application or operation at a time (Graziano 2011).

Virtualization can be utilized and provisioned to execute computing tasks such as data analysis and operation of automated instruments; using this method a single system can operate multiple instruments (Figure 19.3). Thus, it provides the versatility required to speed up operations, allowing for exponentially larger sample sizes and cutting costs by increasing instrument utilization.

Autonomic computing working in tandem with virtualization automates the process through which the user can divide the virtualized resources proportionally to the requirements. By decreasing human interference, automation speeds up the whole process, and reduces the likelihood of human errors and cuts labor costs (Hamdaqa and Tahvildari 2012). This would allow users to benefit from the assistance of all of the analytical instruments without the need of the in-depth knowledge, expertise, and proficiency required with each one of them, which would not be possible in a traditional lab. The user of a smart analytical lab will not need to compromise because of the lack of analytical instruments.

19.4 IoT IN SMART ANALYSIS

The twenty-first century is the era of IoT, which has become greatly intertwined in our lives in the form smartphones and other smart devices. Generally speaking, IoT refers to the communication between everyday electronic objects, which are usually

equipped with ubiquitous intelligence; in case of smart analytical labs the "objects with ubiquitous intelligence" are the computers, smartphones, and smart analytical lab equipment. IoT increases the omnipresence of the Internet by connecting all the objects to it via embedded (rooted) systems (as shown in Figure 19.3), which forms a widespread web of devices interacting with its users as well as other devices (León et al. 2010). This concept can be essentially used in a smart analytical lab where computers, smartphones, and analytical instruments would form a network and complement each other and their user; labs in different cities could share their data and resources easily, which would greatly aid and improve the quality and quantity of research. The IoT can connect all the instruments in the lab and turn it into a "smart," cohesive, and coherent system, which can function with minimal intervention and caters to the users' needs.

IoT and cloud computing are two sides of the same coin; cloud computing requires and enables IoT, as IoT is basically a network of devices, which is the backbone of cloud computing. It enables communication between different elements of the smart analytical lab and provides an easy interface for users to operate the analytical instruments; this eliminates the need for specialized expertise and experience, which is a limitation in a conventional analytical lab. Using IoT it would be possible to operate all analytical lab instruments, small to big, simple to complex, using specialized applications present in our smartphones.

The ideal smart analytical lab would essentially be an environment with minimal manual interference; samples would be analyzed and data would be obtained and interpreted automatically. Due to the connectivity provided by IoT, lab resources could be accessed and used remotely. The smartphone would essentially be a PDA, which would act as a guide, doorway, or interface between the user and the smart analytical lab.

REFERENCES

Alaa, M., A. A. Zaidan, B. B. Zaidan, M. Talal, and M. L. M. Kiah. 2017. "A review of smart home applications based on internet of things." *Journal of Network and Computer Applications*, 97, 48–65. Elsevier Ltd. https://doi.org/10.1016/j.jnca.2017.08.017.

Ali, J., J. Najeeb, M. A. Ali, M. F. Aslam, and A. Raza. 2017. "Biosensors: Their fundamentals, designs, types and most recent impactful applications: A review." *Journal of Biosensors & Bioelectronics*, 8(1), 1–9. https://doi.org/10.4172/2155-6210.1000235.

Amirjani, A. and D. H. Fatmehsari. 2018. "Colorimetric detection of ammonia using smartphones based on localized surface plasmon resonance of silver nanoparticles." *Talanta*, 176, 242–246.

Bandodkar, A. J., S. Imani, R. Nuñez-Flores, R. Kumar, C. Wang, A. M. Vinu Mohan, J. Wang, and P. P. Mercier. 2018. "Re-usable electrochemical glucose sensors integrated into a smartphone platform." *Biosensors and Bioelectronics*, 101, 181–187 (July 2017). Elsevier B.V. https://doi.org/10.1016/j.bios.2017.10.019.

Bueno, D., R. Muñoz, and J. L. Marty. 2016. "Fluorescence analyzer based on smartphone camera and wireless for detection of Ochratoxin A." *Sensors and Actuators B: Chemical*, 232, 462–468.

Das, A. J., A. Wahi, I. Kothari, and R. Raskar. 2016. "Ultra-portable, wireless smartphone spectrometer for rapid, non-destructive testing of fruit ripeness." *Scientific reports*, 6, 32504.

de Oliveira, H. J. S., P. L. de Almeida, B. A. Sampaio, J. P. A. Fernandes, O. D. Pessoa-Neto, E. A. de Lima, and L. F. de Almeida. 2017. "A handheld smartphone-controlled spectrophotometer based on hue to wavelength conversion for molecular absorption and emission measurements." *Sensors and Actuators B: Chemical*, 238, 1084–1091.

Eggenstein, C., M. Borchardt, C. Diekmann, B. Gründig, C. Dumschat, K. Cammann, M. Knoll, and F. Spener. 2017. "A disposable biosensor for urea determination in blood based on an ammonium-sensitive transducer." *Biosensors & Bioelectronics*, 14, 33–41 (January 01). Accessed January 06, 2018. https://www.ncbi.nlm.nih.gov/pubmed/10028647.

Erickson, D., D. O'Dell, L. Jiang, V. Oncescu, A. Gumus, S. Lee, M. Mancuso, and S. Mehta. 2014. "Smartphone technology can be transformative to the deployment of lab-on-chip diagnostics." *Lab on a chip*, 14, 3159–3164 (September 07). https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4117816/.

Fawcett, J. K. and J. E. Scott. 1960. "A rapid and precise method for the determination of urea." *Journal of Clinical Pathology*, 13(2), 156–159. https://doi.org/10.1136/jcp.13.2.156.

Graziano, C. D. 2011. "A Performance Analysis of Xen and KVM Hypervisors for Hosting the Xen Worlds Project." *Master's Thesis*. http://lib.dr.iastate.edu/etd/12215/.

Hamdaqa, M. and L. Tahvildari. 2012. "Cloud computing uncovered: A research landscape." *Advances in Computers*, 86, 41–85. doi:10.1016/b978-0-12-396535-6.00002-8.

Kanchi, S., M. I. Sabela, P. S. Mdluli, Inamuddin, and K. Bisetty. 2018. "Smartphone based bioanalytical and diagnosis applications: A review." *Biosensors and Bioelectronics*, 102, 136–149 (August 2017). Elsevier B.V. https://doi.org/10.1016/j.bios.2017.11.021.

Kim, S. C., U. M. Jalal, S. B. Im, S. Ko, and J. S. Shim. 2017. "A smartphone-based optical platform for colorimetric analysis of microfluidic device." *Sensors and Actuators B: Chemical*, 239, 52–59.

León, O., J. Hernández-Serrano, and M. Soriano. 2010. "Securing cognitive radio networks." *International Journal of Communication Systems*, 23(5), 633–652. https://doi.org/10.1002/dac.

Lin, B., Y. Yu, Y. Cao, M. Guo, D. Zhu, J. Dai, and M. Zheng. 2018. "Point-of-care testing for streptomycin based on aptamer recognizing and digital image colorimetry by smartphone." *Biosensors and Bioelectronics*, 100, 482–489.

Ravi, K., Y. Khandelwal, B. S. Krishna, and V. Ravi. 2018. "Analytics in/for cloud-an interdependence: A review." *Journal of Network and Computer Applications*, 102, 17–37 (August 2017). https://doi.org/10.1016/j.jnca.2017.11.006.

Ray Sarkar, B. C. and U. P. S. Chauhan. 1967. "A new method for determining micro quantities of calcium in biological materials." *Analytical Biochemistry*, 20(1), 155–166. https://doi.org/10.1016/0003-2697(67)90273-4.

Roda, A., E. Michelini, M. Zangheri, M. Di Fusco, D. Calabria, and P. Simoni. 2016. "Smartphone-based biosensors: A critical review and perspectives." *TrAC—Trends in Analytical Chemistry* 79, 317–25. Elsevier B.V. https://doi.org/10.1016/j.trac.2015.10.019.

Stergiou, C., K. E. Psannis, B. G. Kim, and B. Gupta. 2018. "Secure integration of IoT and cloud computing." *Future Generation Computer Systems* 78, 964–75. Elsevier B.V. https://doi.org/10.1016/j.future.2016.11.031.

Werner, W., H. G. Rey, and Wielinger, H. 1970. Über die Eigenschaften eines neuen Chromogens für die Blutzuckerbestimmung nach der GOD/POD-Methode. *Fresenius' Zeitschrift Für Analytische Chemie*, 252(2–3), 224–228. https://doi.org/10.1007/BF00546391.

Xu, X., A. Akay, H. Wei, S. Wang, B. Pingguan-Murphy, B. E. Erlandsson, X. Li et al. 2015. "Advances in smartphone-based point-of-care diagnostics." *Proceedings of the IEEE*. July 15.

Zak, B. 1958. "Simple procedure for the single sample determination of serum copper and iron." *Clinica Chimica Acta*, 3(4), 328–334. https://doi.org/10.1016/0009-8981(58)90021-4.

Zhang, D. and Q. Liu. 2016. "Biosensors and bioelectronics on smartphone for portable biochemical detection." *Biosensors and Bioelectronics* 75, 273–284. Elsevier. https://doi.org/10.1016/j.bios.2015.08.037.

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

20 Crop and Animal Farming IoT (CAF-IoT)

Neha Agnihotri, Soumyadipto Santra, and Shampa Sen

CONTENTS

20.1 Introduction 328
20.2 IoT in Crop Farming 328
20.2.1 Traditional Farming Practices 328
20.2.2 Modern Farming Technologies 329
20.2.2.1 Problems with Traditional Farming Practices 329
20.2.2.2 What is Modern Farming Technology 329
20.2.3 IoT and Smart Agriculture 329
20.2.3.1 Soil Moisture Sensor 330
20.2.3.2 PIR Sensors 330
20.2.3.3 Arduino 331
20.2.3.4 IoT-Based Smart Farming Stick Using Arduino and Cloud Computing 331
20.3 IoT for Animal Husbandry 331
20.3.1 Traditional Animal Husbandry 331
20.3.1.1 Introduction 331
20.3.1.2 Husbandry 332
20.3.2 Technological Developments 334
20.3.2.1 Geo-Informatics Technologies 334
20.3.2.2 Remote Sensing 334
20.3.2.3 ShearEzy 334
20.3.2.4 Information Technology in Molecular Biology 335
20.3.2.5 Information Technology in Simulation Studies 335
20.3.2.6 Sperm Analyzer (Aidmics Biotechnology) 335
20.3.3 Automation and IoT 336
20.3.3.1 Biosensors 336
20.3.3.2 IoT-Based Beef Cattle Breeding System 338
20.4 Conclusion 339
References 339

20.1 INTRODUCTION

The internet of things (IoT) is an innovation which gathers and trades information from physical things utilizing electronic sensors and the Internet. The (IoT) innovation is presently enabling ranchers to interface with devices on the web to enhance agriculture.

The Agriculture industry must defeat expanding water scarcity, constrained accessibility of fields, increasing costs, while meeting the expanding utilization needs of a worldwide population estimated to increase 70% by 2050 (FAO 2016).

While applying IoT to the farming industry, sensors collect billions of readings on ecological information such as temperature, moisture, drinking water, sustaining rates, CO_2 fixation, sodicity, and pH levels. This information gathered will have applications with regards to agricultural administration, profitability, and productivity, while even filling in as a ready framework for agriculturists.

Apart from farming, IoT innovations can significantly facilitate animal husbandry, where ranchers can now insert web-associated sensors inside their animals without any discomfort (Neethirajan 2017). Utilizing data from these sensors, agriculturists can screen their general health parameters and help farmers locate their animals and check for any imminent danger to them.

IoT innovations enables ranchers to screen their gear. Sensors can be incorporated into tractors to decide whether it is running proficiently. A notice will be sent to the rancher before repairs should be made. This will help avoid sudden breakdowns enabling it to remain in the field longer to expand efficiency. Adopting a proactive strategy can enable hardware to be utilized longer and decrease costs.

20.2 IoT IN CROP FARMING

20.2.1 Traditional Farming Practices

After the two world wars, customary agribusiness, prospered around the globe and became the most rehearsed type of agriculture as science and technology progressed simultaneously. Artificial chemical products are used to treat the soil and plants and are in general harmful to the ecosystem. Their purpose is to help prevent diseases or bugs in plant and form the backbone of customary farming.

The outcome is that these chemicals—and the same applies to items utilized as a part of natural agriculture—end up in our food. It has been proven these substances gather in our tissues and when their levels are high they increase the incidence of diseases that is like cancers, as these chemicals can act as carcinogens.

The examination found that agriculturists are quite happy to give up conventional cultivating techniques and move to better sustainable and fruitful techniques. Customary cultivating devices, for example, the techniques to get land ready for agriculture purpose. Kraal compost is the principle soil ripeness technique sought after by agriculturists.

20.2.2 Modern Farming Technologies

20.2.2.1 Problems with Traditional Farming Practices

Indian agricultural sector is in a troublesome state because of the absence of motorization and the shortage of mechanical progress. The agriculture in India

- Needs significant specialized learning
- Produces from low grades of seeds
- Does not utilize composts and pesticides judiciously
- Needs a satisfactory water system foundation
- Needs adequate credit of capital for development

Alongside this, the modern agricultural practices like crop rotation are not acknowledged in India. Majority of Indian ranchers plant one crop in the same field for a considerably long time. This prompts exhaustion of the soil with a consequential reduction in harvest yield. The goal of current farming practices is to keep up the richness of the soil throughout the year.

20.2.2.2 What is Modern Farming Technology

Current cultivating innovations are aimed to enhance traditional cultivating practices. It makes utilization of best quality seeds, better water management systems, improved modern equipments, composts and pesticides.

Present day agrarian practices utilize automated hardware for watering systems, working, and collecting alongside crossbred seeds. It also involves the incorporation of big data for smart farming (Wolfert et al. 2017). Majority of India's population depends on farming as a means of living, hence any innovations in this sector will impact the economy of the country. The agricultural land in India are distributed sparsely, thus making communication and subsequent automation troublesome.

On the other hand, land for modern agriculture is being consolidated into larger farms. This leads to easy access for mechanized equipment and the use of hybrid seeds for increased productivity and disease resistance (Schmitz and Moss 2015). The capital invested behind present day agribusiness is significantly more than what is invested in conventional cultivating advancements. Current agriculture picks up from endowments on vitality, water system, seeds and manures.

Modern farming has developed with innovations such as irrigation via tube wells, sprinklers, and dribbling systems. Modern agriculture is far more advanced and big in terms of capacity due to the use of tractors, motorized hardware for working, furrowing and reaping. Correspondingly, with the development of better temperature and moisture controlled storage rooms, it is now possible to shield from water, creepy crawly bugs and also provides additional warmth.

20.2.3 IoT and Smart Agriculture

Agriculture assumes an imperative role in the advancement of the nation. In India, around 70% of the population relies on cultivating, and 33% of the country's capital

originates from cultivation. Problems concerning agriculture have continually hindered the improvement of the nation. The solution to these issues is modernizing current conventional agribusiness techniques. A schematic representation of the working of agricultural IoT has been demonstrated in Figure 20.1.

IoT has many applications in agriculture, some of which are mentioned below:

- *Managing pest*: Frequently, the ranchers hard work is annihilated by predators or vermin bringing about enormous misfortune to the farmers. To anticipate these circumstance, IoT enables recognizing the movement of predators or vermin utilizing passive infrared (PIR) sensors. This data can be utilized by the ranchers to prevent destruction by predators.
- *Managing crop water*: Farmers need water to irrigate the crop. This may bring about wastage of water and possible harm to the harvests as well. Therefore, IoT can play an essential role by sending warnings to agriculturists if there should be too much or too little irrigation.

20.2.3.1 Soil Moisture Sensor

A sensor that detects the dampness in the soil, is called a soil moisture sensor. It sends a warning message to the rancher in order to keep the land and harvest from being harmed (Eller and Denoth 1996).

20.2.3.2 PIR Sensors

PIR stands for passive infrared sensor. A PIR-based movement finder is utilized to detect the movements of predators, creatures, people, or different things. It would keep away vermin from causing damage to the crops or herds. Thus, installing these sensors in the field can reduce the need for farmers to constantly monitor them in order to maximize production. But the limitation is here is, PIR sensor cannot be used on all soil types hence the soil needs to be first assessed before using (Frankiewicz and Cupek 2013).

FIGURE 20.1 Flowchart of the agricultural IoT.

FIGURE 20.2 Block diagram of Arduino.

20.2.3.3 Arduino

Both soil moisture sensors and PIR sensors are associated with Arduino to to carry out the activity (Figure 20.2). Arduino will send the information to the database utilizing an ethernet shield, and if there is a crisis it will likewise send a message to the client utilizing a GSM module.

20.2.3.4 IoT-Based Smart Farming Stick Using Arduino and Cloud Computing

In view of IoT (Internet of things) innovation, it is a cultivating stick which has filled up the flaws of traditional agriculture and saved farmers lives. The aim is to build a novel IoT-based agribusiness stick able to collect live data (temperature, soil moisture, and smoke detection) for effective condition sensing, which will help them to do smarter cultivating and increase yields. The agriculture stick also incorporates the Arduino innovative motherboard fitted with different sensors. Thus, this stick combines all the features essential to protect the land as well as harvest. If we look into its make-up it combines the qualities of a soil moisture sensor, a PIR sensor, etc. thus maintaining the overall expense and reducing the load on farmers and machines (Moghe, Lambert, and Divan 2012).

20.3 IoT FOR ANIMAL HUSBANDRY

20.3.1 Traditional Animal Husbandry

20.3.1.1 Introduction

Animal husbandry concerns raising animals for meat, fiber, dairy, eggs, and so on. It incorporates everyday care, specific rearing, and the raising (with proper care) of animals. The history of domesticated animals begins around 13,000 BC during the Neolithic, preceding the cultivation of the first harvests. An extensive variety of different species, such as the horse, water bison, lama, rabbit, and guinea pig are utilized as domesticated animals around the world. The aquaculture of fish, mollusks, and the keeping of honeybees or silkworms is boundless. Insect raising for human consumption takes place in several nations; for instance, cricket farming is a productive industry in Thailand.

Modern animal husbandry relies on production system adapted to the type of land available. Subsistence farming is now being superseded by intensive animal

farming, where for instance meat dairy cattle are kept in high density feedlots, and large numbers of chickens might be brought up in broiler houses or batteries (Kuhnen et al. 2015). On poorer soil, for example in uplands, kept in semi-captivity,and are allowed to wander to feed themselves.

Most domesticated animals are herbivores. Ruminants like dairy cattle and sheep are adapted to feed upon grass, and often also graze outside. Pigs and poultry can't process the cellulose in vegetation, and thus require grains and other high-vitality nourishments.

20.3.1.2 Husbandry

20.3.1.2.1 Systems

Traditional animal husbandry was a source of sustenance for the farmer and his/her family. Killing animals for nourishment was a secondary option, and whenever possible its fleece, eggs, milk, and blood were reaped while the creature was still alive. In the customary cycle of transhumance, individuals and animals moved between summer and winter pastures; the midyear pastures were up in the mountains, and the winter rangelands were in the valleys (Blench 2001). Animals can be kept in free ranges or in intensive farming structures. Broad frameworks includes extensive system that involves animals roaming at will, or under the supervision of herdsman. Farming in western United States includes animals roaming freely at will or under the supervision of herdsmen in open as well as private lands (Starrs 2000).

20.3.1.2.2 Feeding

Herbivores are typically classified into "think selectors" (who specifically eat seeds, leafy foods, and nutritious young foliage), "slow eaters" who mostly eat grass, and "middle of the road feeders" who feed from an entire scope of accessible plant materials. Cows, sheep, goats, deer, and impalas are ruminants; they process sustenance in two stages, biting and gulping in the ordinary way, and afterward spewing the semidigested cud to bite it again and in this way they remove the utmost nourishment value (Science and Dryden 2008).

The dietary needs of these animals are for the most part met by eating grass. Grasses develop from the base of the leaf-sharp edge, enabling it to flourish even when vigorously nibbled or cut ("The Living Planet: A Portrait of The Earth by David Attenborough," n.d.).

In many climates grass growth is seasonal, for instance in the calm summer or tropical stormy season. Other scavenge crops develop as well and a significant number of these, and also crop residues, can be used to provide for the nutritious needs of domesticated animals in the lean season. Animals raised on extended ranges may subsist completely on grazing, yet more intensively kept domesticated animals will require more vigorous and protein-rich nourishments. Nutrition is for the most part found in grains and cereal by products, fats and oils, and sugar-rich nourishments, while proteins may originate from fish or meat, dairy items, vegetables, and other plants, frequently the products of vegetable oil extraction (Science and Dryden 2008). Since pigs and poultry are nonruminants they are unfit to process the cellulose in grass and different rummages, so they are fed completely

on grains and other high-nutrition foodstuffs. The elements for the animals' feed can be procured on the homestead or can be purchased, as commercially available feeds are targeted for the distinctive classes of domesticated animals, their developmental stages, and their particular nourishing necessities. Vitamins and minerals are added accordingly to adjust their diet. Farmed fishes are usually fed pelleted food.

20.3.1.2.3 Breeding

The breeding of ranch animals seldom occurs spontaneously but is managed by farmers with a view of encouraging certain traits that are seen as desirable. These incorporate resistance, productivity, breeding capacities, quick development rates, low bolster (feed) utilization per unit of development, better body features, higher yields, better fiber qualities, and other attributes. Unwanted attributes are, for example, health defects, forcefulness or absence of submission are chosen against (Rodenburg, Breeding, and Centre, n.d.).

Rearing in particular has seen huge increases in efficiency. In 2007, a run of the mill two-month-old oven chicken was 4.8 times larger than in 1957. In the 30 years prior to 2007, the normal yield of a dairy bovine in the United States almost doubled (Rodenburg and Turner 2015). Techniques, such as manual sperm injection and fetus exchange are commonly used today, not just as strategies to ensure females breed routinely, but additionally to help enhance breed hereditary qualities.

20.3.1.2.4 Animal Health

Great cultivation, appropriate feeding, and cleanliness are the fundamental principles of animal wellbeing on the homestead, bringing monetary advantages through increased production. When, despite these safety measures animals become sick they are treated with veterinary pharmaceuticals by the agriculturist and the veterinarian. In the European Union, when agriculturists treat their own animals they are required to follow treatment guidelines and to keep records of the medicines given (Kools, Moltmann, and Knacker 2008). Herds are vulnerable to various diseases and conditions, which may influence their wellbeing. A few known swine fever and scrapie are particular to one sort of stock, while others, similar to foot-and-mouth illness influence all cloven-hoofed animals. For this very reason the body temperature of the animals must be measured and monitored closely (Sellier, Guettier, and Staub 2014). Governments are especially worried about zoonoses—ailments people may contract from animals. Wild animals may harbor diseases able to infect domesticated animals, which may be due to inadequate biosecurity. An episode of Nipah infection, in Malaysia in 1999, was traced back to pigs ending up sick coming in contact with organic product eating flying foxes (*Pteropus hypomelanus and Petropus vampyrus*). The pigs thus passed on the contamination to humans. Avian influenza H5N1 is present in wild bird populations and can be conveyed over vast distances by migrating birds. This infection is highly transmissible to local poultry and people living close to them. Different contagious illnesses affect wild creatures, domesticated animals, and people such as rabies, leptospirosis, brucellosis, tuberculosis, and trichinosis.

20.3.2 Technological Developments

Domesticated animals revenues are invaluable for the wealth of a nation like India. In India, animal farming is no longer a side-revenue to agriculture or a assistance to farming (using cattles to plow the field). Animal farming has transformed into an industry and the most recent reports state that its contribution to the GDP of the country is significantly higher than ever before. Animal husbandry offers a superior degree of financial security for small ranchers whose wages from agriculture suffer because of storms, loss of landholdings, insects, poor planning, and so forth. Despite the fact that the development of the domesticated animal industry is exceptionally encouraging, keeping in mind the end goal to make India a worldwide leader in animal farming, it is logical to integrate in this industry the advances made in different fields. The advancements in Information Technology over the course of recent decades are colossal and offer extraordinary potential in enhancing animal productivity through different measures like eradication of dangerous illnesses, quick and precise infection detection, and so on.

20.3.2.1 Geo-Informatics Technologies

Geographic Information System (GIS) (QGIS Development Team 2015) is an electronic database administration system for capturing, storing, checking, coordinating, controlling, dissecting, and showing information tied to a geographical area. Early identification of an irresistible sickness episode is a vital initial move towards actualizing powerful ailment mediations and diminishing coming about mortality and dismalness. Both topographical and regular spreads of numerous irresistible maladies through the atmosphere, has evoked the use of occasional atmosphere figures as prescient pointers in ailment early cautioning frameworks (EWS) for quite some time. GIS, Remote Sensing (RS), and GPS are the three commonly utilized veterinary geo-informatics tools utilized with this type of data for fast global mapping of information to enable management of animal infections.

20.3.2.2 Remote Sensing

RS has changed the way researchers handle and break down geographic information. Remote detecting alludes to the securing of geographic information without reaching the zone of study. Today, remote detecting by and large rests on satellite pictures and airborne photography/imaging, both of which are sensitive to portions of the electromagnetic spectrum not noticeable to the human eye or to most cameras (e.g., infrared, small-scale waves) (Toth and Jóźków 2016). The capacity to determine otherwise undetectable vitality and to record precise measurements remotely, shows the capability of the apparatus.

20.3.2.3 ShearEzy

The ShearEzy is a result of Australian Wool Innovation (AWI) interest in the Upright Posture Shearing Platform (UPSP) innovations. The product turned out to be financially accessible in late 2007, with the latest adaptation released in June 2012. ShearEzy's benefits include:

- Diminished physical exertion and strain from shearing.
- Can be more beneficial than "on the board" shearing, particularly for the novice or normal shearer.
- Simpler blow patterns and effective animal restraints.
- Simple detachment of crutching's and lower quality fleeces.

Many sheep and rams are effortlessly and safely taken care of without sedation. ShearEzyis an air-worked system, comprising of a shearing stage with an articulated parallel sheep loader. Sheep stroll into the loader on a slanted, nonreinforced incline, and are kept there until prepared to be shorn (De Boos 2001).

20.3.2.4 Information Technology in Molecular Biology

A standout among the most vital examinations for animal wellbeing is the examination of their genomes. Results of genomic examination are always correct and appropriate and thus helps in better understanding of the disease. To create quality tests, the whole genome must be mapped. After mapping, the unique areas have to be identified comparing it with the genome of other infectious agents. Despite the fact that the genome is little in measure, the quantity of bases in every genome is past manual correlation. Numerous products are accessible that can contrast the genome and existing genomes in a quality bank to recognize special territories. Such programming additionally gives degree to building up a phylogenetic tree, atomic clock and for creating preliminaries for PCR response. The phylogenetic tree and sub-atomic clock are basic for sub-atomic scientists to land at conceivable sources of the infection. This discovering acquires essentialness if the infection strains are immunologically particular.

20.3.2.5 Information Technology in Simulation Studies

Another area where IT can contribute significantly is the reconstruction of in vivo conditions. Certain advanced graphics based softwares offer best solution for this. Trafficking of etiological operators, proteins and so forth in the middle of the cells can be considered utilizing this product. Concentrate on these trafficking designs is fundamental to create immunizations for cell related infections and intra cell microorganisms. The part of designs based programming in concentrate the three dimensional structures of antigen and immune response is additionally imperative to dissect the cooperation's between them in vivo. Cooperation of antigen and immune response is fundamental to think about the pathogenesis of any irresistible ailments.

20.3.2.6 Sperm Analyzer (Aidmics Biotechnology)

The iSperm analyzer enables animal husbandry specialists to complete swine sperm assessment quickly and decrease their costs of equipment by about 90%. The iSperm can associate with cell phones and laborers can examine the sperm's morphology, quantity, and motile rating. The information can likewise be stored to enable breeders to keep records and build up a precise database.

Thus, as compared with the traditional process of sperm collection (checking motility rate, concentration, and manually recording all the data), which is time consuming, tedious, and requires labor, this sperm analyzer reduces the overall time spent and also skips the step of manual recording by replacing it with an

automatic recording system. Thus, this can be immediately followed by the artificial insemination if desired.

20.3.3 Automation and IoT

20.3.3.1 Biosensors

Early identification of sicknesses utilizing biosensors enables can prevent their death or spread the illness thus, decreasing the impact of the illness and related social and financial consequences. In order to diagnose a disease early in animals we need a continuous monitoring biosensor to predict the disease at early stages (Diamond, Zhao, and Li 2017).

The utilization of biosensors and wearable innovations is becoming progressively imperative for animal wellbeing management. These devices, if fabricated correctly and utilized effectively, can determine infections in the long run and diminish their financial impact. Such devices are especially valuable for dairy cows and poultry ranches. Rather than depending exclusively on the ranchers' faculties and knowledge, nearby sensors can give solid information about the physical state of the herds. Because of the unrivaled progresses of wearables and sensors, they can produce a leap forward in domesticated animals care, and are guaranteed to be one the most impactful and practicable innovations in the animal farming market. New wearable advancements are being aimed at addressing the issues of pets and farm animals. Items such as solution patches, tracking collars, and sitting posture corrector are being acquired at higher rates and outfit for the more advantageous childhood of homestead creatures. These wearable advances are multifunctional and productive, enabling animal owners to accomplish more in less time. Worldwide development of this sector—in the next 10 years—has been anticipated to take off from $0.91 billion to $2.6 billion.

Sensor and wearable innovations can be embedded on animals to analyze their sweat constituents (McCaul et al. 2017; Luong, Male, and Glennon 2008), measure their body, monitor their behaviors (Sellier, Guettier, and Staub 2014), and develop characteristic sounds (Lee et al. 2015; Kucharzyk et al. 2015), identify pH (Zhang et al. 2017), avert illness, recognize analytes, and recognize the presence of infections and pathogens. Wearable sensors enable ranchers to indentify ailments early, and avoid spread to other animals. Agriculturists can likewise separate sick creatures so as to keep the spread of an ailment contained. Aside from gathering helpful information on animal wellbeing, general homestead work can likewise be made less demanding and more dependable by utilizing biosensors coordinated with cell phones and handheld devices, rather than regular techniques such as redacting notes, keeping a ranch journal, or utilizing basic gear without information sharing capacities.

It's a major challenge to produce great quality and safe meat products to meet the expanding worldwide demand for such items. With greater vested interests comes increasing concerns for managing farm animal wellbeing. Devices placed under their skin, or in their stomachs give animal farmers access to helpful data with respect to their herd's conduct and medicinal conditions. These electronic devices are relied upon for the therapeutic treatment of animals, the recognition of warming and cooling needs, iontophoretic sedate conveyances, and even the protection of wild species. Another

critical utilization of biosensors is antimicrobial recognition. With the intensive and successive use of anti-infection agents in the animal business, antitoxin protection has become a noteworthy risk for agriculturists. Environmental precariousness is caused by the uncontrolled utilization of subrestorative antitoxins in concentrated animal feeding operation, which promotes anti-infection protection in creatures. There is a critical requirement for agriculturists to change to other options to maintain a strategic distance so that animals do not become invulnerable to antitoxin treatment. The dosage of anti-infection agents in the blood serum and muscles of ranch animals ought to be kept within a specific range, and there should be legitimate guidelines to test their antimicrobial levels (Mungroo and Neethirajan 2014). It is difficult to put a restriction on the use of anti-infection agents in domesticated animals, as antitoxins help cure the most widely recognized infirmities, such as enteric and respiratory contaminations. The utilization of anti-infection agents to protect ranch creatures against microbial infection is being perceived likewise all around the globe. To address this issue, the European Union set up a standard to keep a check on antimicrobial protection. This guideline, which has been recommended as a prudent step, concentrates on forbidding certain antimicrobial development products. Most extreme Residue Limits (MRLs) have been set up to limit those antimicrobials still permitted for animal use in the United States and European nations. MRL is that measure of pharmacologically dynamic substances and their inferred metabolites, which is legitimately allowed. Biosensors have been useful in such matters; they can easily recognize antimicrobial levels and caution the ranchers if antitoxins levels surpass a certain range.

In the following decade, the global market for wearable innovations aimed at animals is expected to generate around $1 billion to $2.5 billion, expanding more than 2.5 times. The leader for these exceptional innovations is China, which is producing these items at an extremely low cost, followed by the United States. A lot of cash is spent each year on agricultural research and animal wellbeing management. Be that as it may, it improves profitability to increase animal wellbeing. As a general rule, the financing is intended to give more up to date answers for the issues, as opposed to crossing over any barrier amongst research and industry. Banhazi and Black (Banhazi et al. 2012) have recommended that a thorough system be set up to guarantee farming practices are adequate and promote the flow of learning and research discoveries (Banhazi et al. 2012). In order to know the exact standard of animals' health well being and to counter the current high disease rates. it is evident that a new way should be developed to identify ailments. This move includes supplanting off-site lab tests for homesteads with fast test findings on the ranch itself. The world association for animal wellbeing (OIE) has cautioned that the zoonotic illnesses from intensive farming of domesticated animals can impact their general wellbeing if there is overflow from the cultivated creature repository, and the domesticated animals industry is under overwhelming strain to enhance its biosecurity conventions and improve creature traceability and welfare. Consequently, the animal business and sustenance wellbeing review offices are looking for new instruments and innovations, to empower fast, continuous and on-cultivate checking of sicknesses and record keeping.

Biosensing improvements guarantee enhancements to the execution, cost, and efficiency in animal disease management. Advancement and promotion of dependable, fast tests will permit a prior and more targeted treatment of diseases, conceivably

bringing about decreases in antimicrobial use and enhancements in animal welfare. The biosensors and early detecting tools help in designing the medicines or drugs even before the sickness kicks in and this could shape the administration segments of the coordinated ranch assessment and proAction display (Awasthi et al. 2016). Figure 20.3 shows the schematic working of a biosensor.

The combination of novel indicators and illness identifiers utilizing biosensors would keep animals and the farming industry one step ahead of undetectable infections through the use of satellites and cell phones. Smart and accurate domesticated animal farming and wellbeing management will keep on growing in importance to tackle the expanding demand for nourishment and guarantee sustainability. The biosensing advances—channeled through the IoT worldwide—will advance fast, establishing constant on-site farm monitoring of animal farming infections. The continuous spread of information gathered from the homesteads through these biosensors will have an impact beyond the ranch too; permitting entrance to this data that will demonstrate fundamental to the social permit issues confronting our rural area and will be a key to our proceeded with worldwide aggressiveness.

20.3.3.2 IoT-Based Beef Cattle Breeding System

Beef cattle breeding industry plays an important role in animal husbandry however the traditional methods are unable to meet the growing demands of beef. To deal with the problems of beef cattle breeding industry in China an IOT based beef cattle breeding system that take RFID technology into beef cattle was designed and combined with B/S and C/S structures. The framework gives the help of gathering and figuring out how to the cowshed and steers data, nourishing data, ailment treatment data, pandemic avoidance data, weight data and client data. In the meantime, the settled RFID peruser is utilized with the electronic scale to understand the precise record of weight change amid the procedure of hamburger cows rearing. The planned IOT based beef cattle breeding system has been applied and in use for several years in Lincang, Yunnan Province, China. The running outcomes demonstrated that it

FIGURE 20.3 Biosensor.

is steady and solid, and the genuine needs likewise can be met by each capacity. Generally speaking, the framework is reasonable for fine hamburger steers rearing and can enable the dairy cattle to cultivate understanding the advanced and traceable logical reproducing strategy (Wolfová et al. 2005).

20.4 CONCLUSION

The advantages of IoT innovations in farming operations include, yet are not restricted to, diminishing waste, better pest and animals management, and expanded profitability. New creative IoT applications are addressing these issues and expanding the quality, amount, sustainability, and cost viability of rural production. The present large and small farms can benefit from IoT innovations to screen soil moisture, guide crop management, improve herd nutrition levels, improve pest management system and water system infrastructures, and rapidly analyze operational information. Combined with other sources of data, such as climate predictions, this available knowledge can additionally enhance basic farming practices. On account of headway made by IoT, another era of farming is upon us soon.

REFERENCES

Awasthi, A., A. Awasthi, D. Riordan, and J. Walsh. 2016. "Non-Invasive Sensor Technology for the Development of a Dairy Cattle Health Monitoring System." *Computers* 5 (4): 23. doi: 10.3390/computers5040023.

Banhazi, T. M., H. Lehr, J. L. Black, H. Crabtree, P. Schofield, M. Tscharke, and D. Berckmans. 2012. "Precision Livestock Farming: An International Review of Scientific and Commercial Aspects." *International Journal of Agricultural and Biological Engineering*. 5(3): 1 doi: 10.3965/j.ijabe.20120503.00?

Blench, R. 2001. "'You Can't Go Home Again' Pastoralism in the New Millennium." *Odi* no. May 2001: 106. http://www.odi.org/sites/odi.org.uk/files/odi-assets/publications-opinion-files/6329.pdf. www.org.odi.uk.

De Boos, A. 2001. "Australian Wool - Competitive through Innovation." *International Textile Bulletin* 47 (2): 8–21. https://www.scopus.com/inward/record.uri?eid=2-s2.0-0035005472&partnerID=40&md5=a72c949083a03fcb767137d23231dfa8.

Diamond, D., C. Zhao, and L. Li. 2017. "5th International Symposium on Sensor Science Session Chairs," September 27–29, 2017, Barcelona, Spain.

Eller, H. and A. Denoth. 1996. "A Capacitive Soil Moisture Sensor." *Journal of Hydrology* 185 (1–4): 137–46. doi: 10.1016/0022-1694(95)03003-4.

FAO. 2016. "Food and Agriculture Organization of United Nations." *The Food and Agriculture Organization of the United Nations*. http://faostat.fao.org.

Frankiewicz, A. and R. Cupek. 2013. "Smart Passive Infrared Sensor—Hardware Platform." In *IECON Proceedings (Industrial Electronics Conference)*, 7543–47. doi: 10.1109/IECON.2013.6700389.

Kools, S. A. E., J. F. Moltmann, and T. Knacker. 2008. "Estimating the Use of Veterinary Medicines in the European Union." *Regulatory Toxicology and Pharmacology* 50 (1): 59–65. doi: 10.1016/j.yrtph.2007.06.003.

Kucharzyk, K. H., M. A. Deshusses, K. A. Porter, and H. Hsu-Kim. 2015. "Relative Contributions of Mercury Bioavailability and Microbial Growth Rate on Net Methylmercury Production by Anaerobic Mixed Cultures." *Environmental Science: Processes & Impacts*. 17(9): 1568–77. The Royal Society of Chemistry. doi: 10.1039/C5EM00174A.

Kuhnen, S., R. B. Stibuski, L. A. Honorato, and L. C. P. M. Filho. 2015. "Farm Management in Organic and Conventional Dairy Production Systems Based on Pasture in Southern Brazil and Its Consequences on Production and Milk Quality." *Animals* 5 (3): 479–94. doi: 10.3390/ani5030367.

Lee, J., B. Noh, S. Jang, D. Park, Y. Chung, and H. H. Chang. 2015. "Stress Detection and Classification of Laying Hens by Sound Analysis." *Asian-Australasian Journal of Animal Sciences* 28 (4): 592–98. doi: 10.5713/ajas.14.0654.

Luong, J. H. T., K. B. Male, and J. D. Glennon. 2008. "Biosensor Technology: Technology Push versus Market Pull." *Biotechnology Advances*. doi: 10.1016/j.biotechadv.2008.05.007.

McCaul, M., A. Porter, T. Glennon, R. Barrett, S. Beirne, G. G. Wallace, P. White, F. Stroiescu, and D. Diamond. 2017. "Wearable Sensor for Real-Time Monitoring of Electrolytes in Sweat." *Proceedings* 1 (10): 724. doi: 10.3390/proceedings1080724.

Moghe, R., F. C. Lambert, and D. Divan. 2012. "Smart Stick-on Sensors for the Smart Grid." *IEEE Transactions on Smart Grid* 3 (1): 241–52. doi: 10.1109/TSG.2011.2166280.

Mungroo, N. A. and S. Neethirajan. 2014. "Biosensors for the Detection of Antibiotics in Poultry Industry-A Review." *Biosensors* 4 (4): 472–93. doi: 10.3390/bios4040472.

Neethirajan, S. 2017. "Recent Advances in Wearable Sensors for Animal Health Management." *Sensing and Bio-Sensing Research* 12: 15–29. The Author. doi: 10.1016/j.sbsr.2016.11.004.

QGIS Development Team. 2015. "QGIS Geographic Information System." *Open Source Geospatial Foundation Project*. doi: http://www.qgis.org/.

Rodenburg, T. B., A. Breeding, and G. Centre. n.d. "The Role of Breeding and Genetics in Animal Welfare".

Rodenburg, T. B. and S. P. Turner. 2015. "The Role of Breeding and Genetics in Animal Welfare," 38.

Schmitz, A. and C. B. Moss. 2015. "Mechanized Agriculture: Machine Adoption, Farm Size, and Labor Displacement." *AgBioForum* 18 (3): 278–96.

Science, Animal Nutrition, and Gordon Mcl Dryden. 2008. "DOWNLOAD http://bit.ly/1J5XwKE http://goo.gl/RwOeh http://en.wikipedia.org/w/index.php?search=Animal+Nutrition+Science" 16.

Sellier, N., E. Guettier, and C. Staub. 2014. "A Review of Methods to Measure Animal Body Temperature in Precision Farming." *American Journal of Agricultural Science and Technology*. doi: 10.1016/j.sbsr.2016.11.00410.7726/ajast.2014.1008.

Starrs, P. F. 2000. "Let the Cowboy Ride: Cattle Ranching in the American West," no. March: 386. doi: 10.2307/2649639.

The Living Planet: A Portrait of The Earth by David Attenborough. n.d.

Toth, C. and G. Jóźków. 2016. "Remote Sensing Platforms and Sensors: A Survey." *ISPRS Journal of Photogrammetry and Remote Sensing*. doi: 10.1016/j.isprsjprs.2015.10.004.

Wolfert, S., L. Ge, C. Verdouw, and M. J. Bogaardt. 2017. "Big Data in Smart Farming—A Review." *Agricultural Systems*. doi: 10.1016/j.agsy.2017.01.023.

Wolfová, M., J. Wolf, J. Přibyl, R. Zahrádková, and J. Kica. 2005. "Breeding Objectives for Beef Cattle Used in Different Production Systems: 1. Model Development." *Livestock Production Science* 95 (3): 201–15. doi: 10.1016/j.livprodsci.2004.12.018.

Zhang, Q., J. R. Castro Smirnov, R. Xia, J. M. Pedrosa, I. Rodriguez, J. Cabanillas-Gonzalez, and W. Huang. 2017. "Highly pH-Responsive Sensor Based on Amplified Spontaneous Emission Coupled to Colorimetry." *Scientific Reports* 7: 1–6. doi: 10.1038/srep46265.

Index

A

AAC, *see* Assistive and augmentative communication
Accelerometer sensors, 237
Acid producing bacteria (APB), 93, 94; *see also* Microbial-induced corrosion
ACLU, *see* American Civil Liberties Union
ACO algorithm, *see* Ant colony optimization algorithm
Acute lymphoblastic leukemia (ALL), 82
Acute myeloid leukemia (AML), 82
Acutors, 303; *see also* IoT-based medical devices
Adomian decomposition, 99–101
Advance Research Project Agency Network (APRANET), 247
Agricultural IoT, 330
AI, *see* Artificial Intelligence
ALDs, *see* Assistive listening devices
ALL, *see* Acute lymphoblastic leukemia
ALS, *see* Amyotrophic lateral sclerosis
American Civil Liberties Union (ACLU), 235
American sign language (ASL), 299
AML, *see* Acute myeloid leukemia
Amorphous computing, 233
Amyotrophic lateral sclerosis (ALS), 87, 282
ANA, *see* Antinuclear antibody testing
Analysis of variance (ANOVA), 77
Analytical; *see also* Automated doctor; Smart analytical lab
  instruments, 313
  labs, 313
  softwares, 45
  techniques, 313
ANN, *see* Artificial neural network
ANOVA, *see* Analysis of variance
Ant colony optimization algorithm (ACO algorithm), 4
Antibiograms, 203
Antifungals, 202
Antimicrobials, 202
Antinuclear antibody testing (ANA), 122
Antiseptics, 202
APB, *see* Acid producing bacteria
APRANET, *see* Advance Research Project Agency Network
Arduino, 331; *see also* IoT in crop farming; Microcontrollers for IoT
  Arduino MKR1000, 238
  Arduino Uno, 238
AR modeling, *see* Autoregression modeling
Artifacts, 280
Artificial Intelligence (AI), 117, 194, 231, 281
Artificial neural network (ANN), 2, 4, 36, 78, 184, 213; *see also* Automated doctor; Machine learning
  artificial neuron, 6
  biological neuron, 5
  in biotechnology, 8
  features of, 109
  multilayered perceptron, 7
  sigmoidal function, 5
  sigmoid transfer function, 6
  synaptic weight, 5
ASB, *see* Asymptomatic bacteriuria
ASL, *see* American sign language
Assistive and augmentative communication (AAC), 307; *see also* IoT-based medical devices
Assistive listening devices (ALDs), 301; *see also* IoT-based medical devices
Association rule learning, 64, 118; *see also* Machine learning
Asymptomatic bacteriuria (ASB), 196
Australian Wool Innovation (AWI), 334
Autoimmune diseases, 114, 118–120, 128
  causes of, 119
  diabetes mellitus, 120–122
  grave's disease, 124–125
  incidence of, 115
  inflammatory bowel syndrome, 125–126
  lupus disease, 122–123
  multiple sclerosis, 127–128
  myasthenia gravis, 123–124
  systemic, 119
Autoimmunity, 114; *see also* Autoimmune diseases
Automated doctor, 217, 226
  algorithms, 218
  approach towards, 218
  case study, 223–225
  da Vinci robotic machine, 223–224, 225
  decision tree, 220–221
  deep learning, 221–222
  future research, 225–226
  KNN classification, 219–220
  methodologies, 222–223
  object identification in x-ray images, 222
  orthopilot, 224, 225
  regression, 218–219

Autoregression modeling (AR modeling), 184
AWI, *see* Australian Wool Innovation

B

Back error propagation (BEP), 7
Bacterimia, 196
Bayes, decision trees, back propagation neural networks (BPNNs), 125
Bayesian networks (BNs), 10, 66; *see also* Machine learning
  in biological science, 12
BCDE, *see* Bureau of Communicable Disease Epidemiology
BCI, *see* Brain–computer interface
Beef cattle breeding industry, 338; *see also* IoT for animal husbandry
BEP, *see* Back error propagation
BiCG (Biconjugated gradient method), 110
Big data, 17, 73, 133; *see also* Machine learning; Transcriptomics
  and biology, 19–20
  computing in system biology, 20
  computing platform applications, 21–22
  Hadoop framework, 17, 18
  implementation framework, 17–18
  MapReduce, 18–19
  omics technology, 19
Biocorrosion, *see* Microbial-induced corrosion
Bio-Emergences, 140; *see also* Embryogenesis imaging
Biofilm formation, 94; *see also* Biofilm induced corrosion
  cathodic depolarization process, 95
  formulating, 99
  stages of, 96
Biofilm induced corrosion, 94; *see also* Microbial-induced corrosion
  Adomian decomposition, 99–101
  artificial neural networks, 109
  Butler–Volmer equation modification, 104–107
  classification and regression trees, 107–108
  clustering, 108–109
  data handling techniques, 107
  decomposition rate evaluation, 101
  etch line corrosion, 96, 97
  flask shaped cavitation, 96, 97
  formulating biofilm formation, 99
  kinetics prediction models, 97–98
  K-nearest neighbor method, 108
  Laplace transformation of Plank–Nerst equation, 101–104
  Michaelis–Menten model, 98–99
  migration diffusion equation, 102
  nature of corrosion, 96
  optimal combination, 109–110
  pitting corrosion, 96
  Plank–Nernst law, 102
  support vector machines, 109
  uniform layer corrosion, 96, 97
Biological neuron, 5
Biomass constituents, 53
BIOnic Neurons (BIONs), 303
BIONs, *see* BIOnic Neurons
Biopharmaceuticals, 29
Bioprocess data, 29, 39
  analysis of, 34
  analysis on small-scale model, 31–33
  ANNs for analyzing wine production, 38–39
  case studies, 38
  data analysis steps, 34
  data mining, 36–37
  feature selection, 35–36
  knowledge discovery process, 34
  mining and analysis of, 29
  multiple linear regression, 37–38
  multivariate data analysis, 39
  objective of, 39
  online and off-line, 30
  optimization methods, 31
  preprocessing, 34–35
  random forests, 38
  regularized regression, 38
  sample data for illustration, 32
  softwares and mathematical models, 37
  types of, 30
Bio-Retina, 304; *see also* IoT-based medical devices
Biosensors, 315–316; *see also* Smart analytical lab
  for animal husbandry, 336–338
Biovigil Hygiene Technologies, 252; *see also* Healthcare IoT
BlueSentry software, 306; *see also* IoT-based medical devices
BNs, *see* Bayesian networks
BPNNs, *see* Bayes, decision trees, back propagation neural networks
Braille watches and braille printers, 301; *see also* IoT-based medical devices
Brain, 265, 266; *see also* Brain–computer interface; Electroencelagraphy
  cortex, 267–268
  cortical lobes, 268
  electrical impulse transmission, 269
  functioning of, 268–269
  nervous system, 266
  pyramidal neuron, 270
  signal reception, 269–270
Brain–computer interface (BCI), 265, 284, 306; *see also* Brain; Electroencelagraphy; Internet of things; IoT-based medical devices
  aiding invalids, 282
  applications, 281–282

Arduino Uno, 283
IoT-based applications, 282–283
Brain lesions, 181
background, 182
brain image processing stages, 186
case studies, 187–188
classification methods, 183
further reading, 190
hemorrhagic stroke image, 189
ischemic stroke segmentation challenge, 188–189
MLP network, 185
MRI image of ischemic stroke, 188
mTOP challenge, 189
neural network, 184–185
nonparametric classification, 184
outcome prediction of, 187
parametric classification, 183
regression methods, 184
segmentation of, 186–187
types of, 183
Brainwaves, 276
Bureau of Communicable Disease Epidemiology (BCDE), 199
Butler–Volmer equation modification, 104–107

C

CAD, *see* Computer aided diagnostic
CAF-IoT, *see* Crop and Animal Farming IoT
Canadian Nosocomial Infection Surveillance Program (CNISP), 199–200; *see also* NI programs
Cancer, 292–293
CancerLinQ, 292
computational testing methods, 295–296
TumorTracer, 295
CancerLinQ, 292; *see also* Wearable medical devices
Capillary electrophoresis (CE), 317
Cardiovascular disorders (CVD), 47
CART, *see* Classification and Regression Tree
CAST, *see* Corrupted clique graph
Catheter Associated Urinary Tract Infections (CAUTI), 196–197
CAUTI, *see* Catheter Associated Urinary Tract Infections
CCTE, *see* Computerized Cognitive Testing in Epilepsy
CDR, *see* Cognitive Drug Research Computerized Assessment System
CE, *see* Capillary electrophoresis
Cefaly device, 294; *see also* Wearable medical devices
CellML, 138; *see also* Domain-specific languages
Central Line-Associated Bloodstream Infections (CLABSI), 195–196
Central lines, 195
Central nervous system (CNS), 127
Central venous catheters (CVCs), 195
13CFLUX-2, 60
CGPGA, *see* Coarse-grained parallel genetic algorithm
Chemical sensor, 237
CiED, *see* Cipher for evolutionary design
Cipher for evolutionary design (CiED), 59
CLABSI, *see* Central Line-Associated Bloodstream Infections
Classification, 117; *see also* Machine learning
Classification and Regression Tree (CART), 38, 63, 107–108, 123
CLICK, *see* Cluster Identification via Connectivity Kernels
Cloud computing, 20, 321; *see also* Smart analytical lab
virtualization in, 322
Cluster computing, 20
Cluster Identification via Connectivity Kernels (CLICK), 78
Clustering, 10, 15–16, 64, 118; *see also* Machine learning
algorithms, 78
in biological science, 11
conceptual, 108
expectation maximization, 108
gene expression data, 77–78
for metabolomics data, 65
programs for cluster analysis, 78
Clusters of orthologous (COGs), 64
CNISP, *see* Canadian Nosocomial Infection Surveillance Program
CNS, *see* Central nervous system
CNT, *see* Cognitive Neurophysiological Test
Coarse-grained parallel genetic algorithm (CGPGA), 10
Cognitive Drug Research Computerized Assessment System (CDR), 294; *see also* Wearable medical devices
Cognitive Neurophysiological Test (CNT), 294–295; *see also* Wearable medical devices
COGs, *see* Clusters of orthologous
Colorimetric analysis, 318; *see also* Smart analytical lab
Commercial off the-shelf (COTS), 306
Compact digital lens-free microscopy smartphone, 315
CompuCell 3D, 140–141; *see also* Embryogenesis imaging
Computed tomography (CT), 136–137, 182, 291
Computer aided diagnostic (CAD), 121

Computerized Cognitive Testing in Epilepsy (CCTE), 294; *see also* Wearable medical devices
Confocal microscopy, 134–135
Corrupted clique graph (CAST), 78
Cortex, 267–268; *see also* Brain
Cortical lobes, 268; *see also* Brain
COTS, *see* Commercial off the-shelf
CRISPER (clustered regularly interspaced short palindromic repeats), 281
Crop and Animal Farming IoT (CAF-IoT), 328, 339; *see also* IoT for animal husbandry; IoT in crop farming
Cross infection, 203
CT, *see* Computed tomography
Cultivating innovations, 329
CVCs, *see* Central venous catheters
CVD, *see* Cardiovascular disorders
CXCR4s, 155; *see also* Theranostics

D

Data
  incorporation methods, 48
  safety, 258
  science, 2
Database management system (DBMS), 204
Data mining, 2, 12, 13, 114; *see also* Machine learning
  in autoimmune diseases, 118–128
  in biological science, 13
  classification, 15
  clustering, 15–16
  in diabetes mellitus, 120–122
  in grave's disease, 125
  in inflammatory bowel syndrome, 126
  knowledge discovery process, 14–15
  in lupus disease, 122–123
  in multiple sclerosis, 127–128
  in myasthenia gravis, 124
  in nutrigenomics, 43–50
  outlier analysis, 16–17
  regression, 16
  tools, 134
Da Vinci robotic machine, 223–224; *see also* Automated doctor
  arrangement of, 225
DBMS, *see* Database management system
dbNP, *see* Nutritional phenotype database
Decision tree (DT), 2, 3, 35, 63, 117, 220–221; *see also* Machine learning
  in biological science, 3
  diagram, 4
  induction of, 4
  learning, 63–64
Deep learning, 8, 221–222; *see also* Automated doctor; Machine learning
  algorithm, 9
  in biology, 9
Deep sequencing (Ref-Seq), 83; *see also* Transcriptomics
  data processing, 83
  in health care, 87
  mapping statistics, 86
  metrics, 86
  quality assessment, 83–84
  quality control, 85
  read mapping, 84
  read quantification and counting, 84–85
  RNA-seq experimental steps, 84
  tools and databases, 86
  transcript quantification, 85
Diabetes mellitus, 63, 120; *see also* Autoimmune diseases
  data mining in, 120–122
  tests to diagnose, 120
Digital video disc (DVD), 320
Disability, 299; *see also* IoT-based medical devices
  aids for, 299
  categories, 299
  cognitive impairments, 306–307
  deaf people, 305–306
  disabled to IoT-abled, 301–302
  external devices for, 301
  hearing disability classification, 305
  surviving with, 299–300
  technologies to help, 300–301
  visually impaired people, 303–305
Disinfectants, 202
DNS, *see* Domain name system
Domain name system (DNS), 289
Domain-specific languages, 138; *see also* Embryogenesis imaging
  CellML, 138
  CellML model components, 139
  CellML vs. SBML, 140
  SBML, 139
  SBML model components, 139
DOTA (1,4,7,10-Tetraazacyclododecane-1,4,7,10-tetraacetic acid), 152
DT, *see* Decision tree
DVD, *see* Digital video disc

E

EA, *see* Evolutionary Algorithm
EARL, *see* Emotion Annotation and Representation Language
Early cautioning frameworks (EWS), 334
ECG, *see* Electrocardiography
EcoCyc databases, 62
ECoG, *see* Electrocorticography
EEG, *see* Electroencelagraphy

EIoT, *see* Enterprise IoT
Elasticity coefficient, 58
Electrocardiography (ECG), 291
Electrocorticography (ECoG), 292, 303
Electroencelagraphy (EEG), 254, 270, 280–281, 291, 307; *see also* Brain; Brain–computer interface
artifacts, 280
benefits, 274–275
brainwaves, 276
data acquisition from subject, 271–272
drawback, 275
EEG patterns, 275–277
functioning, 270
KNN, 278, 279
logistic regression, 279–280
MLP-ANN, 279, 280
nasion, inion, and preauricular points, 273
resolution, 280
signal classification, 277
support vector machine, 277
SVM hyperplane, 278
10–20 system, 272–274
Electromyography (EMG), 291
Electronic Product Code (EPC), 288
Electronic stick for visually impaired people, 304; *see also* IoT-based medical devices
Embrace Watch, 291; *see also* Wearable medical devices
Embryogenesis imaging, 133; *see also* Domain-specific languages
Bio-Emergences, 140
CompuCell 3D, 140–141
computed tomography, 136–137
confocal microscopy, 134–135
functional-MRI, 137
future prospects, 141
light-sheet microscopy, 135–136
magnetic resonance imaging, 137
magnetic resonance spectroscopy, 138
microscopy techniques, 134–136
multiphoton microscopy, 135
ongoing projects, 140
ultrasound imaging, 136
Embryology, 134
EMG, *see* Electromyography
Emotion, 169–170
Emotion Annotation and Representation Language (EARL), 172; *see also* Emotion detection system
Emotion detection system, 168, 178–179; *see also* Speech parameters
applications, 176
classification model, 175–176
databases used, 174–175
emotions, 169
features and target labels, 175
HCI, 176–177
HUMAINE's proposal for EARL, 172
medical applications for, 177
negative emotions, 171
pad emotional state model, 171–172
PANA model, 170
Plutchik's model, 170
positive emotions, 171, 172
programming tools used, 174
responsive systems, 178
training and testing dataset, 174
train-test split, 175
Energy management, 243–244
Enhanced data collection, 232
Enterprise IoT (EIoT), 240
Environmental monitoring, 244
Enzyme activity, 58
EPC, *see* Electronic Product Code
Epilepsy, 291–292
cognitive neurophysiological test, 294–295
computerized assessment system, 294
Embrace Watch, 291
FePsy, 295
Neurocog FX, 295
EPS, *see* Extra cellular Polymeric Substances
ERP, *see* Event related potential
ESTs, *see* Expressed sequence tags
Etch line corrosion, 96, 97; *see also* Biofilm induced corrosion
EuclidianDistance, 278
Event related potential (ERP), 275
Evolutionary Algorithm (EA), 4, 60
EWS, *see* Early cautioning frameworks
Expressed sequence tags (ESTs), 74
Extra cellular Polymeric Substances (EPS), 94

F

Facial expression-based computer cursor control systems, 303; *see also* IoT-based medical devices
FePsy, 295; *see also* Wearable medical devices
FiatFlux, 60
Fingerprint matching system, 250; *see also* Healthcare IoT
Flask shaped cavitation, 96, 97; *see also* Biofilm induced corrosion
Fluorescence imaging, 317; *see also* Smart analytical lab
Fluorophores, 317
Flux, 57
Functional magnetic resonance imaging (fMRI), 137, 274

G

GA, *see* Genetic algorithms
General Purpose Graphic Processing Units (GPGPUs), 20
Gene-set enrichment analysis (GSEA), 49
Genetic algorithms (GA), 66–67, 117; *see also* Machine learning
Genome wide association studies (GWAS), 119
Geographic information system (GIS), 305, 334
GIS, *see* Geographic information system
Global positioning system (GPS), 289
Glucometer, 315
GPGPUs, *see* General Purpose Graphic Processing Units
GPS, *see* Global positioning system
GPU, *see* Graphical processing unit
Graphical processing unit (GPU), 20
Graphical user interface (GUI), 159
Grave's disease, 124–125; *see also* Autoimmune diseases
Grid computing, 20
GSEA, *see* Gene-set enrichment analysis
GUI, *see* Graphical user interface
GWAS, *see* Genome wide association studies
Gyroscope sensors, 237

H

Hadoop Distributed File System (HDFS), 18
HAI, *see* Hospital-acquired infection
Hand hygiene, 251; *see also* Healthcare IoT
HandTalk, 305–306; *see also* IoT-based medical devices
HCI, *see* Human–computer interaction
HDFS, *see* Hadoop Distributed File System
Healthcare IoT (H-IoT), 247
  data safety, 258
  ethical obstacles, 256
  fingerprint matching system, 250
  future prospects, 259
  hygiene dispenser, 252
  hygiene monitoring, 251–253
  monitoring consent, 258–259
  organ donation and transplantation, 254–256
  privacy, 256–257
  real-time location services, 248–251
  remote monitoring, 253–254
  sensitivity, 257
  working of, 249
  ZigBee, 254
Hearing aids, 301; *see also* IoT-based medical devices
Hemiplegia, 187; *see also* Brain lesions
Herbivores, 332; *see also* IoT for animal husbandry
HIMSS (Health Care Information and Management Systems Society), 251
H-IoT, *see* Healthcare IoT
HIV, *see* Human immunodeficiency virus
HLA, *see* Human leukocyte antigens
Home automation, 241, 242
Horror autotoxicus, 118
Hospital-acquired infection (HAI; NI), 194, 213–214; *see also* Hospital information systems; NI programs
  bloodstream infections, 195–196
  surgical site infections, 197–198
  types of, 194–195
  urinary tract infections, 196–197
  ventilator associated, 198–199
Hospital information systems, 203; *see also* Hospital-acquired infection
  data collection, 206
  data exclusions, 207
  data for all output types, 206
  data from interview, 207
  data gathering activity, 205
  data preparation activity, 207
  data sources, 205–206
  data type, 204
  data, unstructured, 205
  data variety, 206
  handling missing values, 207
  hospital records, 204
  imbalanced data problem, 207
  implementation, 208
  linear regression and ANN, 213
  one class classification implementation, 208
  Parzen density estimator, 210
  removing redundant variable, 208
  structured data, 204
  SVM, 208–210
  text classification method, 211–213
HUG, *see* University Hospital of Geneva
HUMAINE, *see* Human–machine interaction network on emotion
Human
  body, 114
  mind, 217
Human–computer interaction (HCI), 176–177; *see also* Emotion detection system
Human immunodeficiency virus (HIV), 155
Human leukocyte antigens (HLA), 119
Human–machine interaction network on emotion (HUMAINE), 172; *see also* Emotion detection system
Hygiene; *see also* Healthcare IoT
  dispenser, 252
  monitoring, 251–253
Hypertension, 290

I

IBL, *see* Instance Based Learning
ICD-10, *see* International Relegation of Diseases–10th Revision
ICTs, *see* Information and communication technologies
ICU, *see* Intensive care unit
IEEESA, *see* IEEE Standard Association
IEEE Standard Association (IEEESA), 308
IIoT, *see* Industrial Internet of Things
Image sensors, 237
Immune system, 118; *see also* Autoimmune diseases
Immune tolerance, 118
Immunodeficiency diseases, 118
INCA, *see* Isotopomer Network Compartmental Analysis
Indian agricultural sector, 329
Industrial Internet of Things (IIoT), 242
Inflammatory bowel syndrome, 125; *see also* Autoimmune diseases
  data mining in, 126
Information and communication technologies (ICTs), 230
Instance Based Learning (IBL), 117
Intel Edison, 239; *see also* Microcontrollers for IoT
Intensive care unit (ICU), 200
International Relegation of Diseases–10th Revision (ICD-10), 212
Internet, 247
Internet of Everything (IoE), 230
Internet of things (IoT), 230–231, 245, 248, 287, 313, 328; *see also* Brain–computer interface; Healthcare IoT; Microcontrollers for IoT; Smart analytical lab
  accelerometer sensors, 237
  advantages, 231
  advent of, 288–289
  agriculture, 243
  AI and ML, 236
  applications and everyday electronics, 240–241
  chemical sensor, 237
  compact digital lens-free microscopy smartphone, 315
  compliance, 235
  components, 236
  connectivity, 236–237
  ecosystem risk, 234
  elements, 283
  energy management, 243–244
  enhanced data collection, 232
  environmental monitoring, 244
  flexibility, 235
  growth of M2M connections, 232
  gyroscope sensors, 237
  in health care, 248
  home automation, 241, 242
  image sensors, 237
  limitations, 233
  manufacturing and industry, 242–243
  media and advertisement, 241–242
  medical and health care, 244
  mobile cloud computing, 314
  optical sensors, 237
  organ transplantation, 255, 256
  parts, 314
  privacy, 234–235
  security, 233–234
  sensor-related risk, 234
  sensors and modules, 237–238
  smart technology, 231
  transplantation system, 255
  transportation, 244
  user engagement, 231–232
  water quality sensor, 237
IoE, *see* Internet of Everything
IoT, *see* Internet of things
IoT-based beef cattle breeding system, 338–339; *see also* IoT for animal husbandry
IoT-based medical devices, 301; *see also* Brain–computer interface; Disability
  accessibility, 307
  acutors, 303
  architecture of IoT equipments, 302
  assistive and augmentative communication, 307
  Bio-Retina, 304
  BlueSentry software, 306
  broadband access, 308
  for cognitive impairments, 306–307
  drawbacks, 307
  electronic stick for visually impaired, 304
  for hearing disabilities, 305–306
  obstacle recognition system, 304
  physical and dexterity, 303
  privacy, 308
  RFID-based assistive devices, 304
  security, 308
  SmartVision, 305
  standardization, 308
  Thought Translational Device, 306–307
  virtual leading blocks, 304
  for visually impaired people, 303–305
IoT-based organ transplantation, 256
IoT-based smart farming stick, 331; *see also* IoT in crop farming
IoT-based wearable medical devices, 287; *see also* Internet of things; Wearable medical devices

IoT for animal husbandry, 331; *see also* Crop and Animal Farming IoT
automation and IoT, 336
biosensors, 336–338
geo-informatics technologies, 334
IoT-based beef cattle breeding system, 338–339
IT in molecular biology, 335
IT in simulation studies, 335
remote sensing, 334
ShearEzy, 334–335
sperm analyzer, 335–336
technological developments, 334
traditional animal husbandry, 331–333
IoT health care network, 301; *see also* IoT-based medical devices
and components, 302
IoT in crop farming, 328; *see also* Crop and Animal Farming IoT
agricultural IoT, 330
Arduino, 331
cultivating innovations, 329
modern farming technologies, 329
PIR sensors, 330
smart agriculture, 329–330
smart farming stick, 331
soil moisture sensor, 330
traditional farming, 328–329
IoT in smart analysis, 322–323; *see also* Smart analytical lab
Iron, 92
Isotopomer Network Compartmental Analysis (INCA), 60
iSperm analyser, 335; *see also* IoT for animal husbandry

K

KDD, *see* Knowledge discovery database
k Nearest Neighbour (kNN), 61, 78, 117, 279
algorithm, 220
classification, 219–220
EuclidianDistance, 278
method, 108
with two k values, 221
Knowledge discovery database (KDD), 12, 14–15; *see also* Machine learning
knowledge discovery, 13, 34, 116
knowledge retrieval, 115

L

LAMBERT, *see* Language Acquisition Manipulatives Blending Early Childhood Research and Technology
Language Acquisition Manipulatives Blending Early Childhood Research and Technology (LAMBERT), 306
Laplace transformation of Plank–Nerst equation, 101–104
Learning, 217
sample, 3
Left ventricular assist device (LVAD), 82
Lesion, 182; *see also* Brain lesions
LGBM, *see* Light gradient boosted model
Li-Fi (Light-Fidelity), 237
Light-Fidelity, *see* Li-Fi
Light gradient boosted model (LGBM), 176
Light-sheet microscopy, 135–136
Linear regression (LR), 213
Logistic function, *see* Sigmoidal function
Logistic regression (LR), 175
Log-sigmoid, *see* Sigmoidal function
Low-power wide-area networking (LPWAN), 237
LPWAN, *see* Low-power wide-area networking
LR, *see* Linear regression; Logistic regression
Lupus disease, 122; *see also* Autoimmune diseases
data mining in, 122–123
LVAD, *see* Left ventricular assist device

M

MAA, *see* Macroaggregated albumin
Machine learning (ML), 2, 128, 217, 231; *see also* Big data; Data mining
approaches to, 3
artificial neural networks, 4–8
association rule learning, 64
association rule mining, 118
Bayesian networks, 10–12, 66
big data, 17
categories, 116, 117
clustering, 10, 64–65, 118
data mining, 12, 13
decision tree, 3–4, 63–64
deep learning, 2, 8–9
genetic algorithms, 66–67
knowledge discovery, 116
and knowledge discovery tasks, 116
vs. other computing environments, 2
regression, 117
reinforced learning, 118
vs. statistics, 2–3
supervised learning, 116–117
support vector machines, 9–10
techniques, 61–63
unsupervised learning, 117–118
Machine to machine (M2M), 243
Macroaggregated albumin (MAA), 152
Magnetic resonance (MR), 182
Magnetic resonance imaging (MRI), 137, 182, 291, 313
of ischemic stroke, 188
Magnetic resonance spectroscopy, 138
Magnetoencephlaography (MEG), 274

Major histocompatibility complex (MHC), 119
Mapping statistics, 86
MapReduce, 18–19
Mass balance equation, 54
Massively parallel signature sequencing (MPSS), 74
Massive Online Analysis (MOA), 37
Mass spectrometry (MS), 60
MatLab, 59
mCRPC, *see* Metastatic castration-resistant PCa
MDB, *see* Metal depositing bacteria
MDF, *see* Medium-density fibreboard
MDR, *see* Multi Drug Resistant
Mediatek Linkit One, 240; *see also* Microcontrollers for IoT
Medium-density fibreboard (MDF), 320
MEG, *see* Magnetoencephlaography
Mel-frequency cepstrum coefficient (MFCC), 172; *see also* Speech parameters
  benefits, 174
  determination, 173
  process, 173
Message passing interface (MPI), 20
Messenger RNA (mRNA), 74
Metabolic adjustment (MoMA), 59
Metabolic engineering, 53
  association rule learning, 64
  Bayesian networks, 66
  clustering, 64–65
  comparing cellular metabolism, 55–56
  decision tree learning, 63–64
  FiatFlux, 60
  future aspects, 67
  genetic algorithms, 66–67
  machine learning techniques, 61–63
  mass balance equation, 54
  mathematical models in rengineering, 54–57
  MatLab, 59
  metabolic control analysis, 58–59
  metabolic flux analysis, 57–58
  need for, 54
  OptFlux, 60
  softwares used in, 59–61
  stoichiometric matrix model, 56
Metabolomics, 61
MetaCyc databases, 62
Meta-iodobenzylguanidine (MIBG), 152, 160
Metal corrosion, 92; *see also* Microbial-induced corrosion
Metal depositing bacteria (MDB), 93
Metal reducing bacteria (MRB), 93; *see also* Microbial-induced corrosion
Metastatic castration-resistant PCa (mCRPC), 154
MFCC, *see* Mel-frequency cepstrum coefficient
MHC, *see* Major histocompatibility complex
MI, *see* Mitotic index
MIBG, *see* Meta-iodobenzylguanidine
MIC, *see* Microbial-induced corrosion
Michaelis–Menten kinetics equation, 98–99
Microarray, 74–76; *see also* Transcriptomics
  biological significance, 79
  classification, 78, 79
  clustering gene expression data, 77–78
  data analysis, 76, 77
  databases and tools for data analysis, 81
  differentially expressed gene identification and analysis, 76–78
  in health care, 81–83
  monitoring RNA and array quality, 80
  probes, 74
  programs for cluster analysis, 78
  quality control, 80
  steps to investigate gene expression, 75
  tools and databases, 80
  tools available for functional annotation, 79
  ViroChip system, 83
Microarray Database, 49; *see also* Nutrigenomics
Microbial-induced corrosion (MIC), 91, 92, 110; *see also* Biofilm induced corrosion
  acid producing bacteria, 94
  biofilm formation, 94
  broadly classified phases, 93
  cathodic depolarization process, 95
  metal depositing bacteria, 93
  metal reducing bacteria, 93
  mycetes, 94
  substrate organism interaction, 93
  sulfate reducing bacteria, 93
  in under biofilm, 95
Micro computed tomography (micro-CT), 136–137
Microconsortia, 94
Microcontrollers for IoT, 238; *see also* Internet of things
  Arduino MKR1000, 238
  Arduino Uno, 238
  Intel Edison, 239
  Mediatek Linkit One, 240
  Nanode, 240
  Pinguino 45K50, 239
  Raspberry Pi 2 Model B, 238–239
  Raspberry Pi 3, 239
  Teensy 2.0, 239–240
micro-CT, *see* Micro computed tomography
Micro-MRI, 137
MicroRNA (miRNA), 74
Migraines, 293–294
Migration diffusion equation, 102
Mild Traumatic Brain Injury Outcome Prediction Challenge (mTOP Challenge), 189
MILP, *see* Mixed integrated linear programming
Mining methods, 12; *see also* Machine learning
m-IoT, 303; *see also* IoT-based medical devices
miRNA, *see* MicroRNA
Mitotic index (MI), 153

Mixed integrated linear programming (MILP), 60
ML, *see* Machine learning
MLP, *see* Multilayer perceptron
MLP-ANN, *see* Multilayer Perceptron-Artificial Neural Networks
MOA, *see* Massive Online Analysis
Mobile cloud computing, 314
MoMA, *see* Metabolic adjustment
Most extreme Residue Limits (MRLs), 337
MPI, *see* Message passing interface
MPSS, *see* Massively parallel signature sequencing
MR, *see* Magnetic resonance
MRB, *see* Metal reducing bacteria
MRI, *see* Magnetic resonance imaging
MRLs, *see* Most extreme Residue Limits
mRNA, *see* Messenger RNA
MS, *see* Mass spectrometry
mTOP Challenge, *see* Mild Traumatic Brain Injury Outcome Prediction Challenge
M2M, *see* Machine to machine
Multi Drug Resistant (MDR), 198
Multilayer perceptron (MLP), 6, 185
Multilayer Perceptron-Artificial Neural Networks (MLP-ANN), 279, 280
Multiobjective algorithm, 4
Multiphoton microscopy, 135
Multiple sclerosis, 127; *see also* Autoimmune diseases
  data mining in, 127–128
Myasthenia gravis, 123; *see also* Autoimmune diseases
  data mining in, 124
  diagnostic methods, 123–124
  symptoms, 123
Mycetes, 94; *see also* Microbial-induced corrosion

N

NAD, *see* Nicotinamide adenine dinucleotide
NAFLD, *see* Nonalcoholic fatty liver disease
Nanode, 240; *see also* Microcontrollers for IoT
Nanotheranostics, 150; *see also* Theranostics
Natural Language Toolkit (NLTK), 37
Nervous system, 266; *see also* Brain
NET, *see* Neuroendocrine tumors
Neurocog FX, 295; *see also* Wearable medical devices
Neuroendocrine tumors (NET), 152, 153
Next generation sequencing (NSG), 19
NI (Nosocomial Infection), 250
NI control methods, 201; *see also* NI programs
  health care staff training, 202
  immunization and vaccination, 201
  precautions, 201
Nicotinamide adenine dinucleotide (NAD), 318
Nipah infection, 333
NI programs, 199; *see also* Hospital-acquired infection
  antimicrobial use and resistance, 202–203
  CNISP, 199–200
  cross-infection programs, 203
  failure in hospital, 200
  success in hospital, 200–201
NLTK, *see* Natural Language Toolkit
NMF, *see* Nonnegative matrix factorization
NMR, *see* Nuclear magnetic resonance
Nonalcoholic fatty liver disease (NAFLD), 87
Non-multi drug resistant pathogens, 198
Nonnegative matrix factorization (NMF), 35
NSG, *see* Next generation sequencing
Nuclear magnetic resonance (NMR), 138
Nutraceuticals, 44; *see also* Nutrigenomics
  containing natural food source, 46
  in markets, 46–47
Nutrigenetics, 43; *see also* Nutrigenomics
Nutrigenomics, 43
  conventional methods for data incorporation, 48
  database vs. data mining, 48
  data mining, 48–49
  future prospects, 50
  in health care, 47
  natural food sources, 45–46
  nutraceuticals, 46–47
  nutrigenomic biomarker of health, 44
  nutrigenomic research tools, 45
Nutritional genomics, 43; *see also* Nutrigenomics
Nutritional interventions, 44
Nutritional phenotype database (dbNP), 49

O

Object Naming System (ONS), 289
Obstacle recognition system, 304; *see also* IoT-based medical devices
Ochratoxin A (OTA), 317
ODEs, *see* Ordinary differential equations
Omics, 45
One class classification implementation, 208
Online sign language translator, 301; *see also* IoT-based medical devices
ONS, *see* Object Naming System
OptFlux, 60
Optical sensors, 237
Ordinary differential equations (ODEs), 110
Organ donation and transplantation, 254–256; *see also* Healthcare IoT
Orthopilot, 224, 225; *see also* Automated doctor
OS, *see* Overlapping swarm
OTA, *see* Ochratoxin A
Outlier, 16–17
Overlapping swarm (OS), 11

P

PAC, *see* Pulmonary artery catheters
Pad emotional state model, 171–172; *see also* Emotion detection system
PANA model, *see* Positive activation–negative activation model
Parallel virtual machine (PVM), 20
Partial least squares (PLS), 37
Parzen density estimator, 210
Passive infrared sensors (PIR sensors), 330; *see also* IoT in crop farming
Patient generated health data (PGHD), 292
PBMC, *see* Peripheral blood mononuclear cells
PCA, *see* Principle component analysis
PCa, *see* Prostate cancer
PCR, *see* Polymerase chain reaction
PDA, *see* Personal digital assistant
PDT, *see* Photo-dynamic therapy
Peptide receptor radionuclide therapy (PRRT), 150, 152
Peripheral blood mononuclear cells (PBMC), 81
Peripherally inserted CVC (PICC), 195
Personal digital assistant (PDA), 319
Personalized medicine, 43
PET, *see* Positron emission tomography
PGHD, *see* Patient generated health data
Photo-dynamic therapy (PDT), 293
Photoplethysmography (PPG), 291
PICC, *see* Peripherally inserted CVC
Pinguino 45K50, 239; *see also* Microcontrollers for IoT
PIR sensors, *see* Passive infrared sensors
Pitting corrosion, 96; *see also* Biofilm induced corrosion
Plank–Nernst law, 102
PLS, *see* Partial least squares
Plutchik's model, 170; *see also* Emotion detection system
PoC instruments, *see* Point-of-care instruments
Point-of-care instruments (PoC instruments), 314
Polymerase chain reaction (PCR), 45
Positive activation–negative activation model (PANA model), 170; *see also* Emotion detection system
  emotions, 171
Positron emission tomography (PET), 147, 156, 274, 291
PPG, *see* Photoplethysmography
PPMS, *see* Primary progressive multiple sclerosis
Primary metabolites, 53
Primary progressive multiple sclerosis (PPMS), 127
Principle component analysis (PCA), 35
PRMS, *see* Progressing relapsing multiple sclerosis
Progressing relapsing multiple sclerosis (PRMS), 127
Prostate cancer (PCa), 154
Prostate-specific membrane antigen (PSMA), 152, 154–155; *see also* Theranostics
PRRT, *see* Peptide receptor radionuclide therapy
PSMA, *see* Prostate-specific membrane antigen
Pulmonary artery catheters (PAC), 195
PVM, *see* Parallel virtual machine
Pyramidal neuron, 270; *see also* Brain

Q

Quantitative SPECT, 157–158; *see also* Theranostics
  Bremsstrahlung radiation, 158
  camera dead-time, 158
  camera normalization, 158
  factors affecting measure of, 158
  image-based computational tools, 160–161
  patient specific dose calculations, 159–160
  quantitative preclinical imaging, 159
  quantitative reconstructions, 159
  scatter corrections, 158
  to study organ activity, 160

R

Radial basis function (RBF), 185
Radioactive Iodine uptake (RAIU), 125
Radio frequency (RF), 137
Radio-frequency identification (RFID), 237, 288; *see also* IoT-based medical devices
  based assistive devices, 304
Radioiodine theranostics, 152; *see also* Theranostics
Radio-ligand properties, 154
Radiopharmaceuticals, 151
RAIU, *see* Radioactive Iodine uptake
Random forest (RF), 82
Raspberry Pi, 238–239; *see also* Microcontrollers for IoT
RBF, *see* Radial basis function
Real-time location device, 249
Real-time location systems (RTLS), 248–251; *see also* Healthcare IoT
Red, green, and blue (RGB), 317
Ref-Seq, *see* Deep sequencing
Regression, 117, 218–219; *see also* Machine learning
  cost-function with n-iterations, 219
  variation of cost-function with learning factor, 219
Regulatory on/off regularization (ROOM), 59
Reinforced learning, 118; *see also* Machine learning
Relapsing remitting multiple sclerosis (RRMS), 127
Remote monitoring, 253–254; *see also* Healthcare IoT

Remote Sensing (RS), 334
Resolution, 280
Responsive neurostimulator (RNS), 292
Reverse transcriptase quantitative PCR (RT-qPCR), 74
RF, *see* Radio frequency; Random forest
RFID, *see* Radio-frequency identification
RGB, *see* Red, green, and blue
RNA sequencing (RNA-Seq), 74, 83; *see also* Deep sequencing
RNS, *see* Responsive neurostimulator
Robotic machines, 226; *see also* Automated doctor
ROOM, *see* Regulatory on/off regularization
RRMS, *see* Relapsing remitting multiple sclerosis
RS, *see* Remote Sensing
RTLS, *see* Real-time location systems
RT-qPCR, *see* Reverse transcriptase quantitative PCR

S

SA, *see* Simulated annealing
SAGE, *see* Serial analysis of gene expression
SAX, *see* Symbolic aggregate approximation
SBML (The Systems Biology Markup Language), 134, 139; *see also* Domain-specific languages
  model, 139
SCC, *see* Singlecell capture
SCPs, *see* Slow cortical potentials
Screen readers, 301; *see also* IoT-based medical devices
Secondary metabolites, 53
Secondary progressive multiple sclerosis (SPMS), 127
Self-organizing maps (SOM), 65, 78
Serial analysis of gene expression (SAGE), 74
ShearEzy, 334–335; *see also* IoT for animal husbandry
Sigmoidal function, 5
Simulated annealing (SA), 60, 65
Singlecell capture (SCC), 19
Single nucleotide polymorphisms (SNPs), 43, 81
Single photon emission computed tomography (SPECT), 147, 291
Single Photon Emission Computed Tomography-Computed Tomography (SPECT-CT), 157
SLE, *see* Systemic lupus erythematosus
Slime layer, 94
Slow cortical potentials (SCPs), 307
Smart analytical lab, 313; *see also* Internet of things
  biosensors, 315–316
  cloud computing, 321–322
  colorimetric analysis, 318
  fluorescence imaging, 317
  interconnected devices in smart lab, 321
  IoT in smart analysis, 322–323
  smart analytics, 315
  spectroscopy, 320
  ultraviolet method, 318–319
Smart home, 241, 242
Smart hospital room, 252; *see also* Healthcare IoT
Smart machines, 168
Smart technology, 231
SmartVision, 305; *see also* IoT-based medical devices
SNPs, *see* Single nucleotide polymorphisms
Soil moisture sensor, 330; *see also* IoT in crop farming
SOM, *see* Self-organizing maps
Somatostatin receptors (SSTRs), 153
Somatostatin receptor subtypes 2 (SSTR2), 152
Soybean Genomics, 49; *see also* Nutrigenomics
SPECT, *see* Single photon emission computed tomography
SPECT-CT, *see* Single Photon Emission Computed Tomography-Computed Tomography
Spectroscopy, 320; *see also* Smart analytical lab
Speech parameters, 172; *see also* Emotion detection system
  amplitude and pitch, 172–173
  choice of parameters, 174
  MFCC parameters, 173, 174
Sperm analyzer, 335–336; *see also* IoT for animal husbandry
SPIONs, *see* Super paramagnetic iron oxide nanoparticles
SPMS, *see* Secondary progressive multiple sclerosis
SRB, *see* Sulfate reducing bacteria
SRM, *see* Structural risk minimization
SSI, *see* Surgical Site Infections
SSTR2, *see* Somatostatin receptor subtypes 2
SSTRs, *see* Somatostatin receptors
Stoichiometric matrix model, 56
Stroke, 187; *see also* Brain lesions
Structural risk minimization (SRM), 37
Sulfate reducing bacteria (SRB), 93; *see also* Microbial-induced corrosion
Super paramagnetic iron oxide nanoparticles (SPIONs), 150
Supervised learning, 116–117; *see also* Machine learning
Support vector machine (SVM), 2, 9, 61, 78, 109, 277; *see also* Machine learning
  accuracy scores and logloss, 176
  in biological science, 10
  hyperplane, 278
  as one class classifier, 208
  in predictive approaches, 36–37

Surgical Site Infections (SSI), 197–198
SVM, *see* Support vector machine
SyBiOntKB, 48
Symbolic aggregate approximation (SAX), 35
Symptomatic urinary tract infection, 196
System biology, 20; *see also* Big data
Systemic lupus erythematosus (SLE), 122

T

Tabu search (TS), 65
Taxane-based therapies, 154
Teensy 2.0, 239–240; *see also* Microcontrollers for IoT
Telemedicine, 253; *see also* Healthcare IoT
10–20 system, 272–274
Term frequency (TF), 212
Term frequency–inverse document frequency (TF-IDF), 212
Tetralogy, 134
Text classification method, 211–213
TF, *see* Term frequency
TF-IDF, *see* Term frequency–inverse document frequency
The Institute of Genomics Research (TIGR), 80
Theragnosis, 146; *see also* Theranostics
Theranostics, 146, 161
  agents, 147
  application of, 149, 153
  architecture of, 151
  construction in single integrated system, 146
  CXCR4s, 155
  Ga-68, F-18/(Y-90), 152
  iodine, 152
  Lu-177/Ac-225, 152
  metabolic active radiopharmaceuticals, 150–151
  nanotheranostics, 150
  neuroendocrine tumors, 153
  PET, 156
  PET/CT, 156
  principle of, 147
  problems, 155–156
  PSMA, 154–155
  published papers in, 148
  quantitative SPECT, 157–161
  radiopharmaceuticals, 151
  SPECT/CT, 157
  targets in nuclear medicine, 148–150
  transitioning into clinical reality, 156
  as working model in targeted human organ, 148
  Y-90, 152
  Zr-89, 152–153
Thought Translational Device (TTD), 306–307; *see also* IoT-based medical devices
TIGR, *see* The Institute of Genomics Research
TIVAD, *see* Totally implantable venous access devices
TNF, *see* Tumor necrosis factor
TNF receptor superfamily member 6 (TNFRSF6), 119
TNFRSF6, *see* TNF receptor superfamily member 6
Totally implantable venous access devices (TIVAD), 195
Traditional animal husbandry, 331–333; *see also* IoT for animal husbandry
  animal health, 333
  breeding, 333
  feeding, 332–333
  herbivores, 332
  husbandry, 332
  systems, 332
Transcriptomics, 73; *see also* Big data; Deep sequencing; Microarray
  background, 73–74
  future prospects, 87–88
  methods in, 74
Traumatic brain injuries, 189; *see also* Brain lesions
TS, *see* Tabu search
TTD, *see* Thought Translational Device
Tumor necrosis factor (TNF), 87, 119
TumorTracer, 295; *see also* Wearable medical devices

U

UC, *see* Ulcerative colitis
UID, *see* Unique identification number
Ulcerative colitis (UC), 126
Ultrasound imaging, 136
Ultraviolet method (UV method), 318–319; *see also* Smart analytical lab
Uniform layer corrosion, 96, 97; *see also* Biofilm induced corrosion
Unique identification number (UID), 304
University Hospital of Geneva (HUG), 210
Unsupervised learning, 117–118; *see also* Machine learning
Upright Posture Shearing Platform (UPSP), 334
UPSP, *see* Upright Posture Shearing Platform
Urinary tract infections (UTI), 196
User engagement, 231–232
UTI, *see* Urinary tract infections
UV method, *see* Ultraviolet method
UV-Vis spectrometry, 319

V

Vagus nerve stimulation (VNS), 291
VAP, *see* Ventilator Associated Pneumonia
Vasotrac, 290; *see also* Wearable medical devices

Ventilator Associated Pneumonia (VAP), 198–199
Very small aperture terminal (VSAT), 237
Vibrant Sound Bridge (VSB), 301
ViroChip system, 83
Virtual leading blocks, 304; *see also* IoT-based medical devices
VNS, *see* Vagus nerve stimulation
VSAT, *see* Very small aperture terminal
VSB, *see* Vibrant Sound Bridge

W

Water quality sensor, 237
WBCDD, *see* Wisconsin Breast Cancer Diagnosis Dataset
Wearable medical devices (WMDs), 288, 296
  age of, 289
  cancer, 292–293, 295–296
  Cefaly device, 294
  cognitive drug research, 294
  cognitive neurophysiological test, 294–295
  components of, 288
  computational testing methods, 294–296
  computerized cognitive testing in epilepsy, 294
  detection, diagnosis, and treatment, 290
  Embrace Watch, 291
  epilepsy, 291–292, 294–295
  FePsy, 295
  history of medical devices, 288
  hypertension, 290
  mechanism of, 289–290
  migraines, 293–294
  Neurocog FX, 295
  Vasotrac, 290
Weighted gene coexpression network analysis (WGCNA), 82, 87
WGCNA, *see* Weighted gene coexpression network analysis
WHO, *see* World Health Organization
Wireless monitoring device (WMD), 254
Wireless network sensor (WNS), 253
Wisconsin Breast Cancer Diagnosis Dataset (WBCDD), 295
WMD, *see* Wireless monitoring device
WMDs, *see* Wearable medical devices
WNS, *see* Wireless network sensor
World Health Organization (WHO), 194, 251, 299, 305
World Wide Web (WWW), 247
WWW, *see* World Wide Web

Y

Y-90, 152; *see also* Theranostics

Z

ZigBee, 254; *see also* Healthcare IoT
Zr-89, 152–153; *see also* Theranostics