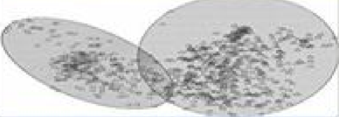
LEARNING From DATA *Made Easy* with R



A Gentle Introduction For Data Science

N.D Lewis

Ego autem et domus

mea serviemus Domino.

LEARNING FROM DATA MADE EASY WITH R

A Gentle Introduction for Data Science.

Dr. N.D. Lewis

Ordering Information: Quantity sales. Special discounts are available on quantity purchases by corporations, associations, and others. For details, email: info@NigelDLewis.com

Image photography by Deanna Lewis

ISBN-13: 978-1523459735
ISBN-10: 1523459735

Contents

Dedicated to Angela, wife, friend and mother extraordinaire.

Acknowledgments

A special thank you to:

My wife Angela, for her patience and constant encouragement.

My daughter Deanna, for taking hundreds of photographs for this book and my website.

And the readers of my earlier books who contacted me with questions and suggestions.

- Illustrations to deepen your understanding.
- Worked examples in R you can easily follow and immediately implement.
- Ideas you can actually use.

Learning from Data Made Easy with R is your very own hands on practical, tactical, easy to follow guide to mastery. This is an exciting time to be involved in data science. Buy this book today and join the data science revolution!

Preface

THANK you for picking up this book. I hope these ideas help you accelerate you data science practice as much as they have helped me and thousands of others. In fact, I hope that this book makes the tools of data science more accessible to you and thousands of people just like you.

There is never enough time to learn everything. You are literally swamped with work and personal responsibilities, projects, deadlines, and an array of miscellaneous tasks that consume your day. This book is designed to be a hands on practical, easy to read and use guide to the most successful ideas, outstanding techniques and usable solutions available to the data scientist for learning from data.

The ideas and practical information you'll find here will work just as well for the individual data scientist working for a small advertising firm, the team of three decision scientists employed by a regional pet foods company, the student completing a data science project for their coursework, or the solo consultant engaged on a forecasting project for the local health authority. You don't need to be a genius statistician or programming guru to understand and benefit from the practical ideas and straightforward solutions discussed in this text.

The focus is on the "how" because as Benjamin Franklin wisely stated "*Tell me and I forget. Teach me and I remember. Involve me and I learn.*" It is practical knowledge that will drive innovation and actual practical solutions for you.

This book came out of the desire to put the powerful technology of machine learning into the hands of the everyday practitioner. The material is therefore designed to be used by the individual whose primary focus is on data analysis and modeling. The focus is solely on those techniques, ideas and strategies that have been proven to work and can be quickly digested and deployed in the minimal amount of time.

On numerous occasions, individuals in a wide variety of disciplines and industries, have asked "how can I quickly understand and use the techniques required to learn from data in my area of interest?" The answer used to involve reading complex mathematical texts and then programming complicated formulas in languages such as C, C++ and Java.

With the rise of R, learning from data is now easier than ever. This book is designed to give you rapid access. It shows you, step by step, how to build each type of model in the free and popular R statistical package. Examples are clearly described and can be typed directly into R as printed on the page.

The least stimulating aspect of the subject for the practitioner is the mechanics of calculation. Although many of the horrors of this topic are necessary for the theoretician, they are of minimal importance to the practitioner and can be eliminated almost entirely by the use of the R package. It is inevitable that a few fragments remain, these are fully discussed in the course of this text. However, as this is a hands on, role up your sleeves and apply the ideas to real data book, I do not spend much time dealing with the minutiae of algorithmic matters, proving theorems, discussing lemmas or providing proofs.

New users to R can use this book easily and without any prior knowledge. This is best achieved by typing in the examples as they are given and reading the comments which follow. Copies of R and free tutorial guides for beginners can be downloaded at `https://www.r-project.org/`. If you are totally new to R take a look at the amazing tutorials at `http://cran.r-project.org/other-docs.html`; they do a great job introducing R to the novice.

In the end, data science isn't about mathematical lemmas or proving theorems. It's ultimately about real life, real people and the application of machine learning algorithms to real world problems in order to drive useful solutions. No matter who you are, no matter where you are from, no matter your background or schooling, you have the ability to master the ideas outlined in this book. With the appropriate software tool, a little persistence and the right guide, I personally believe data science techniques can be successfully used in the hands of anyone who has a real interest.

Greek philosopher Epicurus once said *"I write this not for the many, but for you; each of us is enough of an audience for the other."* Although the ideas in this book reach out to thousands of individuals, I've tried to keep Epicurus's principle in mind–to have each page you read give meaning to just one person - YOU.

A Promise

When you are done with this book, you will be able to implement one or more of the ideas I've talked about in your own particular area of interest. You will be amazed at how quick and easy the techniques are to use and deploy with R. With only a few different uses you will soon become a skilled practitioner.

I invite you therefore to put what you read in these pages into action. To help you do that, I've created **"12 Resources to Supercharge Your Productivity in R**", it is yours for **FREE**. Simply go to *http://www.AusCov.com* and download it now. It's my gift to you. It shares with you 12 of the very best resources you can use to boost your productivity in R.

Now, it's your turn!

Dr. Nigel D. Lewis

How to Get the Most from this Book

THIS is a hands on, role up your sleeves and experiment with the data and R book. You will get the maximum benefit by typing in the examples, reading the reference material and experimenting. By working through the numerous examples and reading the references, you will broaden your knowledge, deepen you intuitive understanding and strengthen your practical skill set.

There are at least two other ways to use this book. You can dip into it as an efficient reference tool. Flip to the chapter you need and quickly see how calculations are carried out in R. For best results type in the example given in the text, examine the results, and then adjust the example to your own data. Alternatively, browse through the real world examples, illustrations, case studies, tips and notes to stimulate your own ideas. This is useful for getting the general idea and also a source of pointers to relevant examples, case studies and literature.

☛ ***PRACTITIONER TIP*** ☚

If you are using Windows you can easily upgrade to the latest version of R using the `installr` package. Enter the following:

```
> install.packages("installr")
> installr::updateR()
```

If a package mentioned in the text is not installed on your machine you can download it by typing `install.packages("package_name")`. For example, to download the `class` package you would type in the R console:

```
install.packages("class")
```

Once the package is installed, you must call it. You do this by typing in the R console:

```
require(class)
```

The `class` package is now ready for use. You only need type this once, at the start of your R session.

Functions in R often have multiple parameters. In the examples in this text I focus primarily on the key parameters required for rapid model development. For information on additional parameters available in a function, type in the R console `?function_name`. For example, to find out about additional parameters in the `naiveBayes` function, you would type:

```
?naiveBayes
```

Details of the function and additional parameters will appear in your default web browser. After fitting your model of interest you are strongly encouraged to experiment with additional parameters.

I have also included the `set.seed` method in the R code samples throughout this text to assist you in reproducing the results exactly as they appear on the page.

The R package is available for all the major operating systems. Due to the popularity of Windows, examples in this book use the Windows version of R.

☛ ***PRACTITIONER TIP*** ☚

Can't remember what you typed two hours ago! Don't worry, neither can I! Provided you are logged into the same R session you simply need to type:

```
history(Inf)
```

It will return your entire history of entered commands for your current session.

You don't have to wait until you have read the entire book to incorporate the ideas into your own analysis. You can experience their marvelous potency for yourself almost immediately.

You can go straight to the section of interest and immediately test, create and exploit it in your own research and analysis.

.

☛ ***PRACTITIONER TIP*** ☚

On 32-bit Windows machines, R can only use up to 3Gb of RAM, regardless of how much you have installed. Use the following to check memory availability:

```
memory.limit()
```

To remove all objects from memory:

```
rm(list=ls())
```

As implied by the title, this book is about understanding and then hands use of data science models; more precisely, it is an attempt to give you the tools you need to build classifiers easily and quickly using R. The objective is to provide you the reader with the necessary tools to do the job, and provide sufficient illustrations to make you think about genuine applications in your own field of interest. I hope the process is not only beneficial but enjoyable.

Applying the ideas in this book will transform your data science practice. If you utilize even one idea from each chapter, you will be far better prepared not just to survive but to excel when faced by the challenges and opportunities of the ever expanding deluge of exploitable data.

As you use these models successfully in your own area of expertise, write and let me know. I'd love to hear from you. Contact me at info@NigelDLewis.com or visit www.AusCov.com.

Chapter 1

The Learning Problem in a Nutshell

We can only see a short distance ahead, but we can see plenty there that needs to be done.

Alan Turing

WHAT on earth is learning from data? Scientists learn from data, so do businesses, governments and charitable organizations. In fact, you would be hard pressed to identify an area in the private, public or charitable sectors where data driven models are not deployed to uncover and exploit relationships in data.

Data is all around us, Amazon makes 250,000 sales/deliveries per day, 100,000 genes can be sequenced in near real time, over 10 billion images are stored on web pages; and in a matter of months the National Health Service in the United Kingdom digitized 60 million health records. We all use data every single day, and a lot of people use it in the course of paying jobs. An analyst at a marketing company has to determine which factors she should include in her audience selection model. The researcher at the local health department measures the incidence of seasonal flu. The meteorologist runs weather models which calculate the likelihood of precipitation, change in temperature and percentage of cloud cover.

Companies in the public and private sectors need talented people who can transform massive amounts of information into actionable, strategic business decisions. Learning from data offers a set of practical techniques and tools that will help you develop robust inductive models to extract usable insights from data. Inductive, simple means the insights are derived from empirical data rather than deduced from theoretical first principles.

The overarching purpose of this book is to help you to make the leap from owning a bucketful of data to turning it into exploitable knowledge. To do this we will leverage theory to shape the way we think about data science challenges. However, this is not a text for those seeking to focus on lemmas, proofs and the minutiae of abstract theoretic detail. It's for those who want to access a hands on practical guide to a significant, successful framework for building useful predictive analytic models, to improve the well-being and profitability of the organizations they work for, and the clients they serve. At the same time, it is important to understand that data science is not a profession for those who lack curiosity or technical aptitude, but no profession dealing with empirical data ever is.

In this chapter you will learn the key difference between inductive and deductive inference, identify the three key elements of the learning problem and discover a straightforward framework for working with inductive models.

The ABCs of Inductive and Deductive Inference

A key difference, as shown in Figure 1.1, between inductive and deductive approaches revolves around the testing of hypothesis. Both approaches begin by observing a phenomenon of interest, however the focus in inductive methods is typically on selecting the best model for prediction. In the deductive approach the focus is on exploring theory, with the data used primarily to test a hypothesis about that theory. The hypothesis is either rejected or accepted based on the "*weight of evidence*" derived from the empirical data.

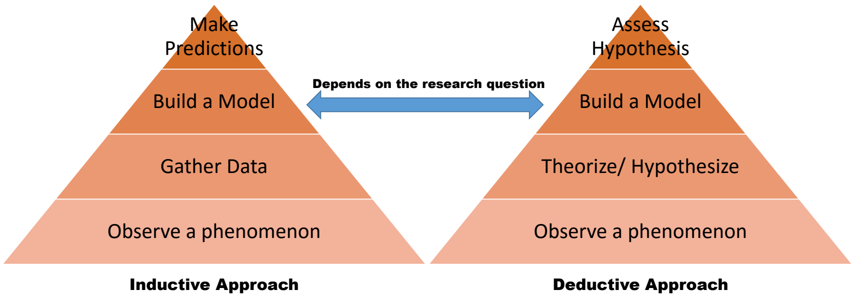


Figure 1.1: Induction and Deduction

Did This Ever Happen to You?

I recall, in my theoretical economics class, the stern warning from the professor that the "*data cannot be trusted*". It seems this experience was not just limited to my class. A famous professor of econometrics explains[1]"*A widely held view in economics is that the current untrustworthiness of empirical evidence and the inability to forecast economic phenomena is largely due to the fact that the economy is too complicated and the resulting data are too heterogeneous to be amenable to statistical modeling.*" You may have had a similar experience.

Once outside of the classroom, and in the real world of empirical analysis, I quickly discovered that provided you have sufficient data and access to the appropriate tools, a data driven inductive approach can yield significant insights.

NOTE... ✍

Successful data driven inductive models exist or are being built in every conceivable area of commerce, industry and government. From the smart phone which recognizes your voice, the robot that performs surgical procedures[2], to the detection of nuclear explosions[3], data driven models are increasingly being used to make decisions.

Unleash the Power of Induction

Whether you are working in the area of medical diagnosis, hand-written character recognition, marketing, financial forecasting, bioinformatics, economics or any other discipline that requires empirical analysis, you will frequently face the situation where underlying first principles are unknown or the systems under study are too complex to be mathematically described in sufficient detail to provide useful results. I have found an inductive data driven approach valuable in all of these situations, and so will you.

NOTE... ✍

Outside of science, deductive analytics probably reached its peak in the discipline of economics, where much of the focus (even today) revolves around testing and assessing the validity of deductive theories of the economy. In fact, the subdiscipline of statistics known as econometrics[4] grew out of the desire of economists for objective validation of theory.

The Yin and Yang of Inference

While inductive and deductive inference are quite different, they can actually be used in a complementary fashion. It is not uncommon for a researcher to plan a project which include inductive and the deductive elements.

If you have worked in the area of empirical modeling for any length of time, you might recognize this situation: You planned to carry out either a inductive or deductive project only to discover along the way that the other approach is better suited to illuminate your research question. An important point to keep in mind is that the use of an inductive or deductive approach depends in part on your data analysis objective[5].

NOTE... ✍

The relative decline in the prominence of deductive inference is partly explained by the high profile successes of data driven modeling. As Italian scholars Matteo Pardo and Giorgio Sberveglieri correctly observed well over a decade ago[6] "*...there is currently a paradigm shift from the classical modeling based on first principles to developing models from data.*" It is interesting to note the shortage of data driven modelers is now worldwide[7].

Three Key Elements of the Learning Problem

The starting point of our discussion is the formulation of the learning problem. Consider, for illustration, the supervised classification problem, where we are given the real valued attribute-response pairs (x, y). Three elements make up the basic learning problem:

1. The response or target variable has K classes, $y \in \{1, 2, ..., K\}$. Often for binary classes ($K = 2$) we assume $y \in \{0, 1\}$ or $y \in \{-1, 1\}$.

2. A probabilistic relationship is assumed to exist between the attributes (or features) x and the target y captured by an unknown probability distribution:

$$P(x, y) = P(x)P(y|x)$$

   The reason for specifying the relationship between x and y in probabilistic terms is because in practice unobserved non-deterministic factors affect the relationship. For example, in a bagel bakery slight variations in temperature, humidity, weight and texture of flour, wear of components and so on will impact the texture, taste and quality of the bagels produced. The variation in output reflects a lack of knowledge of the unobserved factors and results in a probabilistic relationship between x and y. Figure 1.2 visualizes the situation, where the impact of the unobserved factors is captured by the conditional probability that y occurs given x, written as $P(y|x)$.

3. Since data science is an inherently empirical activity, we also observe a sample of N independently, identically distributed (i.i.d) pairs $S = (x, y)$ drawn from the probability distribution $P(x, y)$. The independence assumption means that each new observation yields new and relevant information. The identical distribution means that the observations provide information about the underlying probability distribution which is fixed but unknown. Thus, given an i.i.d assumption it is possible in theory to construct a predictor that is consistent, which means that, as one gets more and more data, the predictor predictions get closer and closer to the true values.

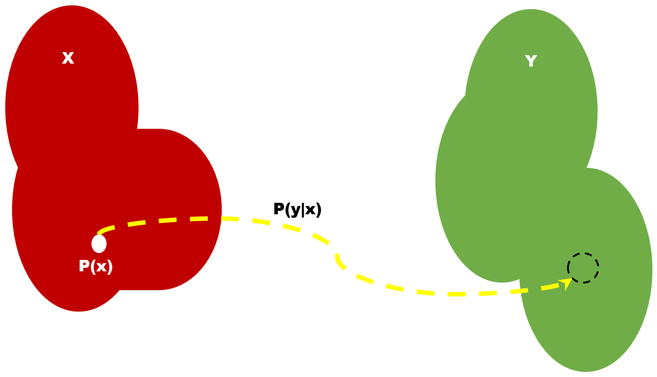


Figure 1.2: Probabilistic setting of the learning problem

The Goal of Learning from Data

Learning from data requires the ability to generalize from past observations in order to predict new circumstances that are related to past observations. The goal is to construct a good predictor $\hat{y}(\theta) = h(x|\hat{\theta})$ from the hypothesis class $\mathcal{H}$ which predicts y from x; where Θ is the vector of parameters.

NOTE... ✍

The symbol ˆ is the standard way to denote an estimate. In general, we obtain estimates of the parameter vector Θ, denoted by $\hat{\Theta}$, which are then used to calculate our predictor $\hat{y}$ from x. So for example, in the case of simple linear regression we would have $\hat{y} = h(x|\hat{\theta}) = \hat{\theta}_1 + \hat{\theta}_2 x$.

The predictor or learner $\hat{y}$ is a *hypothesis* about the true function which generated the data. The hypothesis class $\mathcal{H}$ is the set of functions we allow our algorithm to consider. Its selection requires prior knowledge known as inductive bias. Inductive bias is anything that causes an inductive learner to prefer some hypotheses over other hypotheses. It consists of some fundamental assumption or set of assumptions that the learner makes about the target function that enables it to generalize beyond the training data. For example:

- If $\mathcal{H}$ represents linear regression, then $\hat{y}$ assumes the relationship between the attributes x and the target y is linear.
- In the case of the simple regression $y = \alpha + \beta x^q$; the data scientist may only wish to consider those functions where $\alpha > 0$, $3 \leq \beta \leq 7$ and $q > 1.5$.
- In an optimization algorithm $\mathcal{H}$ consists of the space the algorithm is allowed to search.

NOTE... ✍

Inductive bias is a critical element of your data science practice because as Jonathan Baxter of the London School of Economics explains[8] "*Probably the most important problem in machine learning is the preliminary biasing of a learner's hypothesis space so that it is small enough to ensure good generalisaton* [ability to predict] *from reasonable training sets, yet large enough that it contains a good solution to the problem to be learnt.*"

Clarify the Selection Criterion

To achieve our goal, we need a criterion to choose between the other competing hypotheses in $\mathcal{H}$. Let us denote $L(\hat{y}, y)$ to be the error committed by $\hat{y}$ in predicting y. If $L(\hat{y}, y)$ is large then $\hat{y}_i$ does a poor job of predicting y_i. If $L(\hat{y}, y)$ is small then $\hat{y}_i$ does a good job at predicting y_i.

A common loss function for binary classification labeled so that $y \in \{-1, 1\}$ is:

$$L(\hat{y}, y) = I\left[-y_i \hat{y}_i(\theta) > 0\right]$$

This is an indicator function that takes the value 1 if the sign of y_i is different from the sign of $\hat{y}_i(\theta)$.
Another choice is the squared error loss (known as L2 loss or the least squares method) familiar to you if you have built traditional linear regression models:

$$L(\hat{y}, y) = \left[\hat{y}_i(\theta) - y_i\right]^2,$$

The expected error is defined as:

$$R(\theta) = \frac{1}{2} \int L(\hat{y}, y) P(x, y) dx dy \tag{1.1}$$

The quantity $R(\theta)$ is usually called the expected risk and we will want to select a model that minimizes it. However, since $P(x, y)$ is unknown $R(\theta)$ cannot be directly observed. Instead empirical risk $R_{emp}(\theta)$ calculated from the training sample is minimized.

NOTE... ✍

Since the squared error loss is sensitive to outlying observations the absolute loss function (known as L1 loss) is often specified:

$$L(\hat{y}, y) = \left|\hat{y}_i(\theta) - y_i\right|$$

In this case:

$$R_{emp}(\theta) = \frac{1}{2N} \sum_{i=1}^{N} \left|\hat{y}_i - y_i\right| \tag{1.2}$$

Choose the Learning Task

Now that we have the learning framework in place, we can turn our attention to the actual tasks we, as a Data Scientists, perform. Rather fortunately, it turns out learning from data tasks can be divided neatly into three basic types of activity:

1. Classification or estimation of class decision boundaries. For example, classification of production line eggs by size and color.

2. Regression or the estimation of an unknown continuous function. For example, predicting the average value created by a local musical festival.

3. Probability density estimation. For example, estimating the distribution of Northern pike in Irish coastal rivers.

Throughout this text we will focus primarily on classification because it is the most frequent task faced by the data scientist. However, the lessons we draw are useful for all three types of task.

NOTE... ✍

The framework for the learning problem is akin to the model based approach to statistical induction and statistical inference pioneered in the early part of the last century[9]. In a statistical approach the theory of optimal point estimation is used to obtain estimates of the parameter vector $\hat{\Theta}$; then significance testing using p-values and/ or confidence interval are used to assess $\hat{\Theta}$. In machine learning, the *emphasis* tilts more towards assessing the accuracy of $\hat{y} = h(x|\hat{\theta})$ for predicting y. Although in actual practice a blend of the two approaches is common. In both approaches y is evaluated based on unseen test data.

Notes

[1]Spanos, Aris. "Learning from data: The role of error in statistical modeling and inference." Unpublished manuscript (2012).

[2]This is a very exciting area of applied research. If it is of interest to you see:

- Rosen, Jacob, Blake Hannaford, and Richard M. Satava. "Surgical Robotics." Springer Science+ Business Media, LLC (2011).
- O'toole, Michael D., et al. "A methodology for design and appraisal of surgical robotic systems." Robotica 28.02 (2010): 297-310.
- Taha, Zahari, A. Salah, and J. Lee. "Bone breakthrough detection for orthopedic robot-assisted surgery." APIEMS 2008 Proceedings of the 9th Asia Pacific Industrial Engineering and Management Systems Conference. 2008.

[3]See:

- Kortström, Jari, Marja Uski, and Timo Tiira. "Automatic classification of seismic events within a regional seismograph network." Computers & Geosciences 87 (2016): 22-30.
- Dong, Longjun, et al. "Discrimination of Mine Seismic Events and Blasts Using the Fisher Classifier, Naive Bayesian Classifier and Logistic Regression." Rock Mechanics and Rock Engineering 49.1 (2016): 183-211.

[4]Essentially, statistics for economic analysis. In this approach, empirical models are deduced from theoretical principles, and then used with data to "test" theories. Other social science disciplines use a similar approach.

[5]If you wish to delve deeper, here are three classics:

- Laudan, Larry. Science and hypothesis: Historical essays on scientific methodology. Dordrecht,, Holland: D. Reidel, 1981.
- Reichenbach, Hans. The rise of scientific philosophy. Univ of California Press, 1973.
- Mill, John Stuart. A System of Logic Ratiocinative and Inductive: Being a Connected View of the Principles of Evidence and the Methods of Scientific Investigation. Bombay, 1906.

[6]Pardo, Matteo, and Giorgio Sberveglieri. "Learning from data: A tutorial with emphasis on modern pattern recognition methods." Sensors Journal, IEEE 2.3 (2002): 203-217.

[7]Here are a few fairly typical comments on the situation:

- Business wire reported “The data science workforce shortage means there’s simply not enough talent to handle the immense amount of raw data coming through an organization’s doors each day.” See http://www.businesswire.com/news/home/20160111005886/en/Booz-Allen-Addresses-Data-Science-Talent-Shortage

- The globally distributed and widely read IEEE Spectrum magazine reported “*the demand for data scientists is exceeding the supply. These professionals garner high salaries and large stock option packages...*” See the report by Emily Waltz. Is Data Scientist the Sexiest Job of Our Time? IEEE Spectrum. September 2012. Also available at http://spectrum.ieee.org/tech-talk/computing/it/is-data-scientist-the-sexiest-job-of-our-time. The IEEE is the world’s largest professional organization devoted to engineering and the applied sciences. It has over 400,000 members globally.

- According to the McKinsey Global Institute, the United States alone faces a shortage of 140,000 to 190,000 data scientists with the appropriate skills. Whilst the Harvard Business Review declared data science as the *sexiest* job of the 21st century. See the special report by James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, Angela Hung Byers. Big data: The next frontier for innovation, competition, and productivity. McKinsey Global Institute. May 2011. Also available at http://www.mckinsey.com/insights/business_technology/big_data_the_next_frontier_for_innovation.

- Techcrunch ran with the headline:How To Stem The Global Shortage Of Data Scientists. see http://techcrunch.com/2015/12/31/how-to-stem-the-global-shortage-of-data-scientists/

[8]Baxter, Jonathan. "Learning internal representations." Proceedings of the eighth annual conference on Computational learning theory. ACM, 1995.

[9]See for example:

- Fisher, Ronald Aylmer. "On the mathematical foundations of theoretical statistics." Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character (1922): 309-368.

- Neyman, Jerzy. "Outline of a theory of statistical estimation based on the classical theory of probability." Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences 236.767 (1937): 333-380.

Chapter 2

Supervised Learning

Data Science is a series of failures punctuated by the occasional success.

N.D Lewis

SUPERVISED learning has been used in a wide variety of applied applications. For example, intrusion detection in computer networks[10], hyperspectral band selection[11],non-technical losses detection in power distribution systems[12] and automatic aquatic weed recognition[13]. It is defined as learning in the presence of "*ground truth*". This simply means that the training data are labeled with respect to their class membership. For example, in building a model to determine the quality of china plates each example in your sample would consist of the relevant attributes x (material, temperature fired, and so on), and the actual observed target values y (bone, fine, porcelain).

The basic goal of supervised learning is to learn the predictor $\hat{y}$ given a training sample of $\{x_i, y_i\}$. Given a set of such pairs, a learning algorithm constructs a predictor $\hat{y}$ mapping instances to labels. The predictor $\hat{y}$ can then be used to examine unseen values of the attribute x and predict the appropriate value of y. If y is real valued, we have the regression task; if y takes values from an unordered finite set, we have pattern classification.

The Essential Ingredient for Effective Classification

Classification is perhaps the most frequently encountered decision making task when learning from data. A classification problem occurs when an object (y) of interest needs to be assigned into a predefined finite group or class based on a number of suitably chosen attributes (x) related to that object. A typical example is the identification of wine using a guide book of features such as place of origin, vinification method and style, sweetness, vintage, and varietal used in production. In this case a learning algorithm is trained on a sample using the attributes to determine the class to which a specific wine belongs. Once trained it is used to classify previously unseen wine to a class on the basis of observed value of the attributes. Since evidence can be uncertain and matches partial, the output is often given in terms of rank order or probability that an object belongs to a specific class.

The essential characteristic of the classification problem is that the learning algorithm assigns an object to one of K predefined classes. This does not mean that the "*correct answer*" is necessarily one of those classes. It is quite possible that in designing the classes, the data scientist missed out an important class or group of classes. For example, in a regional classification of wine task, a non-wine drinker might inadvertently exclude Australia[14], or Texas[15], leading to a design induced misclassification of such wines. A typical learning algorithm will only attempt to match the data against the given classes. Poorly designed classes will impact the effectiveness of the classifier. This is why it is particularly important in the design of a data science solution to include domain experts in the process.

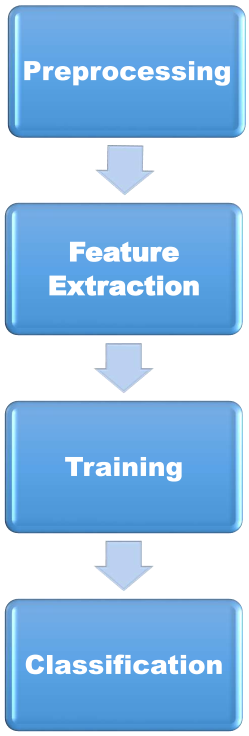


Figure 2.1: A typical approach to classification

Figure 2.1 illustrates a typical approach to the classification problem. Once the problem has been formulated and the initial sample collected. The data is preprocessed and features extracted. This is followed by training of the classifier; once trained it is used in the classification task.

The Answer to How to Determine the Hypothesis Class

In practice you will frequently encounter both linearly separable and non-linearly separable observations. For linearly separable data, and two classes for instance, there exists a hyperplane which divides the input space such that all points of the first class are in one region and those of the second class are in the other region. In two dimensions this is simply a straight line which separates the input space by class.

Take a look at the left panel of Figure 2.2. It would appear the data are linearly separable so a linear decision rule might provide a good choice. Several alternative linear classifiers, $\{\hat{y}_1, \hat{y}_2, \hat{y}_3\}$ are shown in the right panel. All three are consistent given the observations, but each represents an alternative hypothesis regarding the decision boundary.

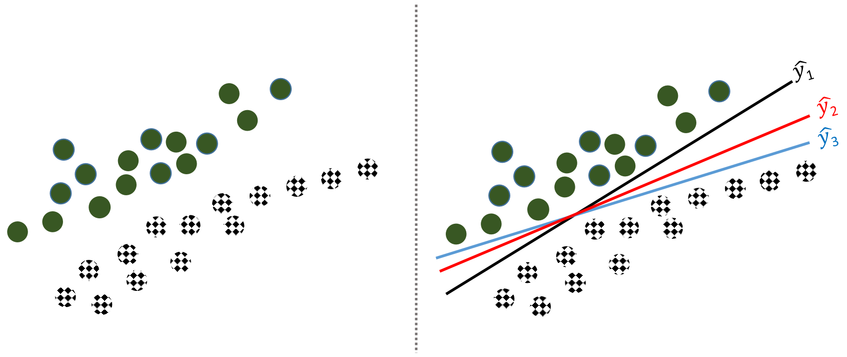


Figure 2.2: Illustration of linearly separable observations

Figure 2.3 (left panel) illustrates the situation where the data are non-linearly separable. In this situation you cannot draw a straight line so that all the "*solid circles*" are on one side, and all the "*checked circles*" are on the other side. The right panel illustrates that decision regions may also be discontinuous.

The accuracy of supervised algorithms varies as a function of the training set characteristics properties including the sample size[16]. In general, classification accuracy tends to improve as the size of the training sample is increased[17]. Therefore, it is important that the training samples be large enough to provide a representative and unbiased description of the class properties. As a general rule of thumb, the more observations you have the more complex can be your hypothesis class, because you will have more empirical evidence to support it. The hypothesis class depends on the characteristics of your data.

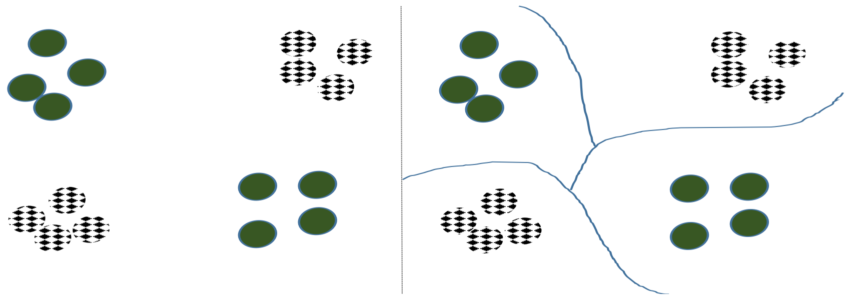


Figure 2.3: Illustration of non-linearly separable observations

The Two Core Approaches to Supervised Learning

Supervised classification can follow two main lines - Classification using generative algorithms via the use of the Bayes rule; or classification via discriminant analysis where the boundaries are directly learnt from data.

The Key to Generative Algorithms

Generative algorithms explicitly model the joint probability distribution $P(y,x)$. In order to do this, they require an estimate of the conditional distribution $P(x|y)$. Once this is obtained a predictive distribution for the target y conditional on the attribute x can be calculated by applying Bayes rule:

$$P(y|x) = \frac{P(x|y)P(y)}{\int_y P(x|y)P(y)dy}$$

Notice that $P(x|y)P(y)$ defines the joint distribution we first met in the set up to the learning problem on page 10. The key thing to be aware of is that generative algorithms try to estimate explicitly how the sample x_i has been generated.

NOTE... ✍

Bayes rule, named after English statistician, philosopher and Presbyterian minister Thomas Bayes, states that given two random variables A and B:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

It is common to refer to $P(A|B)$ as the posterior probability, $P(A)$ and $P(B)$ as prior probabilities and $P(B|A)$ as the likelihood.

Understanding Discriminative Algorithms

In discriminative algorithms the conditional distribution $P(y|x)$ is modeled directly. It turns out that this is all that is needed for classification. Because discriminative algorithms only model the conditional distribution they can have a much simpler structure than generative based models. Some discriminative classifiers make things even simpler by restricting their focus to specific values. For example, the support vector machine focuses on whether $P(x|y)$is greater or less than 0.5.

What is a Bayes Classifier?

A Bayes classifier is a generative classifier that can be used to solve the classification learning task discussed on page 14. To begin let us denote $x = [x_1, ...x_n]$ as the input vector of features/ attributes, and the K classes by $C = \{c_1, c_2, ..., c_K\}$. A

Bayes classifier is a decision rule to estimate the most probably class c_i for an observation given the set of input features or attributes x. It makes good use of Bayes rule.

Conceptually, Bayes rule provides a mechanism to go from the conditional probability of the evidence provided by the input features given the classes $P(x|c_i)$, to the conditional probability of the class given the evidence provided by the attributes $P(c_i|x)$. This is important because in practice, you will often know how frequently some particular evidence occurs, given a known class or outcome. For example, in medical research, the relative frequency of a disease, and the probability of a medical test being positive given the disease are often known. A patient with a positive test result is more interested in the probability of the disease, given that the medical test is positive. Bayes rule allows the computation of this probability.

NOTE... ✍

The probability of observing x given class c_i, denoted by $P(x|c_i)$, are often called the class conditional probabilities. The probabilities of occurrence of different classes, denoted by $P(c_i)$, are generally called the class prior probabilities or base rates.

Since by Bayes rule $P(c_i|x) \propto P(c_i)P(x|c_i)$, the Bayes classifier chooses a class by assigning x to c_i if:

$$P(c_i)P(x|c_i) > P(c_j)P(x|c_j)\ \forall j \in \{1, ..., K\};\ j \neq k$$

The intuition behind multiplying $P(x|c_i)$ by $P(c_i)$ is to give a higher weight to more frequently occurring classes, and lower weight to less likely classes.

A good classification procedure is one that minimizes the costs of misclassification. It turns out the Bayes classifier minimizes loss when risk is defined as the probability of misclassification. It gives the smallest possible misclassification error known as the Bayes error rate. The Bayes rule is the optimal classification rule if the underlying distribution of the data is known.

NOTE... ✍

Vapnik's principle[18] favors the use of discriminative models. It states that: *"When trying to solve some problem, one should not solve a more difficult problem as an intermediate step."* Discriminative algorithms directly estimate the labels; whilst generative models estimate the density of x as an intermediate step.

A Lower Bound on Error

The Bayes error rate (ϵ_{Bayes}) gives a statistical lower bound on the error achievable for a given classification problem and attribute set[19]. This is very appealing for practical purposes because it provides a way to quantify how effective a given classifier is. The Bayes error rate is the "gold standard" for a classifier to attain. However, it does depend on the attributes under consideration, sample size and complexity of the decision surface. It will be higher if only some of the relevant attributes are included, and non-zero if classes overlap.

This is all very exciting, we have a gold standard that we can deploy in our everyday data science practice; all that is left for us to do is the calculation. The probability of error in the Bayes classifier is given by:

$$\epsilon_{Bayes} = \sum_{i=1}^{K} P(c_i) \int_{\mathcal{L}_i^K} P(x|c_i)dx, \tag{2.1}$$

where $\mathcal{L}_i$ is the region where x is classified to c_i, $\mathcal{L} = \cup_{i=1}^{K} \mathcal{L}_i$, and the complement of $\mathcal{L}_i$ is given by $\mathcal{L}_i^K = \mathcal{L} - \mathcal{L}_i$.

Two Easy Tricks to Evaluate the Bayes Error

Unfortunately, obtaining the Bayes error from equation 2.1 entails evaluating the multidimensional integral of possibly unknown multivariate density functions over unspecified regions ($\mathcal{L}_i^K$). Alas, with real world data it is often intractable. In order to estimate the conditional densities a large number of observations are required. The situation gets more difficult as the number of classes increases. As a consequence, in practice we need a trick of some sort. Here are two good ones:

The Mahalanobis Trick

The quickest method[20] that I know about is very easy to use. It is a trick to keep in your back pocket to be used when you need a quick and dirty approximation for a two class problem. I like to refer to it as the "*Mahalanobis trick*"[21]. Here is why, let Σ be the covariance matrix[22] weighted by the class prior probabilities:

$$\Sigma = P(c_1)\Sigma_1 + P(c_2)\Sigma_2,$$

where Σ_1 and Σ_2 are the covariance matrices of the first and second class respectively. Suppose the mean vector for each class are give by μ_1 and μ_2 then the Mahalanobis distance, which measures the closeness (or similarity) between two objects, can be quickly calculated as:

$$\mathcal{D}_M = (\mu_1 - \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2), \qquad (2.2)$$

A bound on the Bayes error can now be calculated as:

$$\epsilon_{Bayes} \leq \frac{2P(c_1)P(c_2)}{1 + P(c_1)P(c_2)\mathcal{D}_M}.$$

This is a neat little approximation because we don't need to say anything about the probability distribution of the two classes.

It is also easy to calculate requiring only the sample means, sample covariances and class priors.

NOTE... ✍

In the general case the Mahalanobis distance is calculated as:

$$\mathcal{D}_M = (x-\mu)^T \, \Sigma^{-1} \, (x-\mu) \, ,$$

where x is the vector or matrix of attributes and μ is the mean vector. Mahalanobis distance can be calculated in R using the `mahalanobis` function in the `stats` package.

The Bhattacharyya Trick

A second approach makes use of the Bhattacharyya distance:

$$\mathcal{D}_B = -\ln \int \sqrt{P(x|c_1)P(x|c_2)dx}$$

Provided the class densities are Gaussian the Bhattacharyya distance simplifies to:

$$\mathcal{D}_B = \frac{1}{8}\,(\mu_2-\mu_1)^T \left(\frac{\Sigma_1+\Sigma_2}{2}\right)^{-1} (\mu_2-\mu_1)$$

$$+\frac{1}{2}\ln \frac{\left|\frac{\Sigma_1+\Sigma_2}{2}\right|}{\sqrt{|\Sigma_1|\,|\Sigma_2|}}$$

Notice that the second term disappears in the case of equal covariances i.e. $\Sigma_1 = \Sigma_2$. We can then use the Bhattacharyya distance to provide a bound on the Bayes error:

$$\frac{1}{2}\left(1-\sqrt{1-4P(c_1)P(c_2)\exp[-2\mathcal{D}_B]}\right) \leq \epsilon_{Bayes}$$

$$\leq \exp[-\mathcal{D}_B]\sqrt{P(c_1)P(c_2)}$$

It is slightly more complicated to calculate than the *Mahalanobis trick*, and it also requires knowledge of the class probability density functions. The advantage is that it provides tighter bounds.

NOTE... ✍

Bhattacharyya distance can be calculated using the `bhattacharyya.dist` from the `fpc` package. For example, suppose $\mu_1 = 4$ and $\mu_2 = 6$ with $\Sigma_1 = \Sigma_2 = 1$ then:

```
require(fpc)
bhattacharyya.dist(4,6,1,1)
[1] 0.5
```

How to Unleash the Power of the Naive Bayes Classifier

The naive Bayes classifier is an approximation to the Bayes classifier. Its construction relies on two main assumptions; First, an absence of hidden or latent attributes; Second all attributes are independent of each other given the class so that:

$$P(x_1, ..., x_n|c_j) \approx \prod_{i=1}^{n} P(x_i|c_j)$$

This assumption reduces the number of parameters to be estimated; and we pick the class with the highest probability i.e. where:

$$\arg \max_{k\in\{1,...,K\}} P(c_j) \prod_{i=1}^{n} P(x_i|c_j)$$

NOTE... ✍

The assumption of conditional independence may appear to be rather a strong claim to make about your data. I often think of it a bit like the requirement in linear regression that the explanatory variables be independent. Despite this unrealistic assumption linear regression has found many useful real world applications.

This classifier is naive in the sense that it makes the strong assumption that the attributes are mutually conditionally independent. This is rarely true in practice. However, naive Bayes classifiers have worked well in many real-world classification situations, such as text classification[23], sentiment analysis[24], medical screening[25] and fault detection[26]. The assumption of mutually conditionally independence appears not to significantly compromise prediction accuracy. Indeed, University of Washington Professor Pedro Domingos and scholar Michael Pazzani have shown the optimality of the naive Bayes classifier even when the conditional independence assumption is significantly violated[27].

An Example that Will Build Your Intuition

If you have ever been to London, you will have noticed that the buses tend to have two levels, and the London cabs are "*boxy*", black and surprisingly roomy. Suppose you have been asked to develop a vehicle identification system using the following vehicle attributes:

1. Whether the vehicle is short;
2. whether the vehicle has four doors;
3. whether the vehicle is black.

Furthermore, a labeled sample of 1000 has been collected containing a total of 300 black cabs, 350 Busses, and 350 regular cars, as shown in Table 1.

Type	Short	Not Short	Four Doors	Not Four Doors	Black	Not Black
Cab	200	100	150	150	225	75
Bus	50	300	25	325	5	345
Car	300	50	250	100	100	250

Table 1: Attributes and vehicle object type

Now suppose you are presented with an unknown vehicle with the following attributes - "*Short*", "*Four Door*" and "*Black*". Is it a cab, bus or regular car?

Using the empirical sample, we can easily calculate the prior probabilities. We have:

1. $P(Cab) = \frac{300}{1000} = 0.3,$
2. $P(Bus) = \frac{350}{1000} = 0.35,$
3. $P(Car) = \frac{350}{1000} = 0.35.$

So, our prior probabilities for vehicle type lean towards "*Car*" or "*Bus*". Our intuition tells us "*Car*" or "*Cab*".

Let's calculate the prior probabilities for the evidence given by the attributes:

1. $P(Short) = \frac{200+50+300}{1000} = 0.55,$
2. $P(Four\ Door) = \frac{150+25+250}{1000} = 0.425,$
3. $P(Black) = \frac{225+5+100}{1000} = 0.33.$

Next, we need to calculate the likelihoods for each group of attributes. First, for "*Short*":

1. $P(Short|Cab) = \frac{200}{300} = 0.667,$
2. $P(Short|Bus) = \frac{50}{350} = 0.143,$
3. $P(Short|Car) = \frac{300}{350} = 0.857.$

Next, for "*Four Door*":

1. $P(Four\ Door|Cab) = \frac{150}{300} = 0.5,$
2. $P(Four\ Door|Bus) = \frac{25}{350} = 0.071,$
3. $P(Four\ Door|Car) = \frac{250}{350} = 0.714.$

Finally, for "*Black*":

1. $P(Black|Cab) = \frac{225}{300} = 0.75,$
2. $P(Black|Bus) = \frac{5}{350} = 0.014,$
3. $P(Black|Car) = \frac{100}{350} = 0.286.$

Now all we need to do is calculate $P(c_j)\prod_{i=1}^{N} P(x_i|c_j)$ for each of the three vehicle classes and pick the largest values as the most likely. For example:

$P(Cab|Short, Four\ Door, Black) = P(Cab) \times P(Short|Cab) \times P(Four\ Door|Cab) \times P(Four\ Door|Cab)$
$= 0.3 \times 0.667 \times 0.5 \times 0.75$
$= 0.075.$

Using a similar method, we find that:

- $P(Bus|Short, Four\ Door, Black) = 0.01$
- $P(Car|Short, Four\ Door, Black) = 0.061$

Finally, since $P(Cab|Short, Four\ Door, Black)$ is the largest value, we assign the unknown observation to that class.

Incredibly Simply Ways to Build the Naive Bayes Classifier in R

Now we are ready to build the naive Bayes classifier in R. To do this, I will walk you through two examples. The first involves simulated data on two attributes, and the second example is from a real life study involving the analysis of thyroid data.

A Simulated Example

Let's use the naive Bayes model to predict the labels shown in Figure 2.4 of two simulated attributes.

The naive Bayes model can be estimated using the `e1071` package. First, load the package:

```
library (e1071)
```

Next, generate 100 examples using two attributes x1 and x2 from a Gaussian distribution:

```
num_attrib <- 2
N <- 100
set.seed(2017)
x <- matrix(rnorm(N*num_attrib),ncol=num_
   attrib)
colnames(x) <- c("x1","x2")
y <- as.numeric((x[,1]^2+x[,2]^2) > 2.3)
```

Let's go through this line by line. The first line specifies the number of attributes, in this case 2. The second line uses `N` to set the sample size equal to 100. Next the `set.seed` method is used to ensure you can reproduce the example. The matrix `x` contains the values of the Gaussian attributes; and the `colnames` method is used to name the attributes `x2` and `x2`. The r object `y` contains the class labels in the form of "`0`" or "`1`".

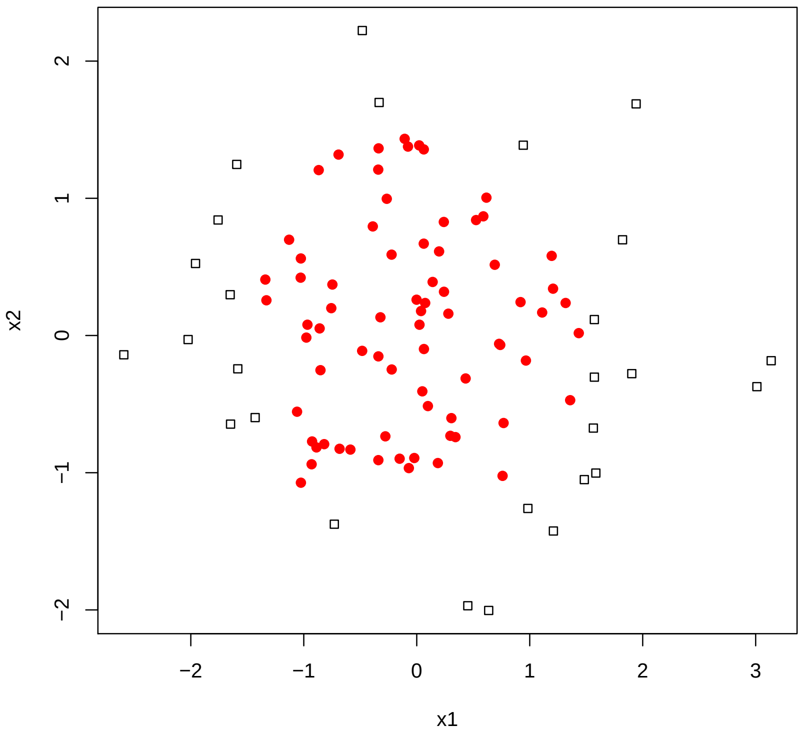


Figure 2.4: Simulated values

You can see the first six class labels in `y` by typing:

```
head(y,6)
[1] 0 0 0 1 0 1
```

The first three instances belong to class "`0`" whilst the forth, and sixth belong to class "`1`".

Take a look at the class of `y`:

```
class(y)
[1] "numeric"
```

It is numeric, it is good practice to transform it into a factor. Here is how to do that:

```
y<-as.factor(y)
```

A little housekeeping to combine the attributes and class labels into the object `data`. I use `as.data.frame` to transform `data` from a matrix to a dataframe. This just makes it a little easier to use with the `naiveBayes` method:

```
data<-cbind(y,x)
data<-as.data.frame(data)
```

Now we are ready to fit the model:

```
fit <- naiveBayes(x,y)
```

The model is stored in the R object `fit`. Our primary task is to predict the class labels using all of the data. This can be achieved using the `predict` method:

```
pred_probs<- predict(fit, data[,-1],type ="
   raw")
```

Notice the use of `type ="raw"` this returns the posterior probabilities. Here are the first few:

```
head(pred_probs)
0 1
[1,] 0.5727871 0.4272129
[2,] 0.8290129 0.1709871
[3,] 0.8437831 0.1562169
[4,] 0.4851285 0.5148715
[5,] 0.8252904 0.1747096
[6,] 0.3305669 0.6694331
```

The first observation has a posterior prob of 0.57 for class 0 and 0.43 for class 1. The predicted label would be class 0. The sixth observation has a probability of 0.33 for class 0 and 0.66 for class 1, so class 1 is the recommended class label.

It is often helpful to view the predicted class labels. To do this set `type="class"`:

```
pred<-predict(fit, data[,-1],type="class")
head(pred)
```

```
[1] 0 0 0 1 0 1
Levels: 0 1
```

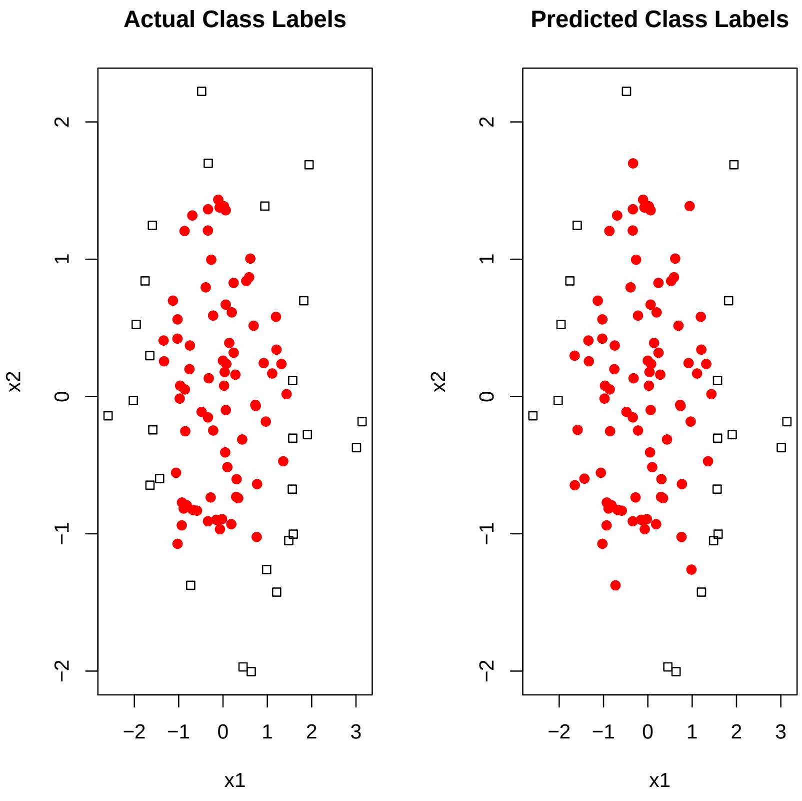


Figure 2.5: Actual and predicted class labels.

Figure 2.5 shows the actual and predicted class labels. It appears the model predicts the labels very well. This is confirmed by the confusion matrix:

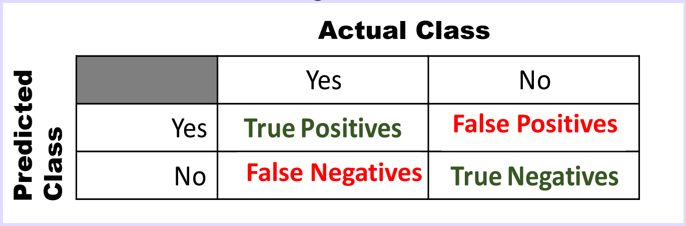
```
table(pred, data$y)

pred  1  2
   0 73  8
```

```
    1   0 19
```

Of course, in using all of the data to build the model we should not be surprised to obtain a good fit. We discuss this issue further in chapter 6.

NOTE...

A confusion matrix is a table that contains information about actual and predicted classifications. It is often used to summarize the performance of a classifier. It has the general form:



Actual Class

Predicted Class	Yes	No
Yes	True Positives	False Positives
No	False Negatives	True Negatives

Misclassification occurs in the false positive and false negative quadrants.

Analysis of Thyroid Gland Data

The `thyroid` data frame[28] in the `mclust` package was constructed from five laboratory tests administered to a sample of 215 patients. The tests were used to predict whether a patient's thyroid function could be classified as euthyroidism (normal), hypothyroidism (under active thyroid) or hyperthyroidism (overactive).

The thyroid data frame contains the following variables:

- **Class variable:**
  - `Diagnosis`: Diagnosis of thyroid operation: Hypo, Normal, and Hyper.

  * Distribution of `Diagnosis` (number of instances per class)
  * Class 1: (normal) 150
  * Class 2: (hyper) 35
  * Class 3: (hypo) 30

- **Continuous attributes:**

  - `RT3U`: T3-resin uptake test (percentage).
  - `T4`: Total Serum thyroxin as measured by the isotopic displacement method.
  - `T3`: Total serum triiodothyronine as measured by radioimmuno assay.
  - `TSH`: Basal thyroid-stimulating hormone (TSH) as measured by radioimmuno assay.
  - `DTSH`: Maximal absolute difference of TSH value after injection of 200 micro grams of thyrotropinreleasing hormone as compared to the basal value.

Figure 2.6 shows the distribution of class labels and box plots of the scaled attributes.

Our first step is to load the packages and data:

```
library (e1071)
data("thyroid",package="mclust")
```

Next we store the thyroid data in the R object `data`, count the number of observations using `nrow` and store the result in `N` ; next a training sample of 150 observations selected at random without replacement is stored in the R object `train`:

```
data<-thyroid
set.seed(2016)
N=nrow(thyroid)
train<-sample(1:N,150,FALSE)
```

Notice that `train` does not contain the actual examples. It is an integer that points to the row locations of the sample. To see this type:

```
class(train)
[1] "integer"
```

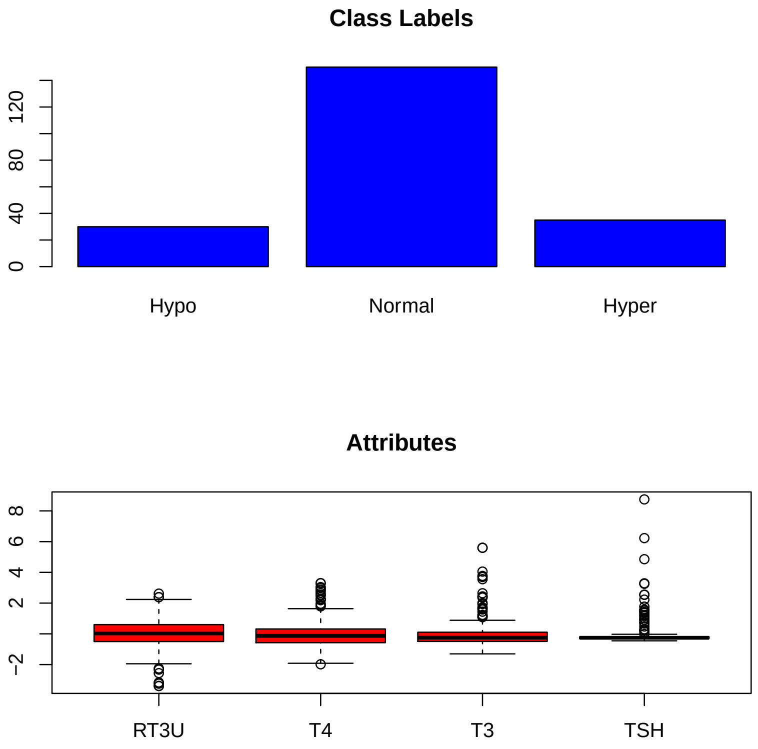


Figure 2.6: Distribution of class labels (top) and box plots (bottom) for the `thyroid` data frame.

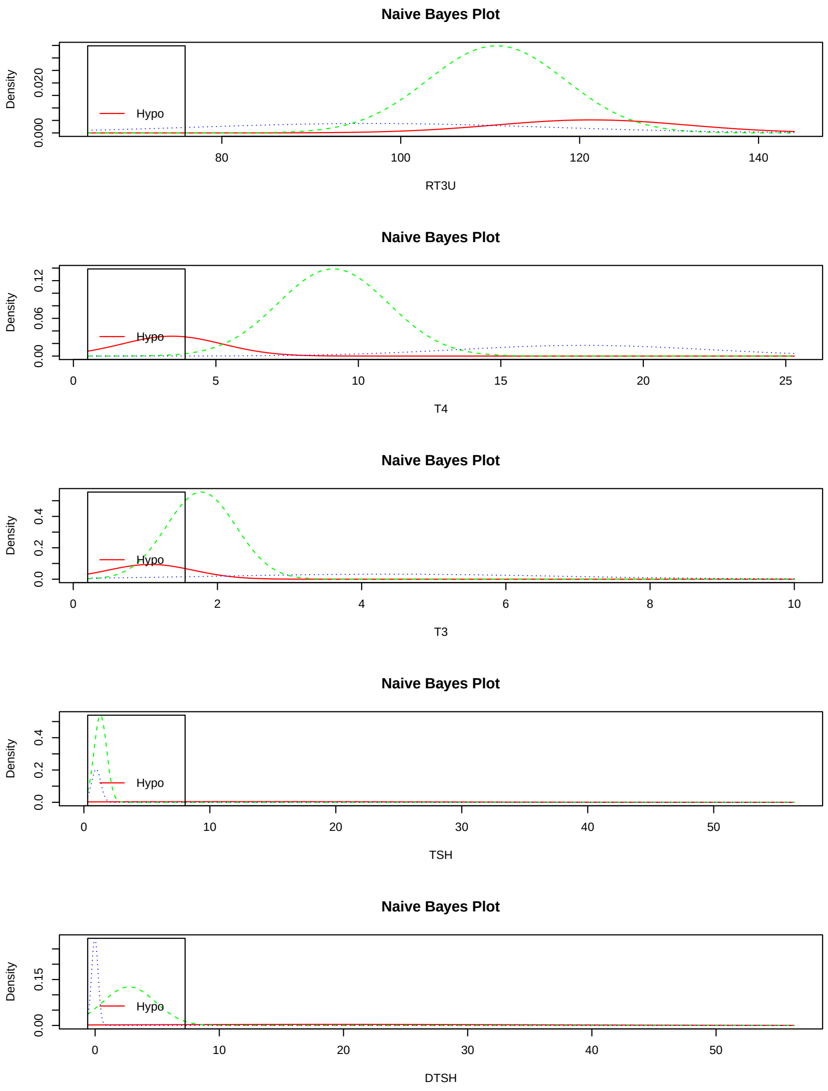


Figure 2.7: Naive Bayes marginal probabilities for `thyroid` data frame.

The first randomly selected example is contained in row 39 of train, the second in row 31, and the third in row 180 and so on:

```
head(train)
[1]  39  31 180  29 101  26
```

Now we are ready to fit the naive Bayes model:

```
fit <- naiveBayes(Diagnosis ~ ., data =
   data[train,])
```

Figure 2.7 visualizes the marginal probabilities of the attribute variables given the class.

Let's explore some of the attributes of the R object `fit`:

```
attributes(fit)

$names
[1] "apriori" "tables"  "levels"  "call"

$class
[1] "naiveBayes"
```

The parameter `apriori` contains the class distribution. To see this type:

```
fit$apriori
Y
  Hypo Normal  Hyper
    21    101     28
```

The parameter `tables` contains the mean and standard deviation of the attributes given the target class[29]. For example, for the attribute `RT3U` you will see:

```
fit$tables
$RT3U
        RT3U
Y             [,1]      [,2]
  Hypo   121.28571 10.715143
  Normal 110.71287  7.704981
  Hyper   96.67857 19.733509
```

Here is how to read the table, conditional on the class Hypo the attribute RT3U has a mean of 121.28 and a standard deviation of 10.71.

Now to the predictions:

```
pred<-predict(fit, data[-train,-1],type="
   class")
head(pred,4)
[1] Normal Normal Normal Normal
Levels: Hypo Normal Hyper
```

Using the head method, we see the first four predictions are all for the Normal class.

The interesting thing about naive Bayes is that the actual probability estimates are often pretty poor[30]. You will frequently find them clustered either close to zero or close to 1. This is a direct consequence of the failure of the independence assumption. Let's look at the first few posterior probabilities in our example, using type="raw":

```
pred_probs<-predict(fit, data[-train,-1],
   type="raw")
head(pred_probs,4)
             Hypo    Normal        Hyper
[1,] 2.393334e-09 0.9153993 8.460072e-02
[2,] 4.553268e-04 0.9995447 1.546719e-10
[3,] 3.143168e-05 0.9944790 5.489571e-03
[4,] 1.058918e-06 0.9999989 2.012965e-44
```

Fortunately, classification accuracy can be quite high even when the posterior estimates are at extreme values. The confusion matrix is given by:

```
table(pred, data$Diagnosis[-train])

pred      Hypo Normal Hyper
  Hypo       8      0     0
  Normal     1     49     0
  Hyper      0      0     7
```

The thing to remember is that classification using class-conditional density estimation exploits the greatest amount of information—the posterior probabilities of class membership and this often results in a very good classifier.

How to Leverage the Value of the k-Nearest Neighbors Algorithm

It is not always possible to assume a specific form for the class-conditional densities. In such circumstances we can use non-parametric density estimation. Non-parametric means the algorithm does not make any assumptions about the underlying data distribution.

The k-Nearest Neighbors algorithm[31] (kNN) is a popular non-parametric lazy learning algorithm where examples are classified based on the class of their nearest neighbors. It is considered a lazy algorithm because it does not use the training data sample to do any generalization. This means the training phase is very fast.

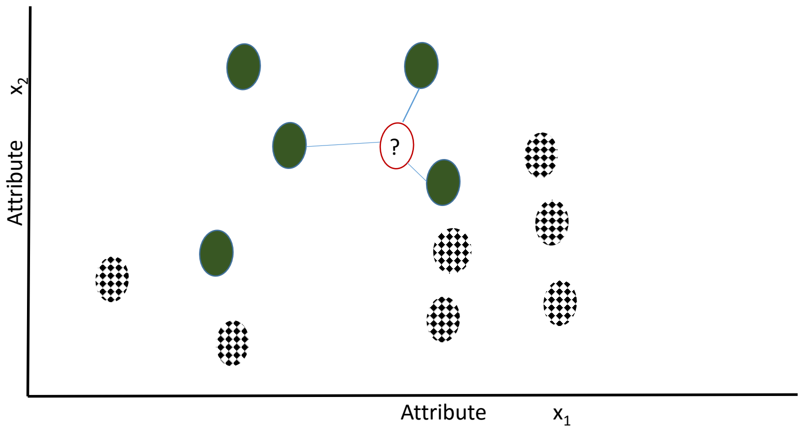


Figure 2.8: Simple illustration of kNN

The basic idea is illustrated in Figure 2.8 where the class of the empty oval needs to be predicted. kNN calculates the distance between samples with the class of the unknown object typically determined by a majority vote. In the case where k=3, the three nearest neighbors are *solid oval.* The predicted class for the empty oval is therefore *solid oval.*

The distance function $\mathcal{D}_{ij}$ is used to measure the closeness between observations i and j. Euclidean distance is often used for continuous attributes. It is calculated as:

$$\mathcal{D}(x_i, x_j) = \sqrt{\sum_{m=1}^{n} (x_{im} - x_{jm})^2}$$

where n is the number of features. Other popular distance metrics for continuous attributes include:

1. Manhattan/ city block: $\mathcal{D}(x_i, x_j) = \sum_{m=1}^{n} |x_{im} - x_{jm}|$
2. Minkowski: $\mathcal{D}(x_i, x_j) = \left(\sum_{m=1}^{n} (|x_{im} - x_{jm}|)^q\right)^{\frac{1}{q}}$
3. Camberra: $\mathcal{D}(x_i, x_j) = \sum_{m=1}^{n} \frac{|x_{im} - x_{jm}|}{|x_{im}| + |x_{jm}|}$

NOTE... ✍

Both Euclidean distance and the Manhattan distance can be regarded as special cases of Minkowski distance. The Euclidian distance results from the selection $q = 2$, the Manhattan distance for the parmeter value $q = 1$. The Canberra distance is a weighted version of the Manhattan distance.

An Example to Boost Your Understanding

Abalone are, mostly sedentary, marine snails and belong to a group of sea critters (phylum Mollusca) which include clams, scallops, sea slugs, octopuses and squid. They cling to rocks while waiting for their favorite food of kelp to drift by. Once the seaweed is spotted they hold it down, with what can only be described as a "foot", and chew on it vigorously (for a snail) with radula - a rough tongue with many small teeth.

Given the healthy diet of fresh kelp it is little wonder these mollusks are considered delicious raw with a little lemon or gently pan-fried tossed in garlic and butter. They are deemed to be so flavorful that the demand for consumption outstripped supply over many years. Today, the white abalone is officially listed as an endangered species. Other species remain plentiful, are harvested regularly to the delight of foodies[32], restaurant goers and seafood chefs[33].

Tasmania supplies around a quarter of the yearly world supply of abalone. The majority of which are either blacklip or greenlip abalone. Let's walk through kNN for classification of blacklip and greenlip abalone. Suppose we are given some standardized data points on age, weight and type of abalone, as illustrated in Figure 2.9. Our goal is to find the class label for the new point denoted with a question mark.

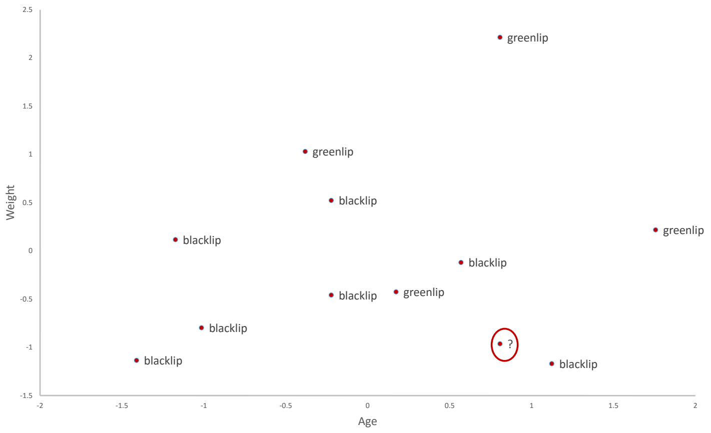


Figure 2.9: Abalone illustration

The sample observations are shown in Table 2 . A total of eleven labeled observations are available, and our task is to predict the class of the 12th observation, which has a standardized age of 0.807 and a standardized weight of -0.963.

Observation	Age	Weight	Class
1	-1.015	-0.797	blacklip
2	-0.223	-0.458	blacklip
3	0.569	-0.12	blacklip
4	-1.411	-1.135	blacklip
5	-0.223	0.523	blacklip
6	1.123	-1.169	blacklip
7	-1.174	0.117	blacklip
8	0.173	-0.425	greenlip
9	1.757	0.218	greenlip
10	0.807	2.215	greenlip
11	-0.382	1.031	greenlip
12	0.807	-0.963	?

Table 2: Attribute and class for each observation

To illustrate the calculation, I use Euclidean distance. So for example, the distance between new observation x_{12} and the 11th observation x_{11} is calculated as:

$$\mathcal{D}(x_{12}, x_{11}) = \sqrt{(x_{12\,1} - x_{11\,1})^2 + (x_{12\,2} - x_{11\,2})^2}$$

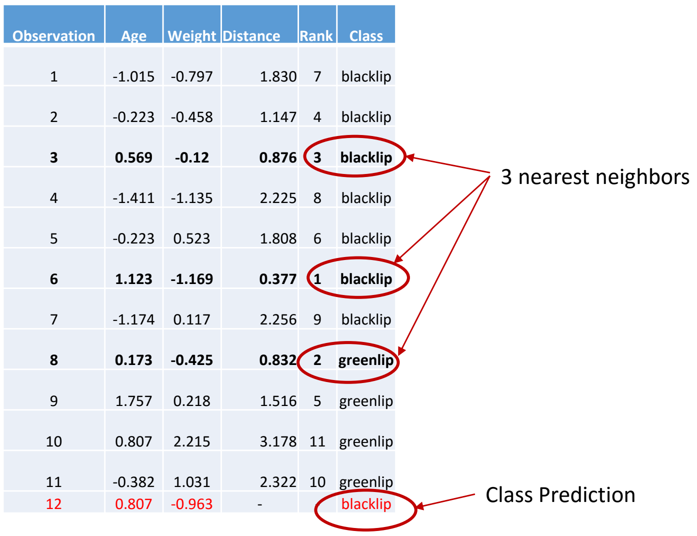
$$= \sqrt{(0.807 - (-0.382))^2 + (-0.963 - 1.031)^2}$$

$$= \sqrt{1.414 + 3.976}$$

$$= \sqrt{5.390}$$

$$= 2.322$$

Table 3 shows the distances for all eleven observations where the closest observation has a rank of 1. Setting k= 3, we see the nearest neighbor is observation 6 which is classified as blacklip, followed by observation 8 which is classified as greenlip, followed by observation 3 which is classified as blacklip. Using these three neighbors and majority voting, the new observation is classified as blacklip.



Observation	Age	Weight	Distance	Rank	Class
1	-1.015	-0.797	1.830	7	blacklip
2	-0.223	-0.458	1.147	4	blacklip
3	**0.569**	**-0.12**	**0.876**	**3**	**blacklip**
4	-1.411	-1.135	2.225	8	blacklip
5	-0.223	0.523	1.808	6	blacklip
6	**1.123**	**-1.169**	**0.377**	**1**	**blacklip**
7	-1.174	0.117	2.256	9	blacklip
8	**0.173**	**-0.425**	**0.832**	**2**	**greenlip**
9	1.757	0.218	1.516	5	greenlip
10	0.807	2.215	3.178	11	greenlip
11	-0.382	1.031	2.322	10	greenlip
12	0.807	-0.963	-		blacklip

3 nearest neighbors
Class Prediction

Table 3: Euclidean distances and rank by closeness

A Straightforward Way to Use k-Nearest Neighbors in R

We will build a kNN model using the `PimaIndiansDiabetes2` data frame contained in the `mlbench` package. This data set was collected by the National Institute of Diabetes and Digestive and Kidney Diseases[34]. It contains 768 observations on 9 variables measured on females at least 21 years old of Pima Indian heritage. Table 4 contains a description of each variable collected.

Name	Description
`pregnant`	Number of times pregnant
`glucose`	Plasma glucose concentration
`pressure`	Diastolic blood pressure (mm Hg)
`triceps`	Triceps skin fold thickness (mm)
`insulin`	2-Hour serum insulin (mu U/ml)
`mass`	Body mass index
`pedigree`	Diabetes pedigree function
`age`	Age (years)
`diabetes`	test for diabetes - Class variable (neg / pos)

Table 4: Response and independent variables in `PimaIndiansDiabetes2` data frame

The data can be loaded into your R session as follows:

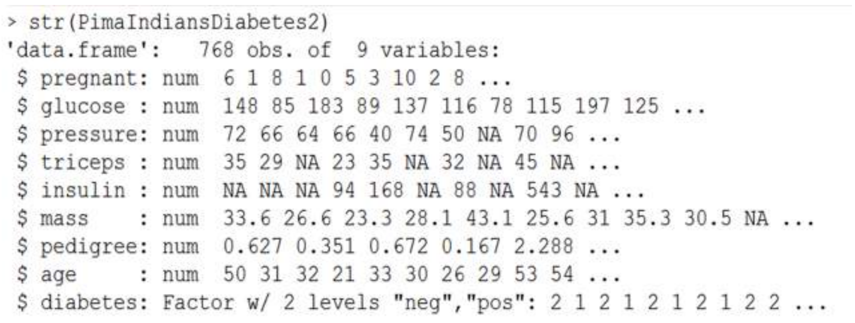
```
data("PimaIndiansDiabetes2",package="
   mlbench")
```

Let's do a quick check to ensure we have the expected number of columns and rows:

```
ncol(PimaIndiansDiabetes2)
[1] 9
nrow(PimaIndiansDiabetes2)
[1] 768
```

The numbers are in line we what we expected.

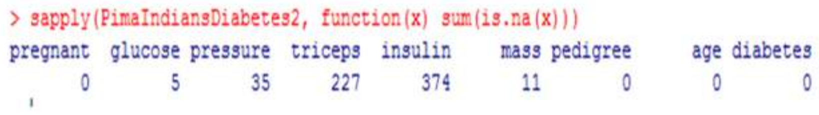
We can use the `str` method to check and compactly display the structure of the `PimaIndiansDiabetes2` data frame:



```
> str(PimaIndiansDiabetes2)
'data.frame':   768 obs. of  9 variables:
 $ pregnant: num  6 1 8 1 0 5 3 10 2 8 ...
 $ glucose : num  148 85 183 89 137 116 78 115 197 125 ...
 $ pressure: num  72 66 64 66 40 74 50 NA 70 96 ...
 $ triceps : num  35 29 NA 23 35 NA 32 NA 45 NA ...
 $ insulin : num  NA NA NA 94 168 NA 88 NA 543 NA ...
 $ mass    : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 NA ...
 $ pedigree: num  0.627 0.351 0.672 0.167 2.288 ...
 $ age     : num  50 31 32 21 33 30 26 29 53 54 ...
 $ diabetes: Factor w/ 2 levels "neg","pos": 2 1 2 1 2 1 2 1 2 2 ...
```

As expected `PimaIndiansDiabetes2` is identified as a data frame with 768 observations on 9 variables. Notice that each row provides details of the name of the attribute, type of attribute and the first few observations. For example, `diabetes` is a factor with two levels `"neg"` (negative) and `"pos"` (positive). We will use it as the classification response variable.

Did you notice the `NA` values in `pressure`, `triceps`, `insulin` and `mass`? These are missing values. We all face the problem of missing data at some point in our work. People refuse or forget to answer a question, data is lost or not recorded properly. It is a fact of data science life! However, there seem to be rather a lot, we better check to see the actual numbers:



```
> sapply(PimaIndiansDiabetes2, function(x) sum(is.na(x)))
pregnant  glucose pressure  triceps  insulin     mass pedigree      age diabetes
       0        5       35      227      374       11        0        0        0
```

Wow! there are a large number of missing values particularly for the attributes of `insulin` and `triceps`. How should we deal with this? The most common method and the easiest to apply is to use only those individuals for which we have complete information. An alternative is to impute with a plausible value the missing observations. For example, you might replace the `NA's` with the attribute mean or median. A more sophisticated approach would be to use a distributional model for the data (such as maximum likelihood and multiple imputation[35]).

Given the large number of missing values in `insulin` and `triceps` we remove these two attributes from the sample and use the `na.omit` method to remove any remaining missing values. The cleaned data is stored in `temp`:

```
temp<-(PimaIndiansDiabetes2)
temp$insulin  <- NULL
temp$triceps <- NULL
temp<-na.omit(temp)
```

We should have sufficient observations left to do meaningful analysis. It is always best to check:

```
nrow(temp)
[1] 724
ncol(temp)
[1] 7
```

So we are left with 724 individuals and (as expected) 7 columns.

Figure 2.10 shows the distribution by age, number of times pregnant alongside box-plots of glucose, blood pressure and bodymass index.

The training set consists of 600 randomly selected without replacement observations, leaving 124 examples for the test set:

```
set.seed(2016)
n=nrow(temp)
train <- sample(1:n, 600, FALSE)
```

The kNN function is in the `class` package:

```
library(class)
```

The attributes are stored in the scaled R object `data`. The class labels are not included in `data`, but removed using `[,-7]` because they are stored in the 7th column of `temp`. This will make our subsequent analysis a little easier:

```
data<-temp[,-7]
data<-scale(data)
```

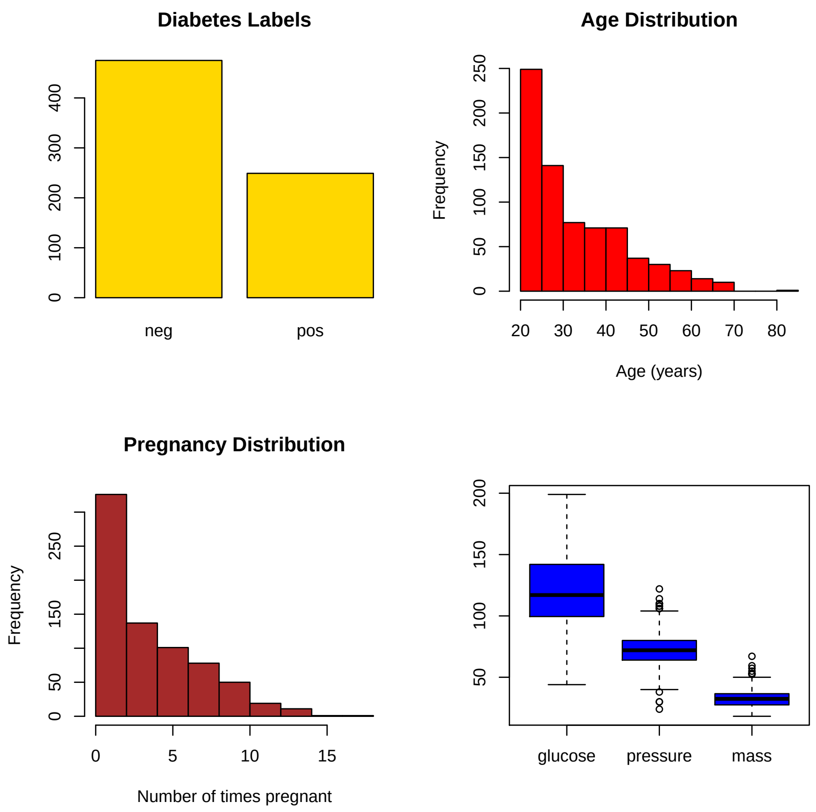


Figure 2.10: Class labels and attributes

Now to the meat of the issue, here is how to specify a kNN in the `class` package:

```
fit<-knn(data[train,],
data[-train,],
temp$diabetes[train] ,
k = 6,
prob=TRUE)
```

Much of this will be familiar to you by now, so I will run through it only briefly. The `knn` function fits nearest neighbors using Euclidean distance. The first argument contains the training sample, the second argument the test sample. You will notice that for this example `k = 6` to use the 6 nearest neighbors. The argument `prob = TRUE` returns the proportion of the votes for the winning class as an attribute of `fit`. To see the class labels and associated probabilities type:

```
fit
```

Let's take a look a the confusion matrix:

```
tab<-table(fit,temp$diabetes[-train])
tab


fit   neg pos
  neg  78  14
  pos  10  22
```

Notice the model did not fit the data perfectly. The off diagonal elements (14 and 10) in the confusion matrix indicate the number of misclassified observations. It is often helpful to calculate the accuracy rate. Here is how I do that:

```
sum(tab[row(tab)==col(tab)])/sum(tab)
[1] 0.8064516
```

The kNN has an overall accuracy rate of around 80%.

Classifier performance is often assessed using a number of metrics derived from the confusion matrix. The overall accuracy (ACC), true negative rate (TNR) and true positive rate (TPR):

$$ACC = \frac{TP + TN}{TP + FP + TN + FN}.$$

The true negative rate or specificity:

$$TNR = \frac{TN}{FP + TN}.$$

The true positive rate or sensitivity:

$$TPR = \frac{TP}{TP + FN};$$

where TP is the true positives, TN is false positives, FN false negatives and TN true negatives.

Here is How to Determine the Optimal Value of k

The parameter k is a positive integer, but how do you choose a suitable value? A small value of k means that noise will have a higher influence on the classification. A large value of k reduces the overall noise but makes the boundaries between classes less distinct. Very large values of k defeat the basic idea behind kNN that nearby points have similar classes. This is because, as k gets very large, a global majority vote of the whole sample results in a constant prediction for all new observations that have to be classified - the most frequent class within the sample is always the predicted class of a new observation.

Choosing the optimal value for k is best done by first inspecting the data. Two simple rules of thumb can help. The first is to set $k = \sqrt{n}$. The second is to choose a value between 3 and 10; this range appears to works well for a wide variety of datasets.

NOTE... ✍

Back in 1967 scholars P.E Hart and T.M Cover proved that as the amount of data approaches infinity, and for any number of classes, the probability of error of the nearest neighbor rule is bounded above by twice the Bayes error rate[36]. The kNN algorithm is therefore guaranteed to yield an error rate no worse than twice the Bayes error rate as the sample size increases.

The Key to Linear Discriminant Analysis

Linear discriminant analysis searches for a linear combination of attributes that best separates two or more classes. It uses, say d, linear discriminant functions which take the general form:

$$g(x) = \beta x = \beta_0 + \sum_{i=1}^{d} \beta_i x_i, \tag{2.3}$$

where w is the weight vector and w_0 is the threshold weight or bias. Taking a two class problem for illustration, the objective is to learn a linear discriminant function which best separates the classes. The resulting combination is then used as a linear classifier using the rule:

$$if\ g(x) > 0\ then\ x \in c_1\ else\ x \in c_2.$$

Back in 1936 British statistician, Sir Ronald Fisher explored the idea of linear discriminant functions for the two class problem[37] - "*When two or more populations have been measured in several characters, $x_l, ..., x_8$, special interest attaches to certain linear functions of the measurements by which the populations are best discriminated.*" Discussing, what has become known as the Iris flower data set Sir Fisher poised the question: "*What linear function of the four measurements $X = \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 + \lambda_4 x_4$ will maximize the ratio of the difference between the specific means to the standard deviations within* [Iris] *species?*"

Sir Fisher noted that classes could be well separated using a linear function like $g(x)$ provided the mean values of each class were sufficiently different from each other. This resulted in the idea of maximizing the ratio of the difference between class means, normalized by a measure of the within-class scatter:

$$J(w) = tr\left[\left(wS_w w^T\right)^{-1}\left(wS_B w^T\right)\right],$$

$J(w)$ is known as the Fisher criterion, where S_B is the between classes scatter matrix and S_W is the pooled within classes scatter matrix. For K classes:

$$S_B = \sum_{i=1}^{K} P(c_i)\left(\bar{x}_i - \bar{x}\right)\left(\bar{x}_i - \bar{x}\right)^T,$$

$$S_W = \sum_{i=1}^{K} P(c_i) S_i,$$

where S_i is the within class covariance matrix of class i, $\bar{x}_i$ is the mean vector for class i, and $\bar{x}$ is the total mean vector given by:

$$\bar{x} = \sum_{i=1}^{K} P(c_i)\bar{x}_i$$

NOTE... ✍

American botanist Edgar Anderson collected the Iris flower data set in Canada's beautiful Gaspésie[38]. It consists of 50 samples from each of three species of Iris: Iris setosa, Iris virginica and Iris versicolor. Four attributes were measured from each sample: the length and the width of the sepals and petals $\{x_1, x_2, x_3, x_4\}$in centimeters. Using a combination of these four attributes, Sir Fisher created a linear discriminant model to better distinguish the species from each other.

For the multiclass problem Fisher's criterion does not approach the Bayes criterion[39]. It therefore is not minimizing misclassification error, instead it maximizes the mean squared distance between classes in the lower dimensional space. However, despite this it has nevertheless delivered solid performance in a wide range of real world classification challenges such as financial analysis of corporations[40], face recognition[41], image classification[42], and medical screening[43].

Solving the Generalized Eigenvalue Problem

It turns out that the optimal projection matrix w can be obtained by solving the generalized eigenvalue problem provided S_W is non-singular:

$$S_B w = \lambda S_W w,$$

and w is obtained by the eigenvectors of the matrix:

$$S_W^{-1} S_B.$$

If the eigen values $\lambda_1 > \lambda_2 > ... > \lambda_n$ then λ_i is the ratio of the between group sum of squares to the within group sum of squares for the i^{th} linear combination, $g_i(x)$; the coefficients of the corresponding eigenvector are the coefficients of $g_i(x)$.

The function $g_i(x)$ is often called a canonical discriminant function, and it turns out given a data sample with n attributes and K classes that it may be possible to determine several, say v, such linear combinations for separating classes. The discriminant functions $g_1(x), g_2(x), ..., g_v(x)$ are linear combinations of the original features chosen in such a way that $g_1(x)$ captures the largest proportion of class differences; $g_2(x)$ captures as much as possible the class differences not captured by $g_1(x)$; $g_3(x)$ captures as much as possible the class differences not captured by $g_1(x)$ and $g_2(x)$, and so on.

NOTE... ✍

Linear discriminant analysis assumes independent continuous attributes from a normal distribution with all classes sharing the same covariance matrix. When dealing with categorical independent attributes, the equivalent method is known as discriminant correspondence analysis.

In general, the number of discriminant functions is the smaller of n and $K-1$. The hope is that the first few functions, say $g_1(x)$ and $g_2(x)$, are sufficient to account for the majority of class differences. To classify a new point a distance metric is specified $\mathcal{D}(.)$ and the new point x_{new} is classified to:

$$\arg \min_k \mathcal{D}(x_{new}w, \bar{x}_k w),$$

where $\bar{x_k}$ is the centriod of the k^{th} class.

Essential Elements for Discriminant Analysis in R

Let's take another look at the thyroid data originally discussed on page 36. First load the packages, create the train and test sets:

```
data("thyroid",package="mclust")
data<-thyroid
set.seed(2016)
N=nrow(thyroid)
train<-sample(1:N,75,FALSE)
train_set<-data$Diagnosis[-train]
```

The above is familiar to you by now, so I won't discuss at length. Just note that in this case we select a training sample of 75 observations.

It is always a good idea to visually inspect your data. Figure 2.11 illustrates this using a 2d scatterplot. The distribution of each attribute is visualized using a histogram along the diagonal; the upper panel shows the correlation coefficient between attributes; and the lower panel the pairwise scatterplots. It is interesting to observe that the scatterplots indicate some separation between classes.

It is also useful to check if the class conditional marginals look Gaussian. Figure 2.12 presents the class conditional density plots. Let's see how well linear discriminant analysis does here.

NOTE... ✍

Generative models usually perform poorly when the assumptions are violated. For example, if $P(x|y)$ is very non-Gaussian, then linear discriminant analysis will not work well. Logistic Regression is more robust to violations of model assumptions, and neural networks are even more robust[44].

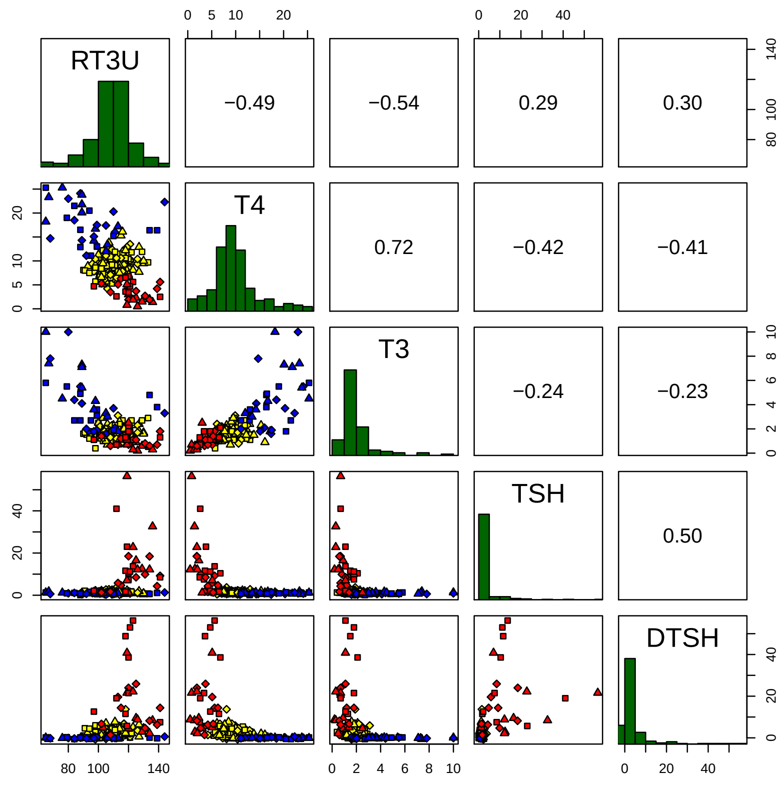


Figure 2.11: Pairwise scatter plot of `thyroid` attributes.

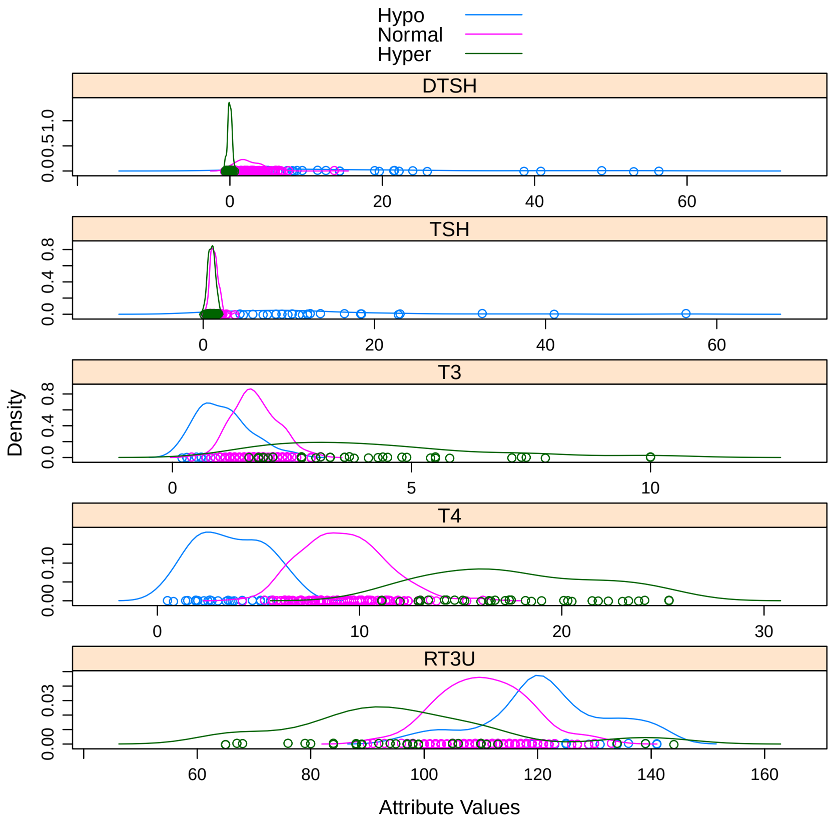


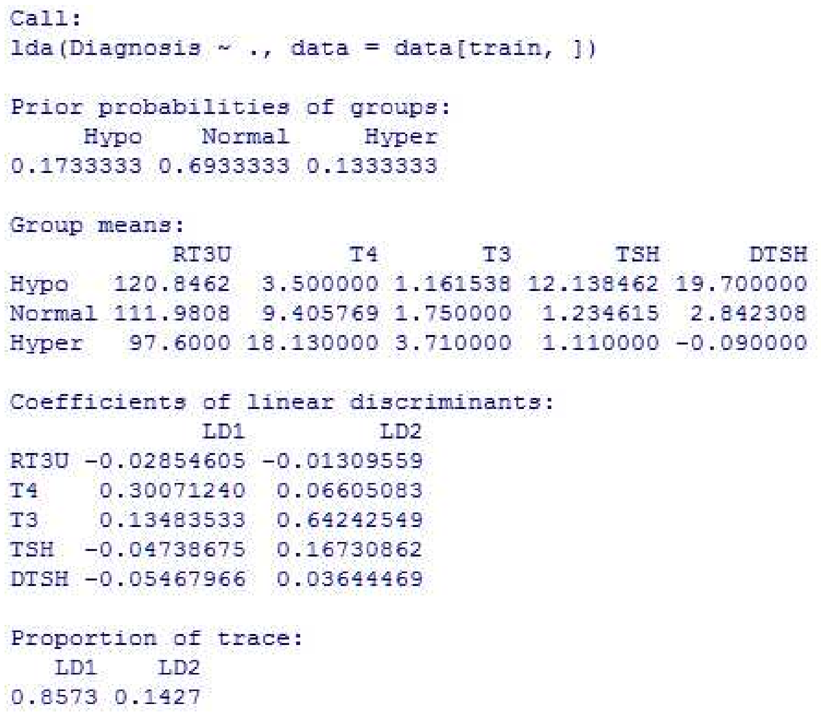
Figure 2.12: Class conditional marginals of `thyroid` attributes

I will use the `lda` function in the package `MASS` to fit the model:

```
require(MASS)
fit<-lda(Diagnosis ~.,data=data[train,])
```

Take a look at the attributes of `fit`:

```
print(fit)
```



```
Call:
lda(Diagnosis ~ ., data = data[train, ])

Prior probabilities of groups:
     Hypo    Normal     Hyper
0.1733333 0.6933333 0.1333333

Group means:
           RT3U        T4       T3       TSH      DTSH
Hypo   120.8462  3.500000 1.161538 12.138462 19.700000
Normal 111.9808  9.405769 1.750000  1.234615  2.842308
Hyper   97.6000 18.130000 3.710000  1.110000 -0.090000

Coefficients of linear discriminants:
             LD1         LD2
RT3U -0.02854605 -0.01309559
T4    0.30071240  0.06605083
T3    0.13483533  0.64242549
TSH  -0.04738675  0.16730862
DTSH -0.05467966  0.03644469

Proportion of trace:
   LD1    LD2
0.8573 0.1427
```

The between class variance is labeled "`Proportion of trace`"; it measures the variance that is explained by successive discriminant functions. In this case 85.7% of the variance is explained by the first discriminant function `LD1` and the remaining 14.7% explained by the second discriminant function `LD2`.

Now we use the model to predict the test set examples, calculate the confusion matrix and the classification error rate:

```
pred<-predict(fit,data[-train,])$class
tab<-table(pred,train_set)

tab
        train_set
pred      Hypo Normal Hyper
  Hypo      13      0     0
  Normal     4     98     8
  Hyper      0      0    17
```

```
mean(pred != train_set)
[1] 0.08571429
```

NOTE... ✍

The `lda` method estimates class priors directly from the data. However, you can specify them yourself using the `prior` argument. For example, to set equal priors for all three classes you would type:

```
fit<-lda(Diagnosis ~.,
data=data[train,],
prior=c(1,1,1)/3)
```

Check the Kind of Model You Want

When I first started to work with data, I built the perfect model. It had a classification error rate of exactly zero. I literally danced for joy! Unfortunately, the amazing prediction tool turned out to be a dud in practice. My biggest mistake was to use the entire sample to estimate the model parameters. Yes, my classification error was zero, but I had used the resubstitution error - the classification error obtained by using all of the available sample data. Most text books will tell you that it is an overly optimistic estimate; and in most cases that is true (but not always).

Let's calculate it for the linear discriminant analysis model we just built:

```
fitR<-lda(Diagnosis ~.,data=data)

predR<-predict(fitR,data)$class
```

```
tabR<-table(predR,data$Diagnosis)

tabR

predR     Hypo Normal Hyper
  Hypo      22      0     0
  Normal     8    150     9
  Hyper      0      0    26

mean(predR != data$Diagnosis)
[1] 0.07906977
```

At 7.9% it is a touch lower than the number we observed in the train and test set previously. I outline why it is not a good idea to use the resubstitution error on its own as a reflection of the true classification error rate in chapter 6. For now, just remember that it is better to partition your data into (at minimum) a training and test set. The test error is a better (often unbiased) estimate of the underlying expected classification error, although we can often do much better with the techniques outlined in chapter 6.

Don't Stop at Linear Discriminant Analysis

Model building is an iterative process that involves choosing an algorithm, evaluating that model and deploying it for action. One of the reasons I enjoying model building in R is the flexibility and shared commonality. You can build a quadratic discriminant function use `qda` instead of `lda` with everything else the same. Take a look at how easy it is:

```
fitq<-qda(Diagnosis ~.,data=data[train,])
predq<-predict(fitq,data[-train,])$class

tabq<-table(predq,train_set)
tabq
        train_set
```

```
predq    Hypo Normal Hyper
  Hypo     15      0     0
  Normal    2     97     6
  Hyper     0      1    19

mean(predq != train_set)
[1] 0.06428571
```

The overall classification error rate is a little over 6%, and slightly better than the linear model.

NOTE... ✍

Quadratic discriminant analysis learns quadratic boundaries and is therefore more flexible than linear discriminant analysis. Unlike linear discriminant analysis, the covariance matrix are not assumed to be identical. When the covariance matrices are diagonal, the classes are conditionally independent; in this case quadratic discriminant analysis is equivalent to the naive Bayes classifier[45].

The Secret of Classification with Logistic Regression

Today logistic regression is one of the most widely taught and used binary models in the analysis of categorical data. It is a staple in statistics 101 classes, dominates quantitative marketing, and is used with numerous mutations in classical econometrics[46]. In numerous areas it has become the benchmark which challenger models must meet and defeat. It is the grandson of linear regression, a member of the generalized linear family of models and has reigned supreme in the medical statistics community for several decades.

The logistic regression model, for classification, is usually formulated mathematically by relating the probability a object belongs to c_i conditional on a vector of features x, through the functional form of a parametric logistic conditional density function:

$$E\left[c_i|x\right] = \frac{1}{\left[1+\exp\left[-\alpha-\beta^T x\right]\right]},$$

where (α, β) are unknown parameters to be estimated, typically using the method of maximum likelihood, from the data. Logistic regression is primarily used in binary classification, so c_i, $i = 1, 2$. Multi-class logistic regression can be used for outcomes with more than two classes / categories, in this case $i = 1, 2, ...K$.

NOTE... ✍

It appears the logistic model was first introduced sometime during the first half of the 19th century by statistician Alphonse Quetlet's student Pierre-François Verhulst. It was discovered, in part, as an exercise to moderate Thomas Malthus's exponential population growth model. Close to 200 years after Verhulst's discovery, the scientific community, commercial sectors and government researchers have fully embraced it as a powerful tool for binary and multiclass classification challenges.

A direct link to the Bayes classifier can be observed if we specify the logistic regression formula as a probability distribution of the class posterior probabilities. For a two class classification problem we have:

$$P(c_i = 1|x) = \frac{1}{\left[1+\exp\left[-\alpha-\sum_{i=1}^{n}\beta_i x_i\right]\right]}$$

$$P(c_i = 0|x) = \frac{\exp\left[-\alpha - \sum_{i=1}^{n} \beta_i x_i\right]}{\left[1 + \exp\left[\alpha + \sum_{i=1}^{n} \beta_i x_i\right]\right]}$$

In this case we predict $c_i = 1$ if:

$$P(c_i = 1|x) > P(c_i = 0|x),$$

which is essentially the decision criteria used for the Bayes classifier.

In theory discriminative classifiers have lower asymptotic error than generative classifiers. Vapnik's principle indicates a preference for discriminative classifiers [47]: "*If you possess a restricted amount of information for solving some problem, try to solve the problem directly and never solve a more general problem as an intermediate step. It is possible that the available information is sufficient for a direct solution but is insufficient for solving a more general intermediate problem.*"

However, in practice, generative classifiers such as naive Bayes often outperform discriminative classifiers such as logistic regression. It turns out the generative naive Bayes model reaches its asymptotic bound at a rate $O(logN)$, whilst the discriminative logistic model approaches it bound at a rate of $O(N)$. The practical implication of this is that the naive Bayes model reaches its asymptotic solution for a much smaller data sample than the logistic model. In other words, if you have a small sample for which you want to build a logistic regression model for classification, also try the naive Bayes model.

Since logistic regression is founded on modeling the odds of an outcome it is perfectly suited to the classification task, and has rightly earned its place as the gold standard in many disciplines.

NOTE... ✍

Well over a decade ago scholars Andrew Ng and Michael Jordan[48] studied the error properties of logistic regression and naive Bayes models. They found that naive Bayes reaches its asymptotic error bound much faster than the discriminative logistic regression classifier. However, as the sample size grows larger, and larger, logistic regression outperforms the naive Bayes classifier. The scholars illustrated the result for 15 real world datasets from the UCI machine learning repository[49].

The Easy Way to Build a Logistic Regression Classifier in R

The purple rock crab (leptograpsus variegatus) inhabits the shoreline of the North Island, Kermadec Islands, Australia and the islands of the South Pacific. It is an athlete, being both fast and strong with a ferocious nip (try picking one up!). This spiny critter comes in two distinct colors - gorgeous blue and radiant orange.

The `crabs` data frame in the `MASS` package contains observations on 200 purple rock crabs. For each crab 8 features were measured - color, sex and 5 morphological measurements shown in Table 5.

Let's build a logistic regression classifier for crab gender using R right now. Load the required package:

```
library("MASS")
```

Figure 2.13 shows the scatter plots, distribution and correlation between the attributes. What stands out is the very high correlation between the attributes. This is not surprising

Name	Description
sp	colour "B" or "O"
sex	Class (Male/Female)
FL	frontal lobe size (mm)
RW	rear width (mm)
CL	carapace length (mm)
CW	carapace width (mm)
pedigree	Diabetes pedigree function
BD	body depth (mm)

Table 5: Class and independent variables in `crabs` data frame

as they are all body size measurements. What to do? One solution is to select a subset of the features. However, the domain experts (zoologists) considered all the features relevant. So rather than discard attributes instead let's use the principal components of the data:

```
pca <- prcomp(crabs[,4:8],
              center = TRUE,
              scale. = TRUE)
```

Notice the attributes are stored in columns 4 to 8 of `crabs`.

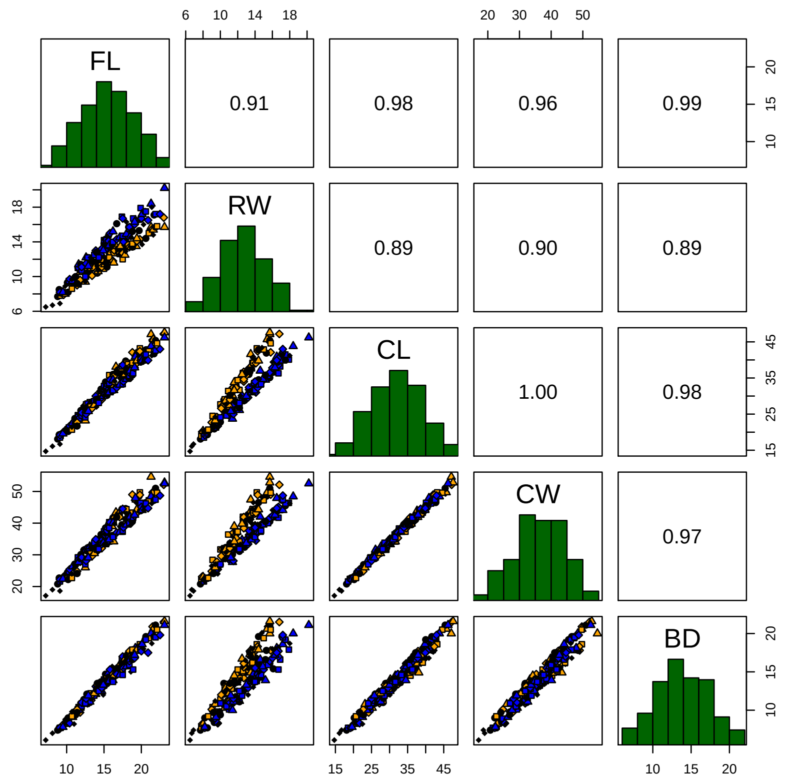


Figure 2.13: Pairwise scatter plot of `crabs` attributes.

The proportion of variation explained by each principal component can be calculated using:

```
prop.pca = pca$sdev^2/sum(pca$sdev^2)

round(prop.pca, digits=3)
[1] 0.958 0.030 0.009 0.002 0.000
```

The first component explains 95.8% of the variation in the feature data and the second component 3%. Since, these two components account for over 98% of the variation in the data, we will use these as our independent variables in the logistic regression model:

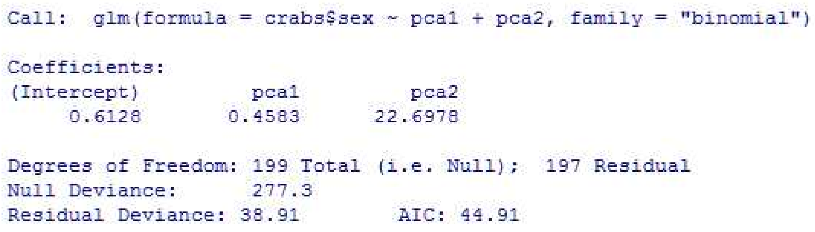
```
pca1<-pca$x[,1]
pca2<-pca$x[,2]
```

For this example we will calculate the resubstitution error. To do this we will need to use all of the data to build the model. It can be estimated in R as follows:

```
fit <- glm(crabs$sex~pca1+pca2, family="
    binomial")
```

The numerical results for the overall fitted model are extracted like this:

```
fit
```



```
Call:  glm(formula = crabs$sex ~ pca1 + pca2, family = "binomial")

Coefficients:
(Intercept)         pca1         pca2
     0.6128       0.4583      22.6978

Degrees of Freedom: 199 Total (i.e. Null);  197 Residual
Null Deviance:      277.3
Residual Deviance: 38.91        AIC: 44.91
```

NOTE... ✍

The following requests also produce useful results: `fit$coef`, `fit$fitted`, `fit$resid`, `fit$effects`, and `anova(fit)`.

Let's put on our Statistician hat for a moment. A simple way to evaluate the overall performance of the model is to look at the null deviance and residual deviance. Null deviance indicates how well the class is predicted by a model with nothing but the intercept. We would expect such a model to be a poor classifier. In this example the null deviance indicates very little fit (a highly significant difference between fitted values and observed values). In statistical language it is a chi square value on 199 degrees of freedom. Adding in our predictors decreased the deviance by 238 points on 2 degrees of freedom. Again, this is interpreted as a chi square value and indicates a highly significant decrease in deviance. The residual deviance is 38.9 on 197 degrees of freedom. We can use this value to assess the overall fit of the model by once again treating this as a chi square value. To do this, we need to figure out the p value of a chi square of 38.9 on 197 degrees of freedom. Here is how to do this:

```
1-pchisq(38.9,197)
[1] 1
```

It yields a p-value of 1 so the null hypothesis (i.e., the fitted model) is not rejected. This indicates that the fitted values are not statistically significantly different from the observed values - the model fits the data well.

Let's calculate the confusion matrix and error rate using the predicted values:

```
pred <- predict(fit, type="response")
predclass <- factor(ifelse(pred>0.5, "M", "
   F"))
table(predclass, crabs$sex)

predclass  F  M
        F 95  4
        M  5 96

mean(predclass != crabs$sex)
[1] 0.045
```

The second line converts the logistic regression probability estimates into class labels, notice I use a cut off of 0.5. The confusion matrix indicate the model fits the data reasonably well with a resubstitution error of 4.5%. Figure 2.14 plots the logistic regression decision boundary and confirms the overall good fit.

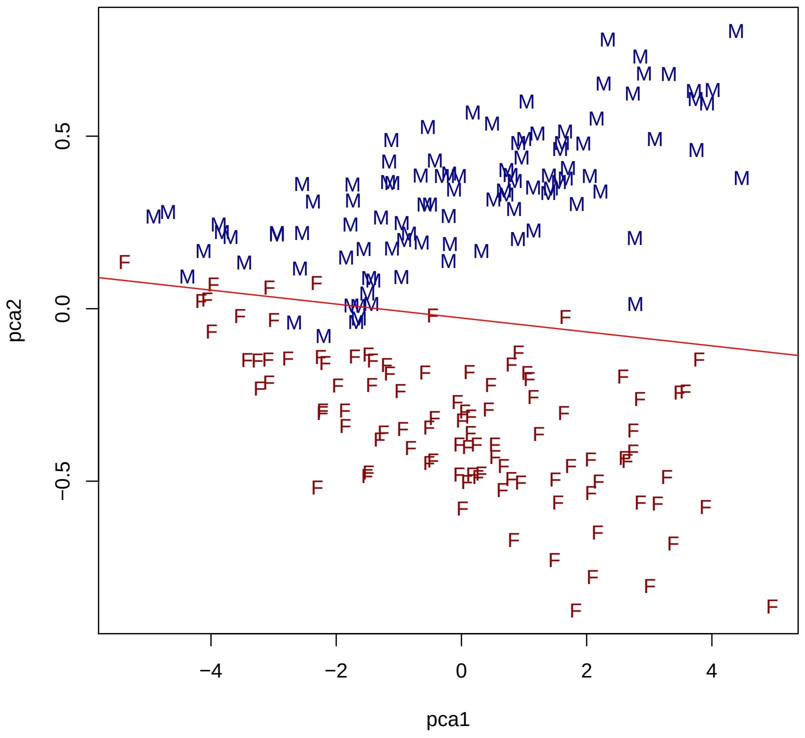


Figure 2.14: Logistic regression decision boundary for `crabs`.

NOTE... ✍

Principal component analysis (PCA), multiple dimensional scaling (MDS) and locally linear embedding (LLE)[50] are frequently used for dimensionality reduction. In PCA linear projections of ordered by variance are computed from eigenvectors of the data covariance matrix. In MDS, a low dimensional embedding that best preserves pairwise distances between data points is the focus of calculation. LLE computes low dimensional, neighborhood preserving embeddings of high dimensional data.

A Damn Good Idea to Inspire Your Creativity and Passion

Not too long ago I was socializing with a friend when the topic of skin cancer came up. Turns out this individual has had a number of lesions removed over the years; and had just returned from hospital having had another lesion zapped; this time on top of the left nostril. All is well for my friend, but as John F. Gilmore III of the Pittsburgh Post-Gazette reports[51] "*The deadliest form of skin cancer, melanoma is one of the fastest growing cancers, not only in the U.S. but also the world. Ninety percent of the cases are linked to exposure to UV rays from the sun or tanning beds.*" Data science has a role to play in the fight against this disease.

University of Guelph graduate Dr. Paul Wighton[52] along with colleagues, use supervised learning to develop automated skin lesion diagnosis[53]. This is an important activity because automated skin lesion diagnosis holds the promise of increased efficiency by rapidly screening lesions; rejecting benign lesions and referring suspicious ones to a medical specialist for further evaluation. I think we can all agree that this is indeed a very worthwhile goal.

Figure 2.15 shows the typical process used for automated skin lesion diagnosis. As Dr. Wighton explains "*First an image is acquired with a digital dermoscope. Next, undesirable artifacts (such as hair or oil bubbles) are identified and, if necessary, replaced with an estimate of the underlying skin color. After this, the lesion is segmented, and discriminative features are then extracted. Finally, supervised learning is used to classify previously unseen images.*"

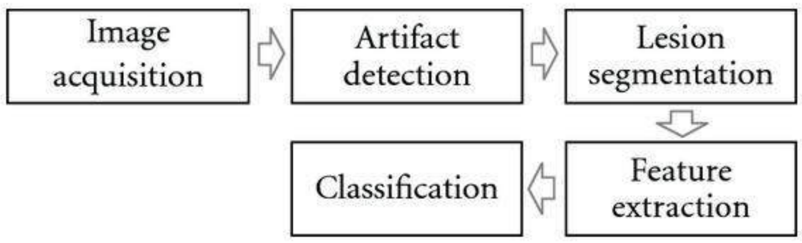


Figure 2.15: Typical process adopted for automated skin lesion diagnosis. Imaged source Wighton et al cited in Note item 53

It is instructive to look at how Dr. Wighton set up the problem for successful analysis. Recall from page 9, the three elements to the learning problem:

1. **Denote the response variable** y and appropriate classes. The researchers characterized the response or target as y_i, which contains class labels defined by $L = l_1, ..., l_{N_L}$, where N_L is the number of possible labels or classes of skin lesion.

2. **Assume a probabilistic relationship** exists between x and y captured by an unknown probability distribution $P(x, y) = P(x)P(y|x)$. The researchers use Vapnik's

principle and select a supervised discriminative classification model. In this case the *Conditional Random Fields model*[54] to estimate $P(y|x)$ directly.

3. **Observe a sample** set of observations $\{x^m, y^m\}$ consisting of the images (x) and labels associated with the target response y.

The researchers investigate their model using a data-set consisting of 116 images. Figure 2.16 illustrates visually the results for three different images. Typically, such images have "holes" where the algorithm performs poorly. What is remarkable about all three of these images is the absence of such distortions. This is quite remarkable. The researchers observe "*...the CRF model offers substantial improvements...*"

Dr. Paul Wighton and associates have found a particularly valuable use for supervised learning; what about you? How will you deploy this amazing technology in your own projects? Get a copy of their wonderful paper, it will ignite your own creativity.

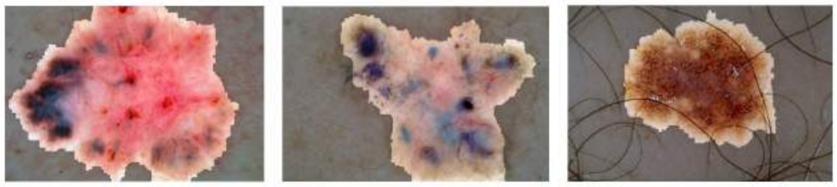


Figure 2.16: Three images produced by using the supervised learning model of Wighton et al. Source Wighton et al cited in Note item 53

Notes

[10]Pereira, Clayton R., et al. "An Optimum-Path Forest framework for intrusion detection in computer networks." Engineering Applications of Artificial Intelligence 25.6 (2012): 1226-1234.

[11]Nakamura, Rodrigo Yuji Mizobe, et al. "Nature-inspired framework for hyperspectral band selection." Geoscience and Remote Sensing, IEEE Transactions on 52.4 (2014): 2126-2137.

[12]Ramos, Caio César Oba, et al. "A new approach for nontechnical losses detection based on optimum-path forest." Power Systems, IEEE Transactions on 26.1 (2011): 181-189.

[13]See:

- Castillo, C., I. Chollett, and E. Klein. "Enhanced duckweed detection using bootstrapped SVM classification on medium resolution RGB MODIS imagery." International Journal of Remote Sensing 29.19 (2008): 5595-5604.
- Pereira, Luis AM, et al. "Aquatic weed automatic classification using machine learning techniques." Computers and electronics in agriculture 87 (2012): 56-63.

[14]According to the wine institute, Australia was the world's fifth largest producer of wine in 2014. See http://www.wineinstitute.org/

[15]Texas is America's 5th largest wine producer, with many local vineyards serving a delicious and stunning variety of wines. See http://www.txwines.org

[16]See for example:

- Foody, G. M., and M. K. Arora. "An evaluation of some factors affecting the accuracy of classification by an artificial neural network." International Journal of Remote Sensing 18.4 (1997): 799-810.
- Zhang, Lefei, et al. "On combining multiple features for hyperspectral remote sensing image classification." Geoscience and Remote Sensing, IEEE Transactions on 50.3 (2012): 879-893.
- Olofsson, Pontus, et al. "Good practices for estimating area and assessing accuracy of land change." Remote Sensing of Environment 148 (2014): 42-57.

[17]See for example:

- Pal, Mahesh, and Paul M. Mather. "An assessment of the effectiveness of decision tree methods for land cover classification." Remote sensing of environment 86.4 (2003): 554-565.

- Foody, G. M., M. B. McCulloch, and W. B. Yates. "The effect of training set size and composition on artificial neural network classification." International Journal of Remote Sensing 16.9 (1995): 1707-1723.
- Zhuang, X., et al. "Optimization of training data required for neuro-classification." Remote Sensing 15.16 (1994): 3271-3277.

[18]For a discussion and application see for example:

- Coppi, Dalia, Simone Calderara, and Rita Cucchiara. "Active query process for digital video surveillance forensic applications." Signal, Image and Video Processing 9.4 (2015): 749-759.
- Wang, Kebin, and Ying Tan. "A new collaborative filtering recommendation approach based on naive Bayesian method." Advances in Swarm Intelligence. Springer Berlin Heidelberg, 2011. 218-227.
- Ewerth, Ralph, and Bernd Freisleben. "Adapting appearance models of semantic concepts to particular videos via transductive learning." Proceedings of the international workshop on Workshop on multimedia information retrieval. ACM, 2007.

[19]If you are interested in the technical aspects of this here are three excellent books to add to your library:

1. P.A. Devijver and J. Kittler. Pattern Recognition: A Statistical Approach. Prentice-Hall, 1982.
2. R. O. Duda, P. E. Hart, and D. G. Stork. Pattern Classification. Wiley, New York, NY, second edition, 2001.
3. K. Fukunaga. The estimation of the Bayes error by the k-nearest neighbor approach. In L. N. Kanal and A. Rosenfeld, editors, Progress in Pattern Recognition 2, pages 169–187. North- Holland, Amsterdam, 1985.

[20]If you know of a quicker or easier method let me know. You can contact me at info@NigelDLewis.com.

[21]See this amazing little book by Devijver and Kittler for additional details:

- P.A. Devijver and J. Kittler. Pattern Recognition: A Statistical Approach. Prentice-Hall, 1982.

Also see these research papers which give additional details:

- Bruzzone, Lorenzo, and Sebastiano B. Serpico. "A simple upper bound to the Bayes error probability for feature selection." Kybernetika 34.4 (1998): 387-392.

- Tumer, Kagan, and Joydeep Ghosh. "Bayes error rate estimation using classifier ensembles." International Journal of Smart Engineering System Design 5.2 (2003): 95-109.

[22]Assumed to be non-singular.

[23]See McCallum, Andrew, and Kamal Nigam. "A comparison of event models for naive Bayes text classification." AAAI-98 workshop on learning for text categorization. Vol. 752. 1998.

[24]See Kharde, Vishal, and Sheetal Sonawane. "Sentiment Analysis of Twitter Data: A Survey of Techniques." arXiv preprint arXiv:1601.06971 (2016).

[25]See Chadha, Ritika, et al. "Application of Data Mining Techniques on Heart Disease Prediction: A Survey." Emerging Research in Computing, Information, Communication and Applications. Springer India, 2016. 413-426.

[26]See Kalisch, Mateusz. "Fault Detection Method Using Context-Based Approach." Advanced and Intelligent Computations in Diagnosis and Control. Springer International Publishing, 2016. 383-395.

[27]See Domingos, Pedro, and Michael Pazzani. "On the optimality of the simple Bayesian classifier under zero-one loss." Machine learning 29.2-3 (1997): 103-130.

[28]Further details at ftp://ftp.ics.uci.edu/pub/machine-learning-databases/thyroid-disease/new-thyroid.names

[29]If the attributes are categorical it gives the conditional probabilities given the target class.

[30]For a discussion of why this is so see:

- Bennett, Paul N. Assessing the calibration of Naive Bayes posterior estimates. No. CMU-CS-00-155. Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 2000.
- Zhang, Harry. "The optimality of naive Bayes." AA 1.2 (2004): 3.
- Rish, Irina. "An empirical study of the naive Bayes classifier." IJCAI 2001 workshop on empirical methods in artificial intelligence. Vol. 3. No. 22. IBM New York, 2001.

[31]See Fix, Evelyn, and Joseph L. Hodges Jr. Discriminatory analysis-nonparametric discrimination: consistency properties. California Univ Berkeley, 1951.

[32]For the foodies among you here are a few East Coast (USA) restaurants which serve abalone steaks:

- Allegretto Vineyard Resort Paso Robles 805-369-2500 http://www.ayreshotels.com/allegretto-resort-and-vineyard-paso-robles

- Artisan Paso Robles 805-237-8084 http://artisanpasorobles.com
- Windows on the Water Morro Bay 805-772-0677 http://www.windowsmb.com
- Madeline's Cambria 805-927-4175 http://www.madelinescambria.com
- The Black Cat Cambria 805-927-1600 http://www.blackcatbistro.com
- Linn's Restaurant Cambria 805-927-0371 http://www.linnsfruitbin.com/Linns_Restaurant.html
- Madonna Inn San Luis Obispo 805-543-30

[33]See for example Chef Rick Moonen who states "*We believe in the importance of buying and serving seafood that comes from abundant populations which are under sound management. All fish on our menu are caught or farmed in a way that is not harmful to the ocean environment or to other ocean creatures. We are strong supporters of local fishing communities and take responsibility for our role in preserving a lasting and diverse supply of seafood.*" For more information on great sustainable seafood visit http://rickmoonen.com/

[34]See http://www.niddk.nih.gov/

[35]For alternative approaches to dealing with missing values see:

1. Roth, Philip L. "Missing data: A conceptual review for applied psychologists." Personnel psychology 47.3 (1994): 537-560.
2. Afifi, A. A., and R. M. Elashoff. "Missing observations in multivariate statistics I. Review of the literature." Journal of the American Statistical Association 61.315 (1966): 595-604.
3. Pigott, Therese D. "A review of methods for missing data." Educational research and evaluation 7.4 (2001): 353-383.
4. Little, Roderick J., et al. "The prevention and treatment of missing data in clinical trials." New England Journal of Medicine 367.14 (2012): 1355-1360.

[36]See Cover, Thomas M., and Peter E. Hart. "Nearest neighbor pattern classification." Information Theory, IEEE Transactions on 13.1 (1967): 21-27.

[37]Get a copy of Fisher's paper. It is great reading and relevant even today. See Fisher, Ronald A. "The use of multiple measurements in taxonomic problems."Annals of eugenics 7.2 (1936): 179-188.

[38]Anderson, Edgar. "The species problem in Iris." Annals of the Missouri Botanical Garden 23.3 (1936): 457-509.

[39]See Loog, Marco, R. P. W. Duin, and Reinhold Haeb-Umbach. "Multiclass linear dimension reduction by weighted pairwise Fisher criteria." IEEE Transactions on Pattern Analysis & Machine Intelligence 7 (2001): 762-766.

[40]Altman, Edward I., Giancarlo Marco, and Franco Varetto. "Corporate distress diagnosis: Comparisons using linear discriminant analysis and neural networks (the Italian experience)." Journal of banking & finance 18.3 (1994): 505-529.

[41]Liu, Chengjun, and Harry Wechsler. "Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition." Image processing, IEEE Transactions on 11.4 (2002): 467-476.

[42]Bandos, Tatyana V., Lorenzo Bruzzone, and Gustavo Camps-Valls. "Classification of hyperspectral images with regularized linear discriminant analysis." Geoscience and Remote Sensing, IEEE Transactions on 47.3 (2009): 862-873.

[43]Chan, Heang-Ping, et al. "Computer-aided classification of mammographic masses and normal tissue: linear discriminant analysis in texture feature space." Physics in medicine and biology 40.5 (1995): 857.

[44]For details of how to build neural networks quickly and easily in R pick up a copy of my book “Build Your Own Neural Network Today” at www.AusCov.Com

[45]Gaussian Naive Bayes classifier. Remember the classes in quadratic discriminant analysis are assumed to be Gaussian.

[46]See for example Maddala’s book. Add it to your personal library. It is a classic and worth every penny:

- Maddala, Gangadharrao S. Limited-dependent and qualitative variables in econometrics. No. 3. Cambridge university press, 1986.

[47]Vapnik, Vladimir. The nature of statistical learning theory. Springer Science & Business Media, 2013.

[48]See the paper by Andrew Ng and Michael Jordan who developed the theory and ran 15 experiments to show this to be the case in practice. The paper is entitled "On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes." Advances in neural information processing systems 14 (2002): 841.

[49]You can visit it at http://archive.ics.uci.edu/ml/

[50]Saul, Lawrence K., and Sam T. Roweis. "Think globally, fit locally: unsupervised learning of low dimensional manifolds." The Journal of Machine Learning Research 4 (2003): 119-155.

[51]See the Pittsburgh Post-Gazette. Local foundation aims to spread melanoma awareness, support research. January 19, 2016.

[52]Paul is a rapidly rising star in his field. He obtained his PhD in Computing Science from Simon Fraser University. At the time of writing

he was a Research Fellow at the Harvard Medical School. You can contact him directly at pwighton@nmr.mgh.harvard.edu.

[53]Paul Wighton, Tim K. Lee, Greg Mori, Harvey Lui, David I. McLean, and M. Stella Atkins, "Conditional Random Fields and Supervised Learning in Automated Skin Lesion Diagnosis," International Journal of Biomedical Imaging, vol. 2011, Article ID 846312, 10 pages, 2011. doi:10.1155/2011/846312

[54]The Conditional Random Fields (CRF) model is an undirected graphical model. If you are familiar with logistic regression, then the multinomial logistic regression model can be seen as the simplest kind of CRF in which there is only one output variable. The CRF model combines the ability to compactly model multivariate outputs y with a large number of input attributes x. This feature makes it useful for representing the dependencies between observations, such as neighboring pixels in an image. For a good overview see C. Sutton and A. McCallum, "An introduction to conditional random fields for relational learning," in Introduction to Statistical Relational Learning, p. 93, 2007.

Chapter 3

Unsupervised Learning

I am still learning.

Michelangelo

FROM time to time you will encounter situations where your only option is unsupervised learning; and great results are certainly possible with this approach. In this chapter we will explore the core idea behind unsupervised learning, get practical hands on experience using powerful techniques to learn from data, and examine an award winning example that you can mirror to create excellent results in your own area of interest.

Unsupervised Learning in a Nutshell

Recall in supervised learning you have both labels and attributes. For example, in a medical setting you might have labels on the health status of patients ("Excellent","Good", "Poor") alongside attributes such as age, weight, blood pressure and so on. In unsupervised learning, there are no labeled samples from which to draw inferences. Instead, you have a pile of observations on the attributes. What are we to do here?

You can try to define clusters of patients, i.e. groups of patients showing similarities, for example patients < 250 pounds and patients >250 pounds. The key is to find useful representations of the information embedded in the data. Unsupervised learning exploits characteristics in data, such as variability and separability, to determine attribute/feature relevance. The goal is to divide a set of unlabeled data into a distinct set of classes, or clusters. Fortunately, Statisticians have developed a number of powerful algorithms specifically designed for this situation. They are typical know as cluster analysis algorithms.

The Two Core Approaches and How They Work

Cluster analysis is an exploratory data analysis tool which aims to group a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups. The groups of similar objects are called clusters. Cluster analysis itself is not one specific algorithm but consists of a wide variety of approaches. Two of the most popular approaches are hierarchical clustering and partition based clustering.

Hierarchical clustering methods attempt to maximize the distance between clusters. They begin by assigning each object to its own cluster. At each step the algorithm merges together the least distant pair of clusters, until only one cluster remains. At every step the distance between clusters, say cluster A and cluster B is updated.

Partitioning clustering methods attempt to minimize the "average" distance within clusters. K-means and fuzzy k means are two of the most popular methods. They typically proceed as follows:

1. Select at random group centers and assign objects to the nearest group.

2. Form new centers at the center of these groups.

3. Move individual objects to a new group if it is closer to that group than the center of its present group.

4. Repeat step 2 and 3 until no (or minimal) change in groupings occurs.

NOTE... ✍

Unsupervised classification using clustering techniques have been very successfully in a wide variety of areas including, intrusion detection in computer networks[55], land-use classification in satellite imagery[56], medical image analysis[57], just to name a few.

Techniques that Crush Unsupervised Learning & How to Build them in R

Here are four of popular clustering tools and how to build them quickly in R.

Technique 1: K-means

The most popular partition based clustering algorithm is the k-means method. It requires a distance metric and the number of centroids (clusters) to be specified. Here is how it works:

1. Determine how many clusters you expect in your sample, say for example k.

2. Pick a group of k random centroids. A centroid is the center of a cluster.

3. Assign sample observations to the closest centroid. The closeness is determined by a distance metric. They become part of that cluster.

4. Calculate the center (mean) of each cluster.

5. Check assignments for all the sample observations. If another center is closer to an observation, reassign it to that cluster.

6. Repeat step 3-5 until no reassignment occur.

Much of the popularity of k-means lies in the fact that the algorithm is extremely fast and can therefore be easily applied to large data-sets. However, it is important to note that the solution is a local maximum, so in practice several starting points should be used.

For our initial analysis we will use the `kmeansruns` function in the `fpc` package. This uses the `kmeans` method in the `stats` package. We also use the `Vehicle` data frame contained in the `mlbench` package for our analysis. The `Vehicle` data set[58] was collected to classify a "Corgie" vehicle silhouette as one of four types (double decker bus, Chevrolet van, Saab 9000 and Opel Manta 400). The data frame contains 846 observations on 18 numerical features extracted from the silhouettes and one nominal variable defining the class of the objects (see Table 6).

Let's load the required packages and data:

```
require(fpc)
data("Vehicle",package="mlbench")
data<-Vehicle[,-19]
set.seed(2016)
```

Notice that the third line stores the Vehicle data minus the class labels in the R attribute `data`.

Since `opel` and `saab` are both types of car, I reclassify them into one class called `car`. So we now have three classes, `car`, `bus` and `van`:

Data-frame Index	R name	Description
1	`Comp`	Compactness
2	`Circ`	Circularity
3	`D.Circ`	Distance Circularity
4	`Rad.Ra`	Radius ratio
5	`Pr.Axis.Ra`	pr.axis aspect ratio
6	`Max.L.Ra`	max.length aspect ratio
7	`Scat.Ra`	scatter ratio
8	`Elong`	elongatedness
9	`Pr.Axis.Rect`	pr.axis rectangularity
10	`Max.L.Rect`	max.length rectangularity
11	`Sc.Var.Maxis`	scaled variance along major axis
12	`Sc.Var.maxis`	scaled variance along minor axis
13	`Ra.Gyr`	scaled radius of gyration
14	`Skew.Maxis`	skewness about major axis
15	`Skew.maxis`	skewness about minor axis
16	`Kurt.maxis`	kurtosis about minor axis
17	`Kurt.Maxis`	kurtosis about major axis
18	`Holl.Ra`	hollows ratio
19	`Class`	bus, opel, saab, van

Table 6: Attributes and Class labels for `Vehicle` data set

```
class<-Vehicle[,19]
levels(class)[levels(class)=="saab"] <- "
   car"
levels(class)[levels(class)=="opel"] <- "
   car"
```

Figure 3.1 shows the reclassified distribution of vehicles, with `car` being the most frequent vehicle type, and a roughly equal split between `van` and `bus`.

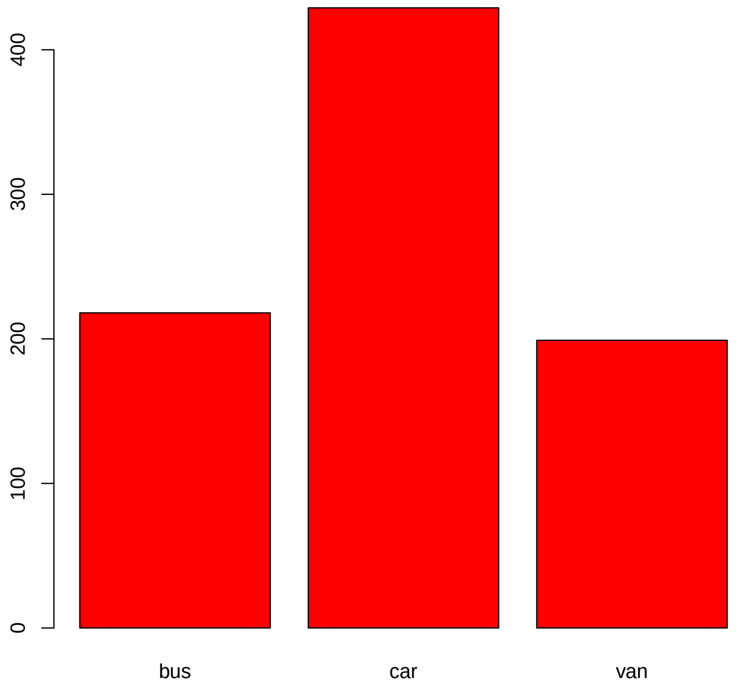


Figure 3.1: Types of Vehicle

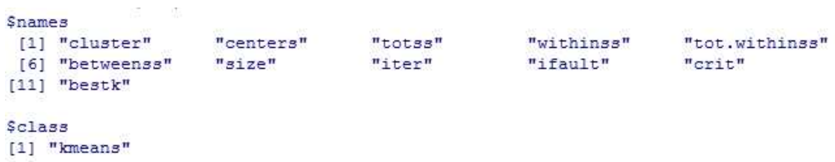
K- means requires you specify the exact number of clusters. However, in practice you will rarely know this number precisely. One solution is to specify a range of possible clusters and use a metric such as the Calinski-Harabasz index or average silhouette width to choose the appropriate number.

Let us suppose we expect the number of clusters to be between 3 and 8. In this case we could call the `kmeansruns` method setting `krange` to range between 3 and 8, and the `criterion` parameter = `"ch"` for the Calinski-Harabasz index. The results of k-means clustering may vary from run to run, due to random selection of initial cluster centers. Therefore, I set `runs = 100` to use 100 different starting clusters; and the observations are scaled using `scaledata=TRUE`:

```
fit <- kmeansruns(data,krange=1:8,
critout=TRUE,runs=100,scaledata=TRUE,
criterion="ch")
2  clusters  588.9425
3  clusters  456.7939
4  clusters  433.9752
5  clusters  381.9492
6  clusters  352.7663
7  clusters  331.3771
8  clusters  309.9925
```

The output is the Calinski-Harabasz index where the optimal number of clusters is selected using the maximum value. The maximum value of 588.9 occurs for two clusters. The R object `fit` contains all of the elements of the optimum model:

```
attributes(fit)
```



```
$names
 [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
 [6] "betweenss"    "size"         "iter"         "ifault"       "crit"
[11] "bestk"

$class
[1] "kmeans"
```

Figure 3.2 shows the bivariate cluster plot, where we see some mixing at the margins of the two clusters. To investigate this a little further Figure 3.3 presents a silhouette plot.

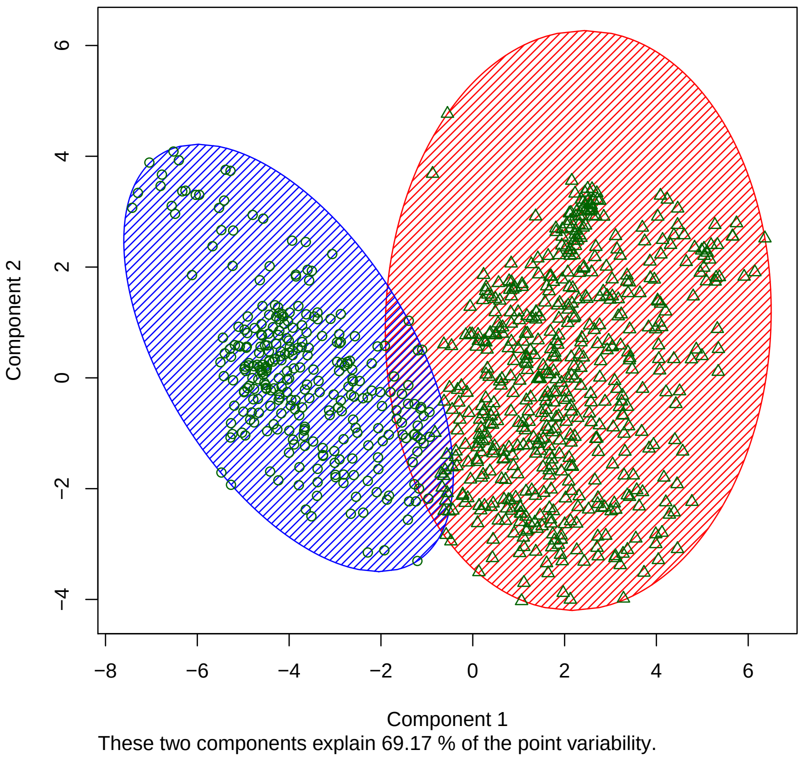


Figure 3.2: Bivariate Cluster Plot

A silhouette plot measures how close each point in one cluster are to points in the neighboring clusters. It has a range of -1 to +1. A value close to +1 indicates an observation is far apart from the neighboring clusters, a value of zero indicates the observation is on the decision boundary; and negative values suggest the observation might have been assigned to the wrong cluster.

Next, the clustering result is compared with the class labels to see whether similar vehicles are grouped together:

```
tab<-table( fit$cluster,class)
```

```
tab
   class
    bus car van
  1  57 234   4
  2 161 195 195
```

The above result shows that cluster 1 to a large extent represents `car`, whilst for cluster 2 there appears to be a large degree of overlap across all vehicle types.

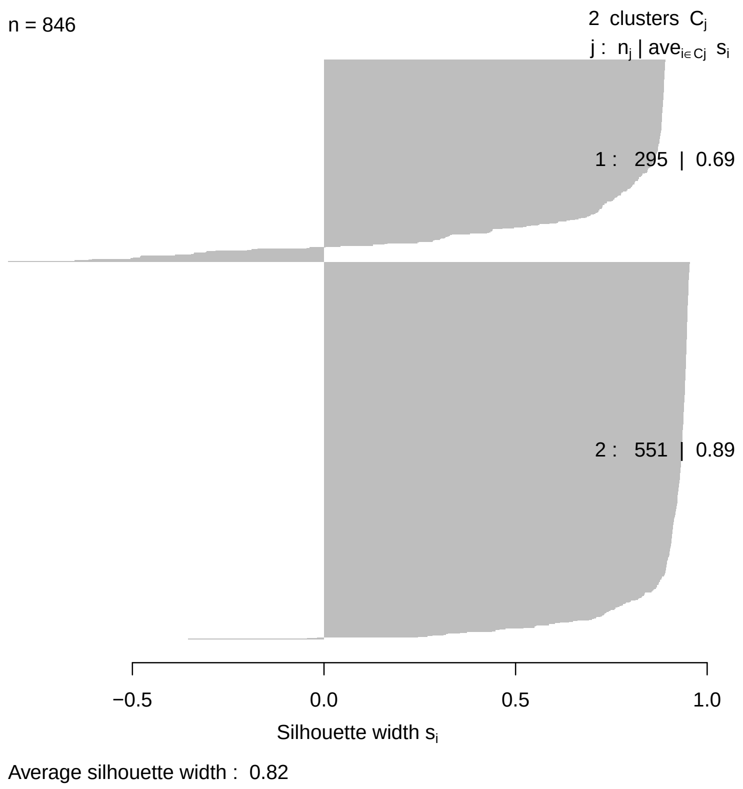


Figure 3.3: Silhouette plot

Technique 2: Fuzzy K-means

The K-means algorithm implicitly assume that there are distinct classes. Much of the time the classes will be vague, uncertain - in a word fuzzy. Fuzzy classification associates a probability that each example belongs to a specific class. The fuzzy-K means algorithms work in a similar way to K-means but produce a probability distribution of class membership.

Let's build a fuzzy K-means classifier continuing with the K-means example on page 81. We will use the `fclust` package and fit a fuzzy K-means model with two clusters (`k=2`). Here is how to do that:

```
require (fclust)
fit<-FKM(data, k=2, m=2, RS=10, stand=1)
```

The function `FKM` calls the standard fuzzy clustering algorithm. The parameter m (1<`m`< ∞) defines the degree of fuzzification. The larger it is, the more fuzzy are the membership values of the clustered data points are. It is common practice[59] to use an initial value of `m = 2`. A total of 10 restarts are specified on the standardized data.

NOTE... ✍

The `fclust` package has a number of fuzzy clustering algorithms available:
`FKM`: standard fuzzy-K means algorithm.
`FKM.gk`: Gustafson and Kessel extension of fuzzy-K means.
`FKM.med`: fuzzy k-medoids algorithm.
`FKM.noise`: fuzzy-K means with cluster noise.

You can view the cluster and the probability of class membership as follows:

```
head(fit$clus)
  Cluster Membership degree
```

```
1         2           0.5654644
2         2           0.9242917
3         1           0.9120457
4         2           0.7712232
5         2           0.5048897
6         1           0.738422
```

The first observation is assigned to the cluster 2 with a membership degree probability of 56.5%, the sixth observation is assigned to cluster 1 with a membership degree probability of 73.8%.

As we saw on page 86 the clustering result can be compared with the class labels to see whether similar vehicles are grouped together:

```
tab<-table( fit$clus[,1],class)
tab
   class
    bus car van
  1  60 243   4
  2 158 186 195
```

The results are similar to those observed with K-means. You can reproduce the K-means results exactly by setting `m = 1.1`:

```
fita<-FKM(data, k=2, m=1.1, RS=10, stand=1)
tab<-table( fita$clus[,1],class)
tab
   class
    bus car van
  1  57 234   4
  2 161 195 195
```

As with K- means you can use a cluster validity index to help determine the value of `k`. This is achieved through the `Fclust.index` function. To get the value of the fuzzy silhouette index for our example you would type:

```
Fclust.index(fit,index="SIL.F")
```

```
The default value alpha=1 has been set for
   computing SIL.F
    SIL.F
0.6353147
```

NOTE... ✍

`Fclust.index` has a large number of potential indices. The most relevant functions for fuzzy cluster analysis are:
`PC`: partition coefficient.
`PE`: partition entropy.
`XB`: Xie and Beni index.
`SIL.F`: fuzzy silhouette.

Technique 3: Hierarchical Cluster Analysis

To illustrate this approach, we will continue with the `data` object derived from the `Vehicle` dataframe. We will use the `pvclust` package:

```
data<-scale(data)
library(pvclust)
set.seed(2016)
```

Notice, in this case I scale the data using `scale`, oftentimes you will want to do this to ensure comparability.

The hierarchical clustering model can be fitted as follows:

```
fit <- pvclust(data,
method.hclust="ward.D",
nboot=5000,
method.dist="euclidean")
```

Let's spend a moment to get to grips with the function call. The `pvclust` function performs hierarchical cluster analysis via the standard R function `hclust`. However, it uses bootstrapped resampling to better estimate the clusters. I use ward's method as the clustering algorithm, set the number of bootstrapped re-samples to 5000 and calculate the distance matrix using euclidean distance.

NOTE... ✍

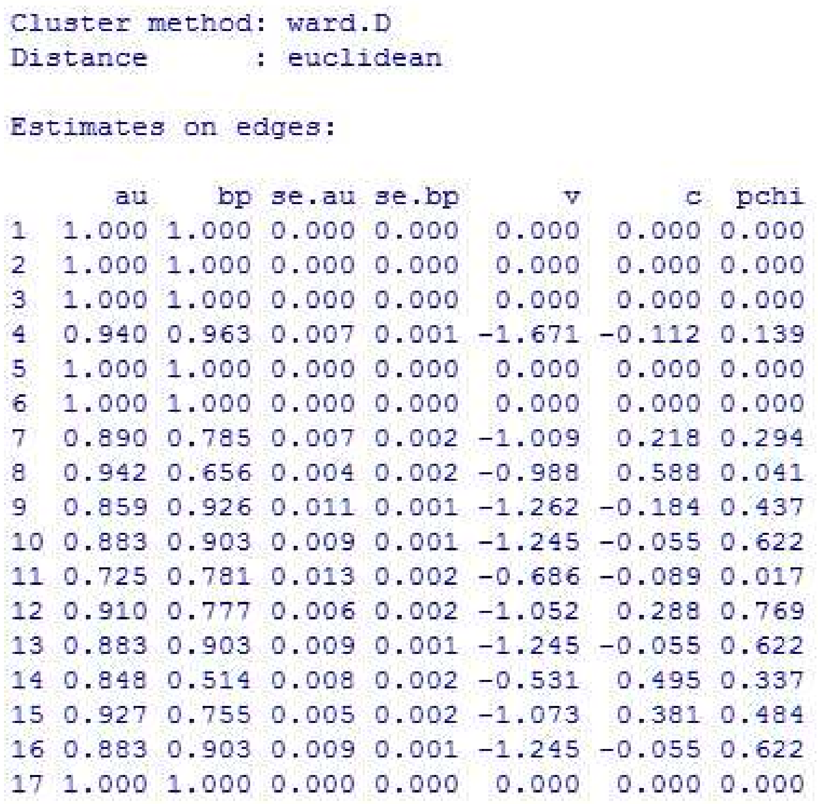
I have had good practical success using ward's method. However, it is interesting to notice that in the academic literature there are two different algorithms associated with the method! The method `ward.D2` squares the dissimilarities before clustering whilst `ward.D"` does not[60]. Other available methods include:

- `average`,
- `single`,
- `complete`,
- `mcquitty`, `median`,
- `centroid`.

For each cluster, quantities called p-values are calculated. The p-value of a cluster takes a value between 0 and 1. Clusters that are highly supported by the data will have p values close to 1.

Two types of p-values are calculated - the approximately unbiased (`au`) p-value and the bootstrap probability (`bp`) p-value. You can see the p-values using:

```
print(fit)
```



```
Cluster method: ward.D
Distance      : euclidean

Estimates on edges:

      au    bp se.au se.bp      v      c  pchi
1  1.000 1.000 0.000 0.000  0.000  0.000 0.000
2  1.000 1.000 0.000 0.000  0.000  0.000 0.000
3  1.000 1.000 0.000 0.000  0.000  0.000 0.000
4  0.940 0.963 0.007 0.001 -1.671 -0.112 0.139
5  1.000 1.000 0.000 0.000  0.000  0.000 0.000
6  1.000 1.000 0.000 0.000  0.000  0.000 0.000
7  0.890 0.785 0.007 0.002 -1.009  0.218 0.294
8  0.942 0.656 0.004 0.002 -0.988  0.588 0.041
9  0.859 0.926 0.011 0.001 -1.262 -0.184 0.437
10 0.883 0.903 0.009 0.001 -1.245 -0.055 0.622
11 0.725 0.781 0.013 0.002 -0.686 -0.089 0.017
12 0.910 0.777 0.006 0.002 -1.052  0.288 0.769
13 0.883 0.903 0.009 0.001 -1.245 -0.055 0.622
14 0.848 0.514 0.008 0.002 -0.531  0.495 0.337
15 0.927 0.755 0.005 0.002 -1.073  0.381 0.484
16 0.883 0.903 0.009 0.001 -1.245 -0.055 0.622
17 1.000 1.000 0.000 0.000  0.000  0.000 0.000
```

For the first edge both `au` and `bp` equal 1. Since the p-values are themselves estimates, they are measured with some error. The `se.au` and `se.bp` columns report the standard errors of the `au` and `bp` p-values respectively. Figure 3.4 shows the approximate 95% level confidence intervals for each edge using the `au` p-values.

The fitted model, with rectangles around groups highly supported by the data, can be visualized using the `plot` function:

```
plot(fit)
pvrect(fit, alpha=0.95)
```

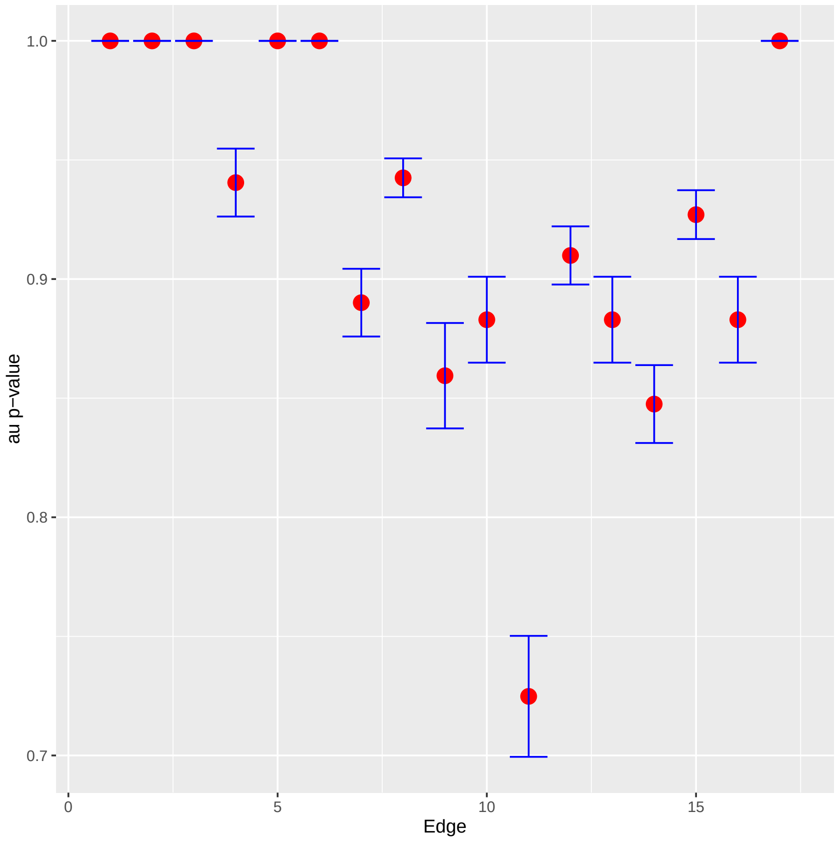


Figure 3.4: Confidence Intervals

For a cluster with `au` p-value > 0.95, the hypothesis that the cluster is the result of random noise is rejected with significance level 0.05; three such clusters can be observed in Figure 3.5.

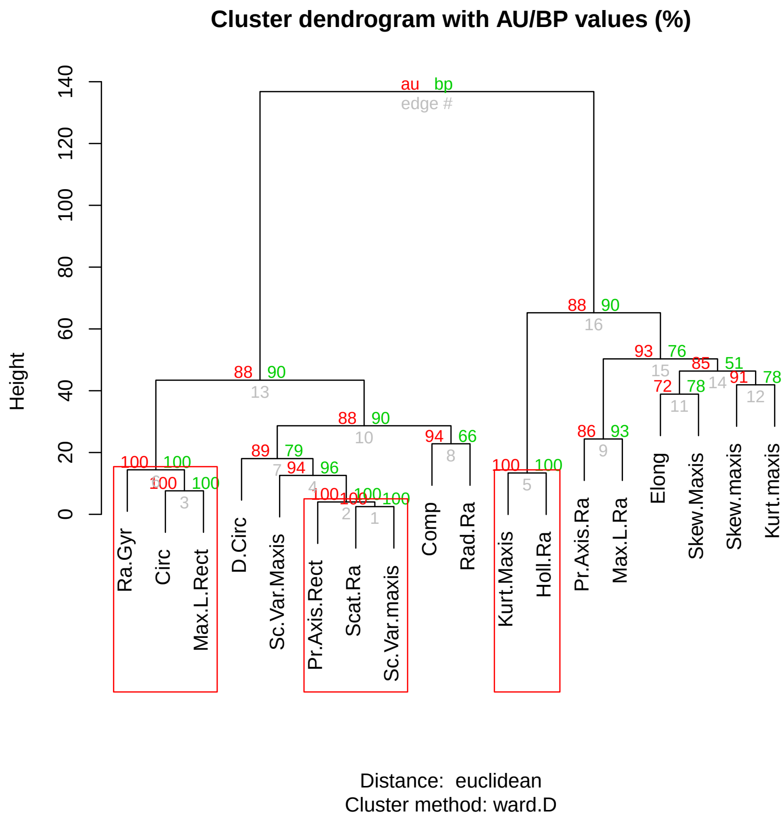


Figure 3.5: Hierarchical Cluster Plot

Technique 4: Model Based Clustering

In model based clustering the data are assumed to be clustered according to a mixture of Gaussian distributions. Each cluster generates Gaussian data distributed around its center. The approach assumes there are G clusters, where each cluster g occurs with probability π_g, and the data within a cluster are Gaussian with a cluster mean μ_g and cluster specific covariance Σ_g. Given the multivariate normal density $\phi()$, the density of each observation x_i is a mixture of Gaussians given by:

$$f(x_i) = \sum_{g=1}^{G} \pi_g \phi(x_i | \mu_g, \Sigma_g),$$

We can use the `Mclust` package for model based clustering in R. It computes the optimum Bayesian information criterion for each of the following Gaussian mixture models.

1. `"EII"` = spherical, equal volume
2. `"VII"` = spherical, unequal volume
3. `"EEI"` = diagonal, equal volume and shape
4. `"VEI"` = diagonal, varying volume, equal shape
5. `"EVI"` = diagonal, equal volume, varying shape
6. `"VVI"` = diagonal, varying volume and shape
7. `"EEE"` = ellipsoidal, equal volume, shape, and orientation
8. `"EVE"` = ellipsoidal, equal volume and orientation
9. `"VEE"` = ellipsoidal, equal shape and orientation
10. `"VVE"` = ellipsoidal, equal orientation
11. `"EEV"` = ellipsoidal, equal volume and equal shape
12. `"VEV"` = ellipsoidal, equal shape
13. `"EVV"` = ellipsoidal, equal volume
14. `"VVV"` = ellipsoidal, varying volume, shape, and orientation.

The models differ primarily due to various restrictions placed on the covariance matrix Σ_g. It turns out that the assumption of multivariate normally distributed clusters implies the clusters are elliptical in shape. The best way to understand the models is to note that each letter in the name of a model corresponds to the constraint placed on the volume, shape and orientation respectively. The constraint can be equal (`E`), variable (`V`) or identity (`I`). For example, in the `EEV` model each cluster has the same volume and the same shape but the orientation of each cluster is allowed to differ. In the EII model, each cluster has the same volume, shape and orientation, see Figure 3.6.

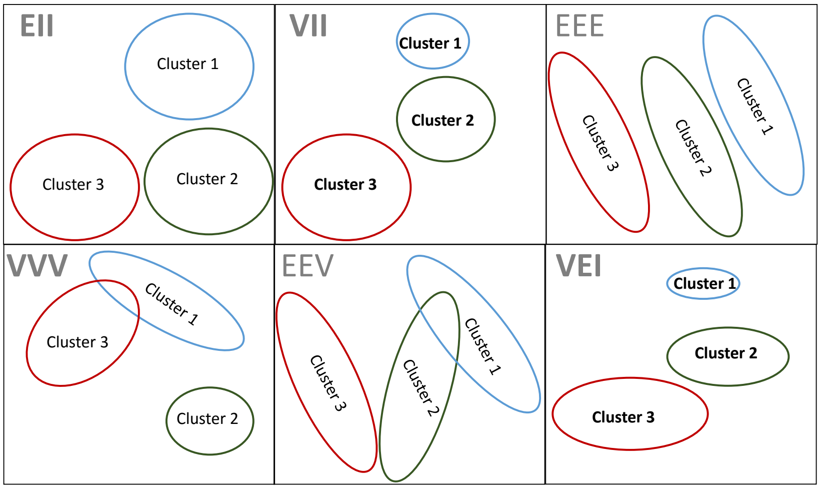


Figure 3.6: Sample of cluster shapes for various models

We will continue with the analysis of the `Vehicle` dataset:

```
data("Vehicle",package="mlbench")
data<-Vehicle[,-19]
library(mclust)
set.seed(2016)
fit<-Mclust(as.matrix(data),G=1:8)
```

The `Mclust` function selects the optimum model using the Bayesian Information Criterion (BIC). The function takes the `data` object as a matrix; and I use `G = 1:8` to instruct the algorithm to return the optimal model using 1 to 8 Gaussian components. To see the optimal model type:

```
fit
'Mclust' model object:
 best model: ellipsoidal, varying volume,
    shape, and orientation (VVV) with 6
    components
```

The optimal model is `VVV` with 6 mixing components.

NOTE... ✍

The Bayesian Information Criterion is calculated as:

$$BIC = 2\log(L) - p\log(N),$$

where L is the likelihood of the data, and p the number of model parameters.
The optimal model has the largest BIC value.

Figure 3.7 plots the Bayesian information criterion (BIC) value by Gaussian model type and number of components. The optimal model is the ellipsoidal, varying volume, shape, and orientation (VVV) with 6 components. Notice that although the BIC is maximized at 6 components, it is actually relatively flat after the second or third component. Therefore, I select a model with 3 components:

```
 fit3<-Mclust(as.matrix(data),G=3,
    modelNames ="VVV")
```

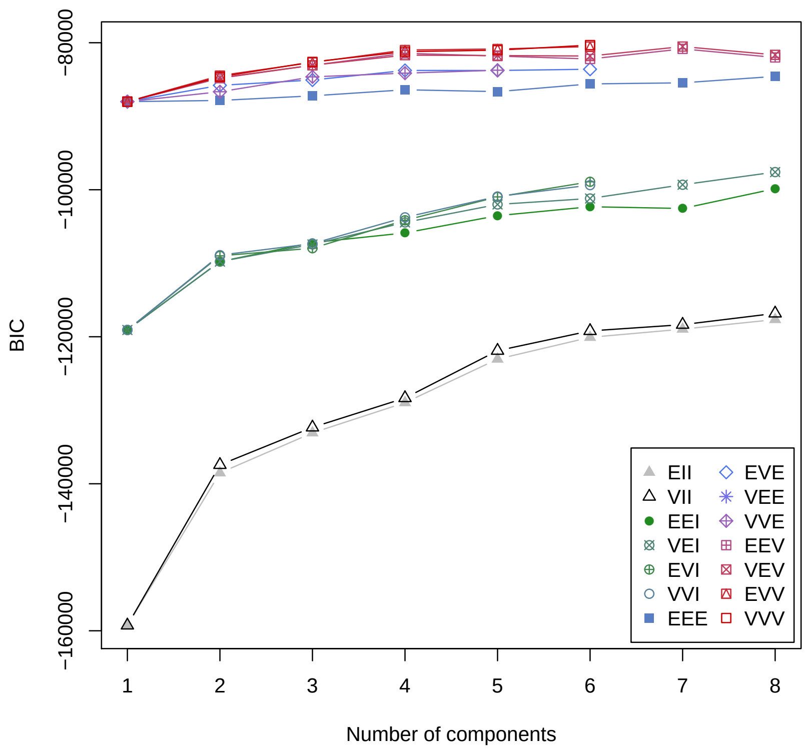


Figure 3.7: Plot of model based clustering results by model type and number of components

Next the clustering result is compared with the class labels to see whether similar vehicles are grouped together:

```
class<-Vehicle[,19]
levels(class)[levels(class)=="saab"] <- "
   car"
levels(class)[levels(class)=="opel"] <- "
   car"

table(fit3$classification,class)
    class
    bus car van
  1   0 317  75
  2 116 112 124
  3 102   0   0
```

The above result indicates that cluster 3 represents `bus`, whilst for cluster 2 there appears to be a large degree of overlap across all vehicle types, and cluster 1 partially represents `car` and `van`.

A Fantastic Winning Example of Unsupervised Learning You Can Emulate

Back in 2012 the first Multimodal Brian Tumor Segmentation (BRATS) challenge was issued[61]. Practitioners, researchers and others interested in the challenge formed "*BRAT packs*" to develop both fully automated and partially automated machine learning solutions.

NOTE... ✍

Multimodal imaging data is used to segment brain tumors; however, because tumors have a large variety of shapes and appearance, expert raters are typically used to manually annotate the images. Machine learning can play a role in automatically segmenting images and thereby improving the efficiency of the identification and diagnosis process.

In October 2012 at the Palais des Congrès Acropolis located in the beautiful French city of Nice, the results of the very first competition were announced[62]. Let's take a look at how one group of scholars rose to the challenge using unsupervised learning; and how you can too.

Javier Juan-Albarracín of the Universitat Politècnica de València along with other colleagues[63] developed an amazingly accurate process for unsupervised brain tumor segmentation. The process comprises four stages: Image preprocessing, feature extraction and dimensionality reduction, unsupervised voxel classification and automatic tumor classes isolation[64].

Data preprocessing is an important task for successful analysis, but rarely discussed in learning theory. As business man Tony Robbins says "*success leaves clues*." Let's take a quick look at how these world class researchers handled this. It will inform your practice.

Data (Image) Preprocessing

Dealing with various types of noise, and removing them, were important tasks undertaken by the researchers ahead of the use of unsupervised learning. The researchers had to overcome artificial noise in the data that made up an image; and also the fact that the data was not necessarily as clean as expected. So the researchers developed a skull stripping technique - removing extraneous objects from the image data.

Dealing with Noise in an Image

Magnetic resonance imaging (MRI) scanners use strong magnetic fields, radio waves, and field gradients to form images. These images are often corrupted by random noise from the acquisition process. Image preprocessing involves removing any noise from the image. The question is how to remove the noise?

One solution would be to take multiple images and use the average. Whilst this will work, it is also time consuming and therefore less efficient in a clinical setting.

A more efficient solution is to assume the noise follows a specific distribution and use a filter to remove it. For simplicity noise is often assumed to be Gaussian; however, in practice it has been shown to follow a Rician distribution[65]. Whatever distribution is assumed, denoising is typically carried out with a standard denoising filter[66]. Javier and colleagues remove noise from their images using a Manjón filter[67].

Another aspect of noise that requires attention in MRI images is "*intensity inhomogeneity*" - low frequency signals that corrupt an image by affecting its intensity levels. A number of bias field correction methods are popular. The researchers use the N4ITK intensity normalization algorithm[68]. To enhance the resolution of the sample images, which came at a 1mm^3 isotropic voxel size, the super resolution algorithm of Manjón et al.[69] was applied.

Takeaway: Whilst diminishing the effect of noise in a sample can be achieved by multiple acquisition and averaging, in practice filtering leads to more efficient image processing. Our role as data scientists is to help improve efficiency of processes using data and models. When a choice such as "averaging" or "filtering" occurs, the data scientist chooses the most practically efficient option.

Skull Stripping

Another component which is important in brain imaging is "*skull stripping*". This involves removing the skull, extra-meningeal and non-brain tissues from the images.

The original images presented to the researcher were supposed to have been fully stripped; however, several instances were observed in the sample by the researchers where skull, extra-meningeal and non-brain tissues were present. Such material could impact the performance of the unsupervised classifiers and therefore had to be removed. Figure 3.8, illustrates the situation for one image.

Takeaway: Even with a "*clean sample*", it is always advisable to check for noise, missing data or other artifacts which could impact the analysis.

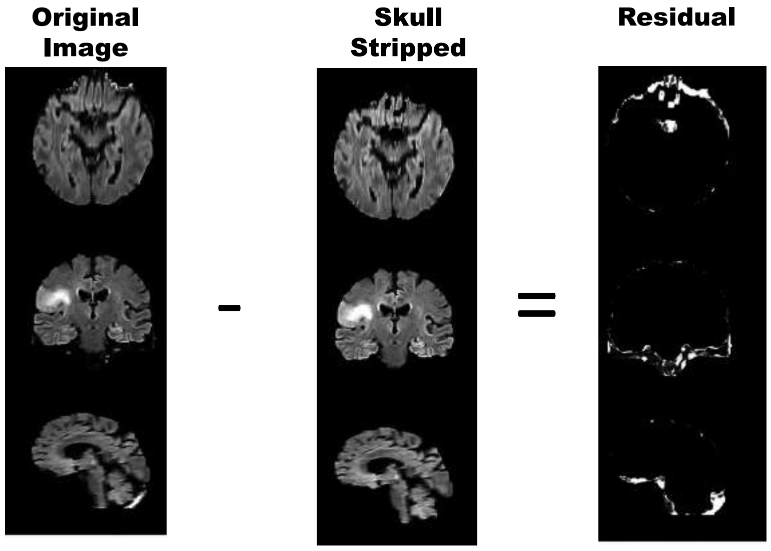


Figure 3.8: Original sample image, skull stripped imaged by researchers and the residual. Image adapted from Juan-Albarracín et al. Cited in Note sec. 63

The Wining Combination

Javier Juan-Albarracín and colleagues evaluated four unsupervised classification algorithms on the cleaned and processed data - K-means, Fuzzy K-means, Gaussian Mixture Model (GMM), and the Gaussian Hidden Markov Random Field (GHMRF).

The researchers comment "*Several unsupervised classification algorithms were evaluated to assess its pros and cons, ranging from the most restrictive algorithms in terms of class-conditional probabilistic models (K-means and Fuzzy K-means) to more sophisticated models with more degrees of freedom such as GMM or GHMRF. The last one, also introduces statistical dependencies between adjacent variables of the model, that penalizes neighbouring voxels with different classes. Hence, this structured prior aims to model the self similarity presented in* *the images, leading the algorithm to a more homogeneous segmentation than the non-structured classification techniques.*"

Figure 3.9 shows an example of the final segmentation for three patients (P1, P2, P3) using each method. In terms of the BRATS Challenge[70], the researchers "*...achieved the 1st position in the ranking of the unsupervised methods of the Test set, and the 7th position in the general ranking, mainly against supervised approaches.*"

Javier Juan-Albarracín and colleagues offer a perfect example of how to maximize the value of unsupervised learning. What ideas does this give you?

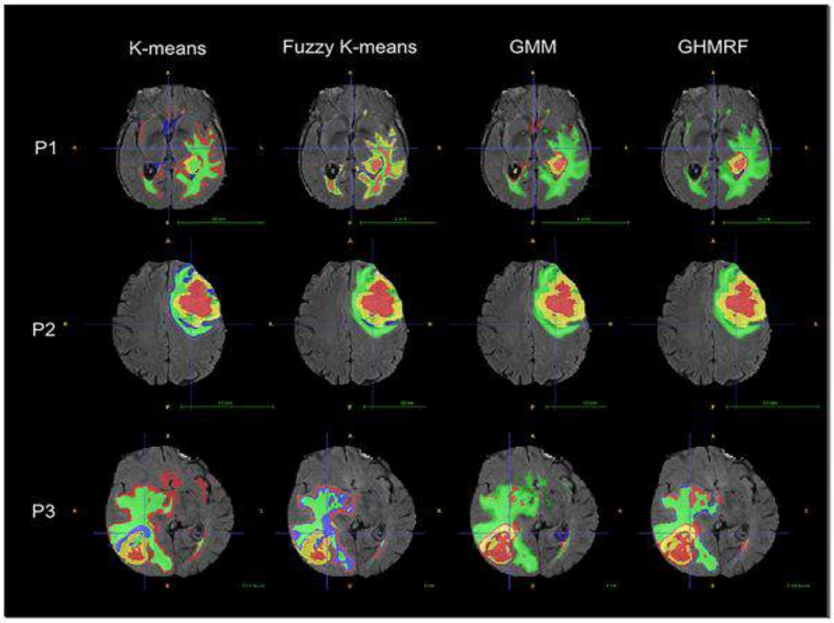


Figure 3.9: Examples of final segmentation of 3 patients of BRATS 2013 dataset computed by the different unsupervised algorithms. Image source Juan-Albarracín et al. Cited in Note sec. 63

Notes

[55]Costa, Kelton AP, et al. "A nature-inspired approach to speed up optimum-path forest clustering and its application to intrusion detection in computer networks." Information Sciences 294 (2015): 95-108.

[56]Pisani, Rodrigo Jose, et al. "Toward Satellite-Based Land Cover Classification Through Optimum-Path Forest." Geoscience and Remote Sensing, IEEE Transactions on 52.10 (2014): 6075-6085.

[57]Ng, H. P., et al. "Medical image segmentation using k-means clustering and improved watershed algorithm." Image Analysis and Interpretation, 2006 IEEE Southwest Symposium on. IEEE, 2006.

[58]This data set comes from the Turing Institute, Glasgow, Scotland.

[59]See for example McBratney, A. B., and JJ de Gruijter. "A continuum approach to soil classification by modified fuzzy k-means with extragrades." Journal of Soil Science 43.1 (1992): 159-175.

[60]For further details see - Murtagh, Fionn, and Pierre Legendre. "Ward's hierarchical agglomerative clustering method: Which algorithms implement ward's criterion?." Journal of Classification 31.3 (2014): 274-295.

[61]See http://braintumorsegmentation.org/

[62]See here to see the winners and rankings:http://www2.imm.dtu.dk/projects/BRATS2012/results.html

[63]Juan-Albarracín J, Fuster-Garcia E, Manjón JV, Robles M, Aparici F, Martí-Bonmatí L, et al. (2015) Automated Glioblastoma Segmentation Based on a Multiparametric Structured Unsupervised Classification. PLoS ONE 10(5): e0125143. doi:10.1371/journal.pone.0125143

[64]The sample data was taken from the public BRATS 2013 Test and Leaderboard datasets.

[65]When assuming a Gaussian distribution the power of the noise can be easily estimated from the standard deviation of the pixel signal intensity in an image region with no signal. This in theory can result in up to a 60% underestimation of the true noise power. For further details, see Gudbjartsson, Hákon, and Samuel Patz. "The Rician distribution of noisy MRI data." Magnetic resonance in medicine 34.6 (1995): 910-914.

[66]See Diaz, Idanis, et al. "A critical review of the effects of de-noising algorithms on MRI brain tumor segmentation." Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE. IEEE, 2011.

[67]The benefit of this filter is that it does not impose a uniform distribution noise over the image but rather adapts the strength of the filter depending on a local estimation of noise. It can also handle both correlated Gaussian and Rician noise. See:

Manjón, José, et al. "Adaptive non-local means denoising of MR images with spatially varying noise levels." Journal of Magnetic Resonance Imaging 31.1 (2010): 192-203.

[68]Tustison, Nicholas J., et al. "N4ITK: improved N3 bias correction." Medical Imaging, IEEE Transactions on 29.6 (2010): 1310-1320.

[69]Manjón, José V., et al. "Non-local MRI upsampling." Medical image analysis 14.6 (2010): 784-792.

[70]BRATS 2013 competition.

Chapter 4

Semi Supervised Learning

The first time I was in a statistics course, I was there to teach it.

John Tukey

In many situations, acquisition of a sufficient number of labeled data observations is difficult or costly. Computer scientists Avrim Blum and Tom Mitchell, writing in the late 1990's, gave a classic example[71]: "*Suppose that we want a program to electronically visit some web site and download all the web pages of interest' to us, such as all the CS faculty member pages, or all the course home pages at some university. To train such a system to automatically classify web pages, one would typically rely on hand labeled web pages. These labeled examples are fairly expensive to obtain because they require human effort. In contrast, the web has hundreds of millions of unlabeled web pages that can be inexpensively gathered using a web crawler. Therefore, we would like our learning algorithm to be able to take as much advantage of the unlabeled data as possible.*"

Scholars John Lafferty, Zoubin Ghahramani and Xiaojin Zhu report that "*obtaining a single labeled example for protein shape classification, which is one of the grand challenges of biological and computational science, requires months of expen-*

sive analysis by expert crystallographers." In situations such as these semi-supervised learning techniques can bring a clear benefit in terms of reduced time and cost of acquiring expertly annotated datasets[72].

Semi-supervised learning uses a large amount of unlabeled data together with a much smaller quantity of labeled data. Given a set of labels for each class $L = \{1, ...K\}$, only l ($l < N$) points have labels $\{(x_1, y_1), ..., (x_l, y_l)\}$ with $\{x_u, ..., x_N\}$being unlabeled ($u = l + 1$). The goal, of course, is to accurately predict the labels of the unlabeled points.

NOTE... ✍

The distinction between supervised and semi-supervised learning was made in Stanley Fralick's 1967 technical report[73] "*Learning to Recognize Patterns Without a Teacher*". It remains a fascinating read.

How Can Unlabeled Data Can Help?

The situation for a binary classification problem is illustrated in Figure 4.1. In this illustration the dotted circles represent the unlabeled data, whilst the solid circles represent labeled observations from class -1 and +1 respectively. The classifier seeks to determine an appropriate boundary between the classes, denoted by the dotted vertical line. It seems clear the use of unlabeled observations can help in the determination of the boundary.

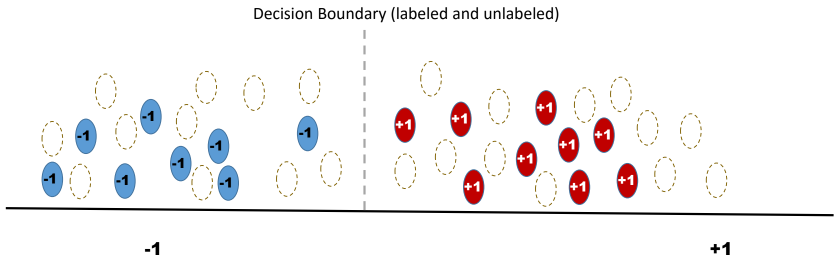


Figure 4.1: Binary classification problem

Might it even be possible to improve on a supervised classifier with the use of unlabeled data? Possibly, take for example the situation, discussed by Belkin et al[74] and shown in Figure 4.2. In the left panel we are given the set of labeled points, and our intuition is to use a linear boundary to separate the two groups. Suppose that we are given a large number of additional unlabeled points, as shown in the right panel. A more productive decision boundary appears to be circular. As Belkin et al comment "*...it is self-evident that in the light of this new unlabeled set, one must re-evaluate one's prior notion of simplicity* [linear decision boundary]. *The particular geometric structure...suggests that the most natural classifier is now the circular one...*".

Imagine, with a mixture of labeled and unlabeled data not only equaling but exceeding the performance of supervised classifiers! Sound a bit like a free lunch? Well, as the notice in the refectory at my undergraduate campus stated in big bold letters "ALL LUNCHES HAVE TO BE PAID FOR". This is true in machine learning (and for that matter life too!).

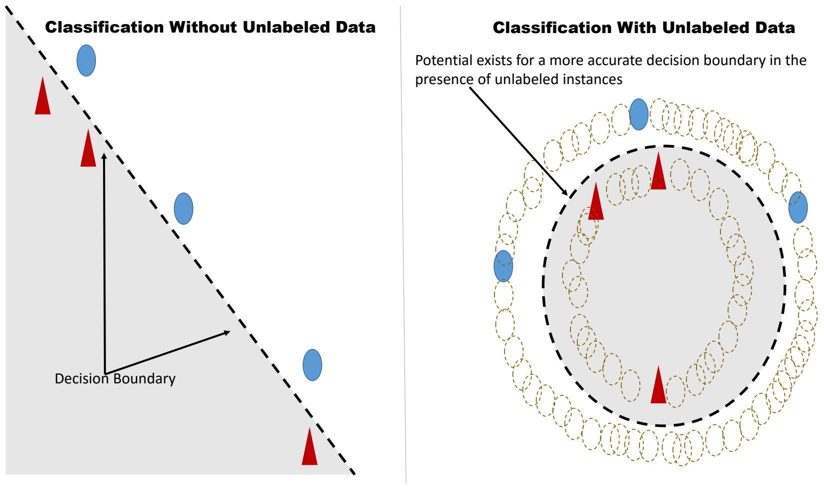

Classification Without Unlabeled Data
Decision Boundary
Classification With Unlabeled Data
Potential exists for a more accurate decision boundary in the presence of unlabeled instances

Figure 4.2: Improved classification with semi-supervised learning

NOTE... ✍

One popular rule of thumb is to set the training sample to 10 to 30 times the number of classes[75]. This suggests that for a supervised classifier to perform well, it is necessary to be trained on hundreds, or even thousands, of labeled instances. This may be in-feasible. Consider some of the classification applications in your own line of research. How easy is it to obtain large numbers of classified labels?

The Consistency Assumption

So what is the price we have to pay by using semi-supervised learning? First, there is no guarantee that using unlabeled data will always work. Returning back to the elements of the learning problem outlined on page 9, we can at least say that any knowledge on $P(x)$ that we gain through the unlabeled data has to covey information that is useful in the inference of $P(y|x)$. If this is not the case, then knowledge of $P(x)$ is unlikely to be of much use.

Second, consistency of the data is a key requirement. It is typical characterized as consisting of:

1. **Local consistency:** Where nearby points are likely to have the same label. This implies if two points x_1 and x_2 are close to each other, then so should be the corresponding outputs of y_1 and y_2. So for example, if x_1 and x_2 are close in the intrinsic geometry of $P(x)$ then the conditional distributions $P(y|x_1)$ and $P(y|x_2)$ are assumed to be similar.

2. **Cluster consistency**: Points in the same structure (such as a cluster or manifold[76]) are more likely to have the same label.

Clearly, without the assumption of consistency, it would not be possible to generalize from a finite training set to a set of possibly infinitely many unseen test cases.

A Super Simple Way to Try Semi-supervised Learning

Here is a simple way to carry out semi-supervised learning in practice. Run your favorite clustering algorithm on x_l and x_u. Then label all points within a cluster by the majority of labeled points in that cluster.

Figure 4.3 illustrates the situation. Three clusters are identified. In the left panel we observe, for instance that cluster 1 contains two unlabeled points, with the majority of points labeled as "*solid circle*", therefore the points in this cluster are labeled "*solid circle*" (right panel). In cluster 3 (left panel), the majority of points are unlabeled, however one point is labeled *"checked circle"*, and therefore points in this cluster are labeled *"checked circle"*.

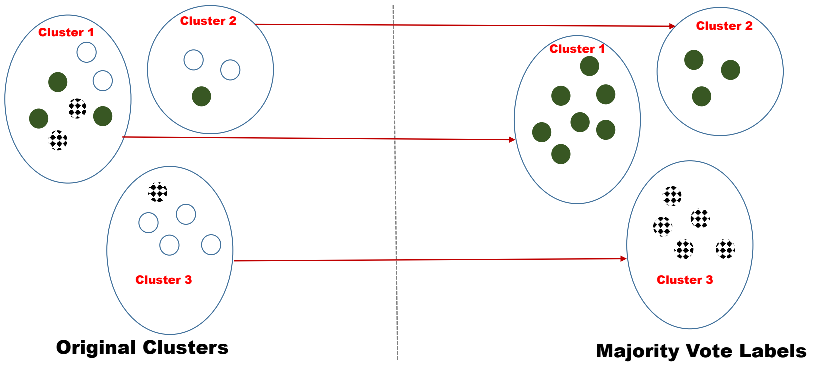


Figure 4.3: Simple rule for semi-supervised labeling

The Self Learning Algorithm

The above approach is a precursor for the self learning algorithm. It is a popular and very practical approach to semi-supervised learning in which the predictor $h(x)$ is first trained with a small amount of labeled data, and then used to classify the unlabeled data. It became popular after considerable success in natural language processing[77]; and follows four general steps:

Step 1: Train $h(x)$ using the labeled pairs (x_l, y_l).

Step 2: Predict x_u

Step 3: Add $(x_u, h(x))$ to the labeled data

Step 4: Repeat

Let's build a self learning classifier using R. For the illustration I use the `thyroid` dataframe and the naive Bayes classifier previously discussed on page 35. First load the data and create a training sample of 150 observations:

```
data("thyroid",package="mclust")
data<-thyroid
set.seed(2016)
N=nrow(thyroid)
train<-sample(1:N,150,FALSE)
```

Since the class variable (`Diagnosis`) has labels (which won't be the case in practice), I randomly select 100 observations from the training set and replace the labels with missing values (`NA`):

```
data_modified <- data[train,]
missing <- sample(150,
size=100,
replace =FALSE)
data_modified[missing,"Diagnosis"] <- NA
```

The R object `data_modified` contains the class variable `Diagnosis` with the missing labels. As a check type:

```
sum(is.na(data_modified$Diagnosis))
[1] 100
```

To build a self learning classifier in R you can use the `SelfTrain` function from the `DMwR` package. Naive Bayes is accessible via the `naiveBayes` function in package `e1071`:

```
library(DMwR)
library (e1071)
```

The `SelfTrain` function requires you to create a function that takes the prediction probabilities from your chosen classifier, and for each observation returns the class label with the highest probability. Here is how to do this for `naiveBayes`:

```
nbayes <- function(mod,dat) {
   probs <- predict(mod,dat,
   type="raw")
   data.frame(class=colnames(probs)
[apply(probs,1,which.max)],
   probs=apply(probs,1,max))
}
```

Let's go through this in some detail. The function is called `nbayes` with the `probs` object holding the predicted class label probabilities via `type="raw"`. The data frame is created using three elements. The first element uses the `colnames` method to assign column names to the dataframe using the classes contained in `prob`. For the `thyroid` data set these classes are `Hypo`, `Normal` and `Hyper`.

The second element uses the `apply` function and `which.max` parameter to indicate which column (index) contains the largest probability.

The third element again uses the `apply` method, but this time with the `max` parameter to choose the maximum value from a row.

The `SelfTrain` function can be called as follows:

```
fit <- SelfTrain(Diagnosis ~ .,
data=data_modified,
learner("naiveBayes",
list()),"nbayes")
```

The object `fit` contains the fitted model. Predictions using the test set and the confusion matrix are obtained as follows:

```
pred<-predict(fit,data[-train,2:6])
table(pred,data$Diagnosis[-train])
```

```
pred      Hypo Normal Hyper
  Hypo       8      0     0
  Normal     1     49     1
  Hyper      0      0     6
```

It is interesting to observe that the confusion matrix is similar to that obtained using the full labeled dataset on page 40.

NOTE... ✍

There is a risk that early mistakes in classification could reinforce themselves and therefore diminish the utility of the approach. However, a number of variations to reduce the chance of this have been proposed. A common solution is to add only the most confident $(x_u, h(x))$ to the training data. Another variant adds all the $(x_u, h(x))$ to the labeled data, each weighted by confidence.

Semi-Supervised Model Based Learning with R

Let's build a semi-supervised model using an approach related to the model based clustering technique we discussed on page 94. We will use the `Vehicle` data set for our analysis. First we load the data and make suitable transformations:

```
data("Vehicle",package="mlbench")
data<-as.matrix(Vehicle[,-19])
set.seed(2016)
class<-Vehicle[,19]
levels(class)[levels(class)=="saab"] <- "
   car"
levels(class)[levels(class)=="opel"] <- "
   car"
N=nrow(data)
```

Next, the training and testing samples need to be defined:

```
train <- sort(sample(1:N, N * 0.3))
data_train <- data[train,]
class_train <- class[train]
```

```
test <- setdiff(1:N, train)
data_test <- data[test,]
class_test <- class[test]
```

Here we assume 30% of the data is labeled, with the remaining unlabeled. The R object `data_test` holds the unlabeled test data. We also use `class_test` to hold the known class labels, and we will use these to evaluate the misclassification rate.

Model based semi-supervised clustering can be carried out using the `upclass` package with the `upclassify` function:

```
library(upclass)
fit <- upclassify(data_train,
class_train, data_test,
class_test)
```

In practice, you will not have `class_test` and so it is optional. The `upclassify` function uses the log likelihood and the expectation–maximization algorithm to obtain maximum likelihood estimates for the unknown parameters of the model. The algorithm then uses these estimated parameters to classify the unlabeled data. The optimal model is returned using the Bayesian information criterion:

```
summary(fit)

Model Name
 VVV
Log Likelihood
 -39972.33
Dimension
 18
Ntrain
 253
Ntest
 593
bic
 -83780.01
```

```
Total Misclassified:  35
Misclassification Rate:   5.902 %
```

The optimal model (`VVV`) has a BIC of -83780, a error rate of 5.9% with 35 observations out of the test set of 593 observations misclassified.

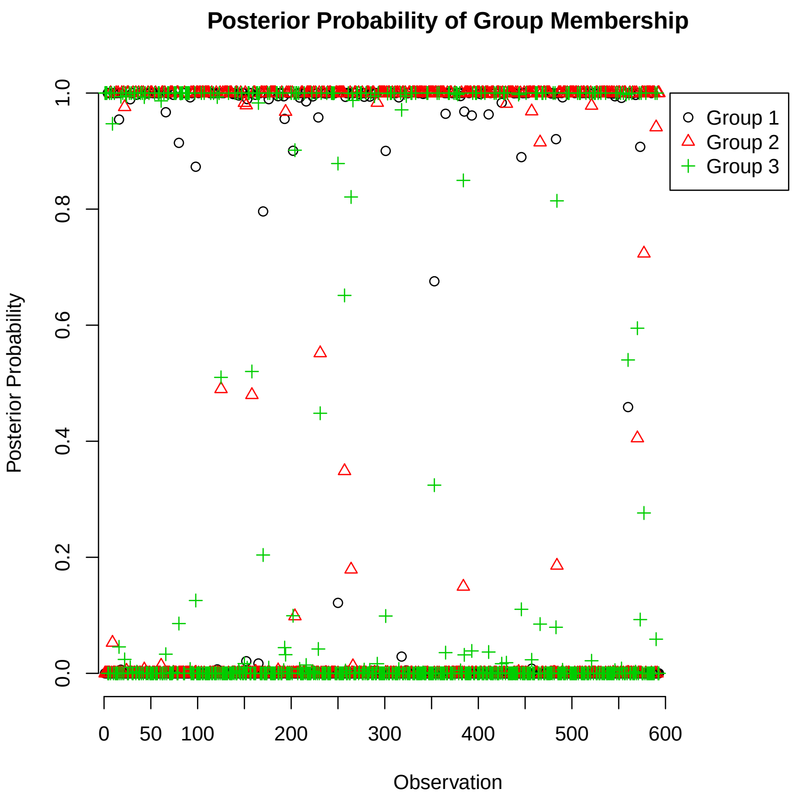


Figure 4.4: Posterior Probability of Group Membership

Figure 4.4 shows the posterior probability of group membership plot. It gives the z-values of the test set. Most of the points are classified with a value close to 1 or 0, indicating a very high probability of belonging to the assigned cluster. However, there are a number of observations in the 0.4 to 0.6 range, indicating greater uncertainty as to their true classification.

NOTE... ✍

The upclassify function fits the following Gaussian mixed component models -"EII", "VII", "EEI", "VEI", "EVI", "VVI", "EEE", "EEV", "VEV", "VVV". See page 94 for further details.

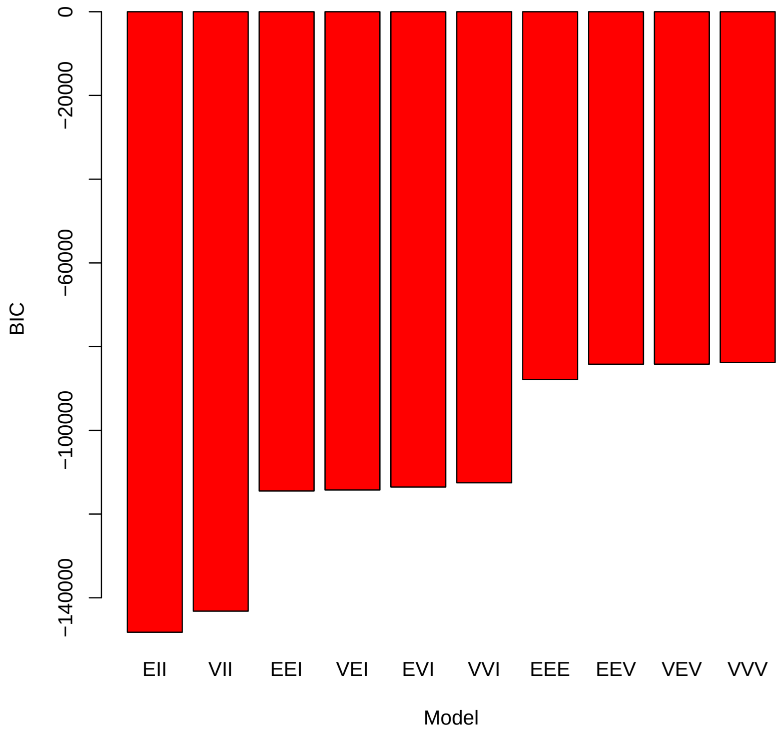


Figure 4.5: BIC by model

It is always a good idea to look at the BIC values of all the models estimated. This is because the optimal value is often times only a shade higher than other "close" models which have a better misclassification rate. Figure 4.5 shows that the optimal model is only a touch higher than two other models (`VEV` and `EEV`). Let's take a look at the misclassification rates of these two models:

```
n<-length(test)
fit[["VEV"]]$test$mis/n*100
[1] 4.384486

fit[["EEV"]]$test$mis/n*100
[1] 4.215852
```

Both models have a lower misclassification rate than `VVV`.

In practice you will not have access to labeled data for the test set. Fortunately, you can use the Brier score to approximate the misclassification rate:

$$B = \frac{100}{2N} \sum_{g=1}^{G} \sum_{i=1}^{N} \left(l_{ig} - \pi_{ig}\right)^2 ,$$

where l_{ig} are the labels assigned by the model to each observation and π_{ig} are their a posterior probabilities.
The Brier score equals zero for a perfect prediction. We can use it as an approximation to the misclassification rate. Here are the values for the each of the models:

```
fit[["Best"]]$test$Brierscore
[1] 5.15087

fit[["VEV"]]$test$Brierscore
[1] 4.041225

fit[["EEV"]]$test$Brierscore
[1] 3.864652
```

They are all slightly lower than the actual misclassification rates from each model. But nonetheless, it serves as a useful proxy.

Master this Practical Illustration using Land Classification

Let's look at a very practical application of semi-supervised learning. Back in 1086, the finishing touches were being made to the Great Survey (more commonly known as the Doomsday book). It had been ordered by King William the Conqueror to figure out who owed what when it came to taxes.

King William's primary motivation was maximizing his taxation revenue from the peasants and landed classes. For the small English hamlet of Bedworth, Warkwickshire (see Figure 4.6) the survey indicates a population of 10 head of households, with a total tax assessed as 4 geld. In terms of land use, 6 ploughlands, 16 acres of Meadow with 1.5 leagues of Woodland.

King William's Doomsday book was the first recorded land use survey in Great Britain. Accurate assessment of land use is important for understanding how human activity and natural factors impact a region. Knowledge of current land, and changes over time is useful for formulating sustainable use of land resources.

Today, land use data can be rapidly obtained by remote sensing using hyperspectral sensors. Researchers Chunyang Wang and colleagues[78], primarily of Henan Polytechnic University[79], the first dedicated mining university in Chinese history and the oldest higher college in Henan province, use semi-supervised self learning for land use classification.

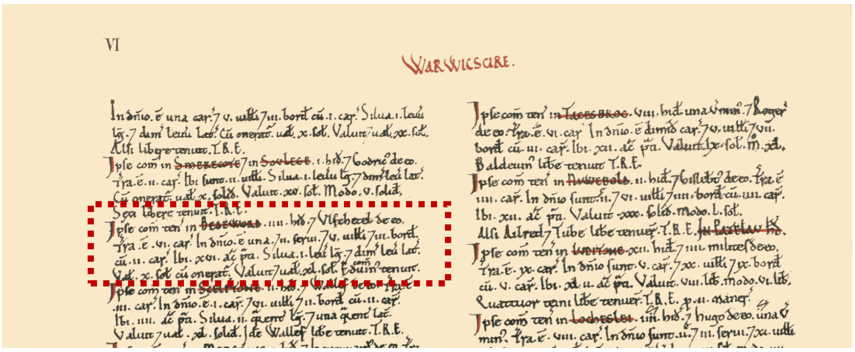


Figure 4.6: Doomsday Book page for Warwickshire. The small town of Bedworth is highlighted. *Source http://opendomesday.org/*

The study region consisted of an area of the Inner Mongolia autonomous region of China. Data was obtained from the EO-1 satellite Hyperion sensor with a total of 124 out of 242 bands selected for analysis.

Here is how the researchers set up the problem for successful semi-supervised learning (see page 9):

1. **Denote the target variable** y and appropriate classes. The researchers characterized k labeled responses $\{y_1, ..., y_k\}$ composed of a total of C class labels.

2. **Assume a probabilistic relationship** exists between x and y captured by an unknown probability distribution: $P(x, y) = P(x)P(y|x)$. The researchers us multinomial logistic regression to model $P(y|x)$ directly.

3. **Observe a sample**. In this case a set of k labeled observations $\{(x_1, y_1), ..., (x_k, y_k)\}$ consisting of the images (x) and labels (y)[80]. The unlabeled data consists of $\{x_{k+1}, ..., x_u\}$.

The self learning approach adopted by Wang and colleagues followed similar steps to that outlined on page 112. The most likely $(x_u, h(x))$ were added to the training data using the normalized Rény entropy (with $\alpha = 2$) as a measure of confidence[81]:

$$R_\alpha(X) = -P(x_i) \ln \left(\sum_{i=1}^{n} P(x_i)^\alpha \right)$$

Figure 4.7 shows the original study region alongside the image annotated with the training sample sites used by the researchers. Figure 4.8 shows the original image alongside the semi-supervised reproduction. Visually, at least, the representation appears pretty close.

The researchers also compared the semi-supervised multinominal logistic regression to fully supervised neural network and support vector machine. Figure 4.9 displays the result. Visually, all the images appear pretty similar, with perhaps the support vector machine and semi-supervised method being the most similar. Wang et al report that the overall accuracy of the neural network was around 93%, a little lower than the support vector machine at around 95%. However, the semi-supervised multinominal logistic regression had an accuracy a touch over 97%. In this example a mixture of labeled and unlabeled data not only equaled but exceeded the performance of supervised classifiers. The researchers conclude by noting "*The experimental results show that the method has a high precision overall classification.*"

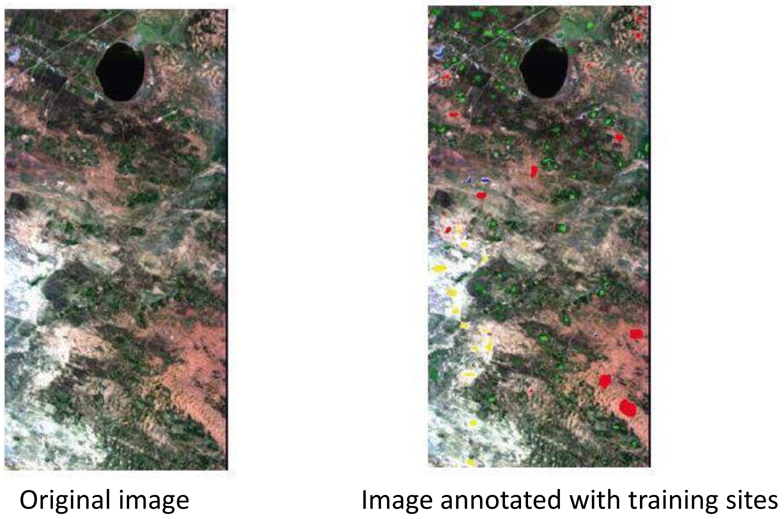


Figure 4.7: Original image (left) and image annotated with training sites (right). Source adapted from Wang et al. Cited in Note sec. 78.

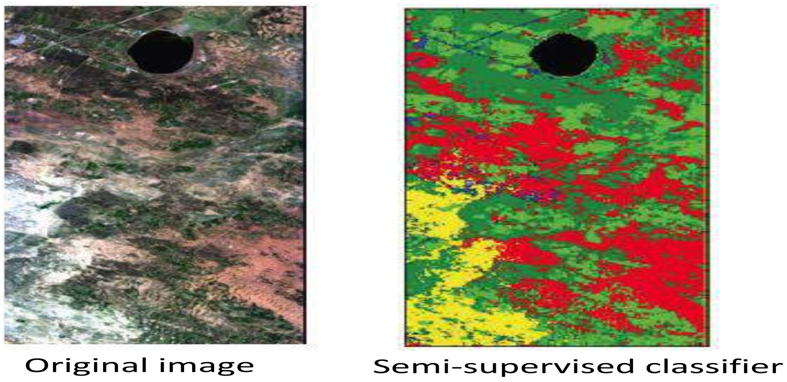


Figure 4.8: Original image (left) and Semi-supervised classifier (right). Source adapted from Wang et al. Cited in Note sec. 78.

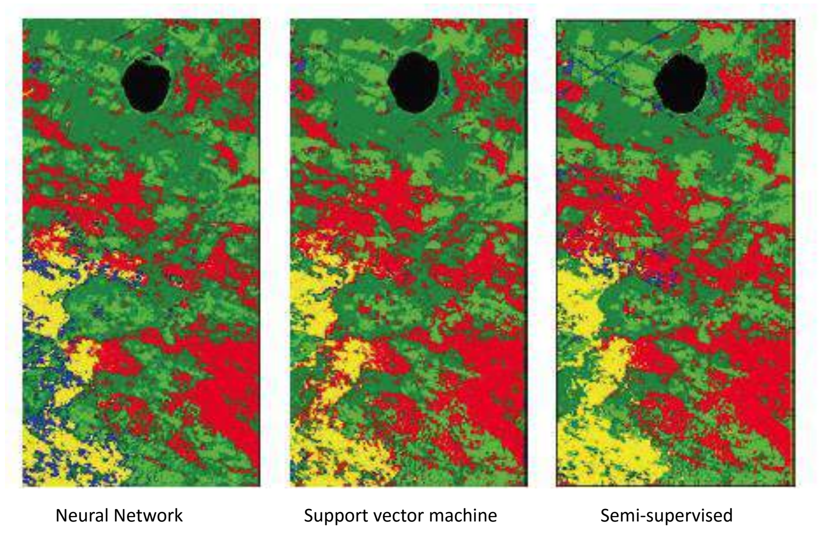


Figure 4.9: Semi-supervised methods alongside fully supervised methods. Source adapted from Wang et al. Cited in Note sec. 78.

Notes

[71]Blum, Avrim, and Tom Mitchell. "Combining labeled and unlabeled data with co-training." Proceedings of the eleventh annual conference on Computational learning theory. ACM, 1998.

[72]See for example, Lee, Chi-Hoon, et al. "Learning to model spatial dependency: Semi-supervised discriminative random fields." Advances in Neural Information Processing Systems. 2006.

[73]See Fralick, Stanley C. "Learning to recognize patterns without a teacher." Information Theory, IEEE Transactions on 13.1 (1967): 57-64.

Also see the earlier paper by Henry Scudder:

Scudder, Henry J. "Probability of error of some adaptive pattern-recognition machines." Information Theory, IEEE Transactions on 11.3 (1965): 363-371.

[74]See the amazing paper by Ohio State Professor of Statistics Professor, Computer Science & Engineering Mikhail Belkin, the amazingly wonderful and sadly late Partha Niyogi - Louis Block Professor in Computer Science and Statistics at the University of Chicago; and Machine Learning and Intelligence guru Vikas Sindhwani.

See: Belkin, Mikhail, Partha Niyogi, and Vikas Sindhwani. "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples." The Journal of Machine Learning Research 7 (2006): 2399-2434.

[75]See for example:

- Mather, Paul, and Magaly Koch. Computer processing of remotely-sensed images: an introduction. John Wiley & Sons, 2011.
- Piper, Jim. "Variability and bias in experimentally measured classifier error rates." Pattern Recognition Letters 13.10 (1992): 685-692.

[76]The data are assumed to lie on a sub-manifold within a high dimensional representation space.

[77]A useful overview is given in Baker, Janet M., et al. "Developments and directions in speech recognition and understanding, Part 1 [DSP Education]." Signal Processing Magazine, IEEE 26.3 (2009): 75-80.

Also take a look at Yarowsky, David. "Unsupervised word sense disambiguation rivaling supervised methods." Proceedings of the 33rd annual meeting on Association for Computational Linguistics. Association for Computational Linguistics, 1995.

[78]Chunyang Wang, Zengzhang Guo, Shuangting Wang, Liping Wang, and Chao Ma, “Improving Hyperspectral Image Classification Method for Fine Land Use Assessment Application Using Semisupervised Machine Learning,” Journal of Spectroscopy, vol. 2015, Article ID 969185, 8 pages, 2015. doi:10.1155/2015/969185

[79]Originally known as the Jiaozuo Coal Mining School, established in 1909 by the British Syndicate Company. Ltd.

[80]In the study *d* represents the number of spectral bands, each pixel represents a vector.

[81]Rényi, A. "Some fundamental questions of information theory MTA III Oszt. Közl. 10 251 Rényi A 1960 On measures of information and entropy." Proceedings of the Fourth Berkeley Symposium on Mathematics, Statistics and Probability. 1960.

Chapter 5

Statistical Learning Theory

"In theory, theory and practice are the same. In practice, they are not"

R. Ahlswede

STATISTICAL Learning Theory (SLT) caught my eye several years ago. As I read about this incredible approach I became very excited. Through it, a new tool, known as a Support Vector Machine (see page 136), was developed and looked very promising. It is not often a brand new methodology comes along that has the potential to significantly alter data science practice, and here it was, with a practical tool for me to play with!

Unlike classical econometrics, a sub-branch of statistics, the form of the "true model" is accepted as unknown in SLT; and inference comparisons between models are based on finite samples rather than asymptotic statistics. Econometricians spend considerable amounts of energy trying to uncover the underlying data generating mechanism and correct model specification. Advanced courses on how to do this are taught at the great academic institutions across the globe; and many trained Econometricians pass their entire careers focused on this one single task.

However, in SLT there is no attempt whatsoever to uncover the "true" model. Rather, the goal is simply to identify the best possible model from a given set of models. In essence SLT is a series of mathematical tools for comparing different models, say say$\{{}^1\hat{y}(\theta), ..., {}^m\hat{y}(\theta)\}$, estimating their relative performance and selecting the strongest candidate. Sounds familiar? Well, it's kind of what Data Scientists do on a daily basis! Now you can see why I was so excited to discover SLT! Let's look at how you can use it in practice.

NOTE... ✍

The emphasis in classical statistics is often placed on testing nested models - parametric models of the same type who differ in the values of the parameters; in SLT the models need not even be of the same form. As IBM researchers Jonathan Hosking, Edwin Pednault, and Madhu Sudan explain[82]: "*In the classical* [Statistics] *approach, the form of the model is assumed to be known and, hence, emphasis is placed on estimating its parameters. In statistical learning theory, the correct model is assumed to be unknown and emphasis is placed on estimating the relative performance of competing models so that the best model can be selected.*"

The Vapnik-Chervonenkis Generalization Bound

SLT assumes the attributes x are drawn independently from a fixed but unknown distribution and the output y is from a probability distribution function conditional on x, also fixed but unknown. The relative performance of competing models is measured through empirical risk $R_{emp}(\theta)$ because we cannot calculate the expected loss $R(\theta)$ directly (see page 11 for a recap). We really want to find a bound on the risk such that with high probability:

$$R(\theta) \leq R_{emp}(\theta) + \delta,$$

The hope is that δ is not too large. This will at least inform us how good our empirical risk measure performs as an estimate of the true risk; and we can select the model which minimize this quantity.

All of this leads to one of the key theorems of Statistical Learning Theory that an upper bound for the expected risk $R(\theta)$ can be obtained using empirical risk $R_{emp}(\theta)$. Russian Mathematicians Vladimir Vapnik and Alexey Chervonenkis showed, in the context of two class classification, that with probability $1 - \eta$:

$$R(\theta) \leq R_{emp}(\theta) + \sqrt{\left(\frac{\mathtt{h}\left(\log\left(\frac{2N}{\mathtt{h}}\right)+1\right) - \log\left(\frac{\eta}{4}\right)}{N}\right)} \quad (5.1)$$

where $\mathtt{h}$ is a non-negative integer called the Vapnik-Chervonenkis dimension. The right side of 5.1 is known as the "*risk bound*" or sometimes "*guaranteed risk*". The second term on the right hand side is called the Vapnik-Chervonenkis confidence which depends on the chosen class of models.

NOTE... ✍

In the definition of empirical risk on page 13, no probability distribution is involved, this implies that $R_{emp}(\theta)$ is a fixed number for a particular choice of θ and for a particular training sample.

What is the Vapnik-Chervonenkis Dimension?

The VC dimension of a set of functions $\{{}^{1}\hat{y}(\theta), ..., {}^{m}\hat{y}(\theta)\}$ is the largest number of points that can be separated (i.e. shattered) in all possible ways by members of $\{{}^{1}\hat{y}(\theta), ..., {}^{m}\hat{y}(\theta)\}$. To get the basic idea take a look at Figure 5.1. It contains three data points represented by two classes of triangles ("empty" and "solid") fitted using the class of straight line models $\{{}^{1}\hat{y}(\theta), ..., {}^{m}\hat{y}(\theta)\}$. The straight lines in the panels can separate the data into at most three points; hence the VC dimension of the class of straight lines in the plane is three.

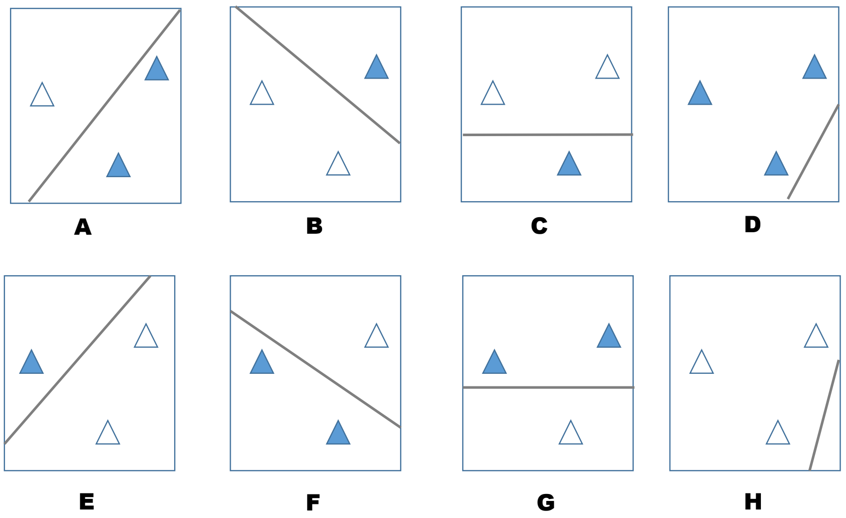


Figure 5.1: Shattering of at most three points

Figure 5.2, illustrates that in the case of four points in a plane, two cases cannot be separated with a linear classifier (panels E and F). These two cases require a classifier of higher complexity, with a higher VC dimension. A nonlinear curve could separate the points, in which case it must have a VC dimension greater than three.

NOTE... ✍

The VC confidence depends on the chosen class of functions $\{{}^1\hat{y}(\theta), ..., {}^m\hat{y}(\theta)\}$, whereas the empirical risk $R^i_{emp}(\theta)$ and actual risk $R^i(\theta)$ depend on the one particular function chosen, say for example ${}^i\hat{y}(\theta)$.

The key point is that no matter how we assign the labels to points, there's always a hypothesis in $\{h_1(x|\hat{\theta}), h_2(x|\hat{\theta}), ...h_m(x|\hat{\theta})\}$ which "*explains*" the labeling perfectly. The VC-dimension of a hypothesis class can therefore be understood conceptually as the maximum number of data points for which with high probability you are guaranteed to find a model that fits the data perfectly.

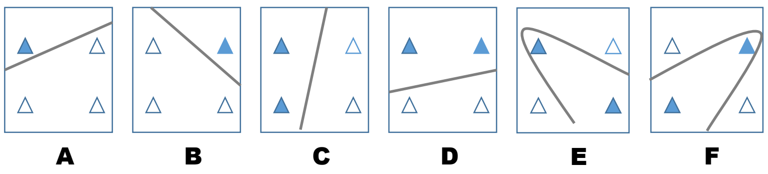


Figure 5.2: The situation with four points

Intuitively speaking, the VC dimension of a set of functions is a measure of their capacity or complexity. It measures the ability of $\hat{y}$ to model increasingly more complex classifiers. A higher ability allows modeling of more complex relationships as shown in Figure 5.3. I love the way Stanford University professors Trevor Hastie, Robert Tibshirani, and Jerome Friedman explain it in their excellent book[83] it "*...is a way of measuring the complexity of a class of functions by assessing how wiggly its members can be.*"

NOTE... ✍

Bottom line: If you can describe a lot of different phenomena with $\{{}^1\hat{y}(\theta), ..., {}^m\hat{y}(\theta)\}$ then the VC dimension is large.

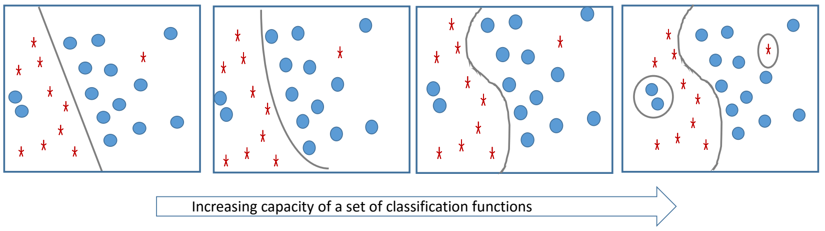


Figure 5.3: Illustration of increasing VC dimension

NOTE... ✍

Here is another way to think about the VC bound:
$Test\ Error \leq Train\ Error + Model\ Class\ Complexity$
The training error depends on the selected model. The complexity depends on the specific class of models (linear regression, non-linear regression and so on).

If the family of models $\{{}^1\hat{y}(\theta), ..., {}^m\hat{y}(\theta)\}$ is so flexible that it performs without error on the training sample, then it must be fitting the noise as well as the underlying relationships. As a result, you end up with a model that looks great relative to the training data, but then performs extremely poorly when applied to new data. Less capacity is less likely to over-fit, but restricts what can be modeled.

Bell Laboratories researcher Christopher Burges has a fun way of explaining capacity[84] "*...too much capacity is like a botanist with a photographic memory who, when presented with a new tree, concludes that it is not a tree because it has a different number of leaves from anything she has seen before; a machine with too little capacity is like the botanist's lazy brother, who declares that if it's green, it's a tree. Neither can generalize well.*"

The theoretical value of the VC dimension stems in part from its shear flexibility. As IBM researchers Jonathan Hosking, Edwin Pednault, and Madhu Sudan explain[sec. 82]: "*Because VC dimension is defined in terms of model fitting and number of data points, it is equality applicable to linear, nonlinear and nonparametric models, and to combinations of dissimilar model families. This includes neural networks, classification and regression trees, classification and regression rules, radial basis functions, Bayesian networks, and virtually any other model family imaginable.*"

The Key to Structural Risk Minimization

The overall approach to structural risk minimization is illustrated in Figure 5.4. A nested sequence of models of increasing VC dimensions $h_1(x|\hat{\theta}) < h_2(x|\hat{\theta}) < \cdots < h_m(x|\hat{\theta})$ are fitted to the test sample. Given several different hypothesizes, structural risk minimization chooses the model which gives the lowest upper bound on the actual risk (denoted by h_* and $^*\hat{y}$ in Figure 5.4).

In practice there is a delicate balance to be struck between model complexity and risk. If you use a high capacity set of models you get a low training error; however, you might overfit. If on the other hand you select a low capacity set of models, you won't achieve a low training error. Structural risk minimization is a worst-case model selection criteria because, you choose the learning machine which minimizes the upper bound on risk.

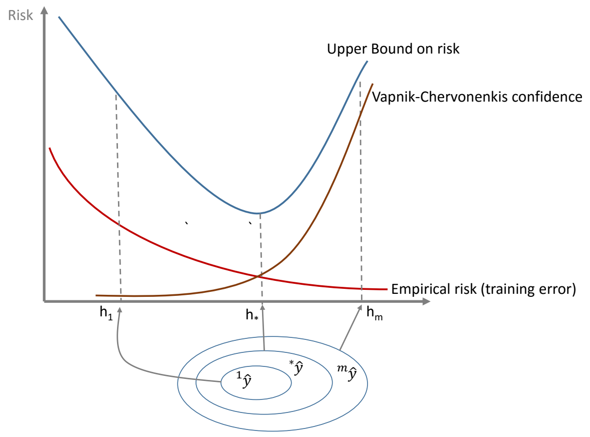


Figure 5.4: Structural risk minimization

NOTE... ✍

The best generalization performance, in terms of the ability to learn any training set without error, is achieved when the right balance is struck between accuracy attained on a specific training sample and capacity.

The Best Advice on Using Statistical Learning Theory in Practice

Alas, SLT is not the "sliver bullet" for data science I had hoped for when I first discovered it. One practical drawback to the

Vapnik-Chervonenkis approach is that it can be difficult to determine the VC dimension of a set of models. This issue has blighted its widespread adoption as Stanford University professors Trevor Hastie, Robert Tibshirani, and Jerome Friedman discover in trying to use SLT to build a kNN model[85] *"For k-nearest neighbors, we used the quantity N/k...we do not know if it corresponds to the VC dimension."* Other researchers make simplifying assumptions for example, Giorgio Corani and Marino Gatto in their study of approaches to density dependence and model selection in demographic models state[86] "*For the sake of simplicity, we assume...h coincides with the number of parameters of the model, i.e. $h = d$.*"

Furthermore, it is also possible for a particular learning machine, say ${}^{i}\hat{y}(\theta)$ to have the same empirical risk as, say ${}^{j}\hat{y}(\theta)(i \neq j)$, whose function set has higher VC dimension and much better performance. For example, the kNN classifier (see page 41) with $k = 1$ has infinite VC dimension and zero empirical risk. This is because, provided no two points of the opposite class lie on top of each other, any number of points, labeled in any configuration, will be successfully learned by the algorithm.

The fact that a classifier has infinite VC dimension does not guarantee poor performance. My best advice is that offered by many a professor of statistics that models with fewer parameters or degrees of freedom are preferable to those with more, since they are less likely to overfit the data. This brings up an important practical point, it is possible for the risk bound to be greater than 1, in which case Structural Risk Minimization is useless as a model selection criteria. More practical model selection techniques are discussed in chapter 6. It is in the development of the Support Vector Machine where SLT has had the most practical impact on data science.

How to Master the Support Vector Machine

The support vector machine, which I first came across through a series of heavy weight papers by Professor Vladimir Vapnik and others at Royal Holloway, University of London[87], is a supervised machine learning algorithm that can be used for both regression and classification. What was slightly unusual, and caught my eye, was that the support vector machine (SVM) was explicitly based on a theoretical model of learning; came with certain theoretical guarantees about performance; did not suffer from the curse of dimensionality and unlike the neural networks I was building at the time, was not susceptible to local minima.

In practice, as illustrated in Figure 5.5, the majority of classification problems require nonlinear boundaries in order to determine the optimal separation[88]. The core idea of SVMs is that they construct an optimal hyperplane for linearly separable data in a multidimensional space using a kernel function. Figure 5.6 illustrates how SVMs achieve this. The left side of the figure represents the original sample (known as the input space) which is mapped to a feature space of a higher dimension[89] before performing a linear classification in that higher dimensional space. The process of rearranging the objects for optimal separation is known as transformation and occurs through the use of a kernel function. Notice that in mapping from the input space to the feature space the mapped objects are linearly separable in the new space.

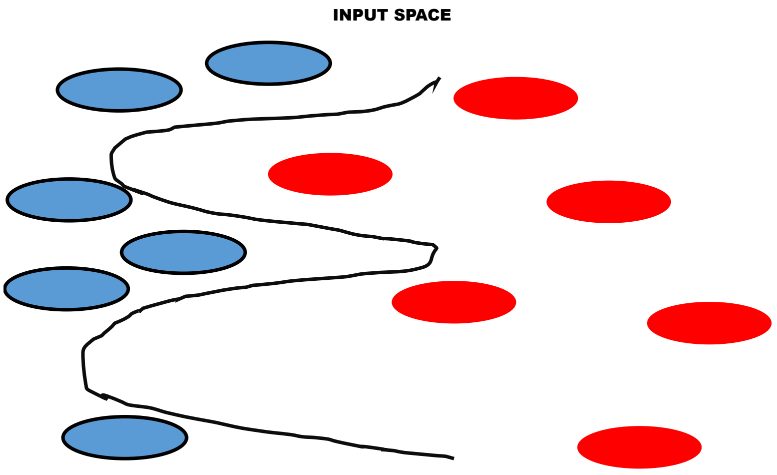


Figure 5.5: Nonlinear boundary required for correct classification

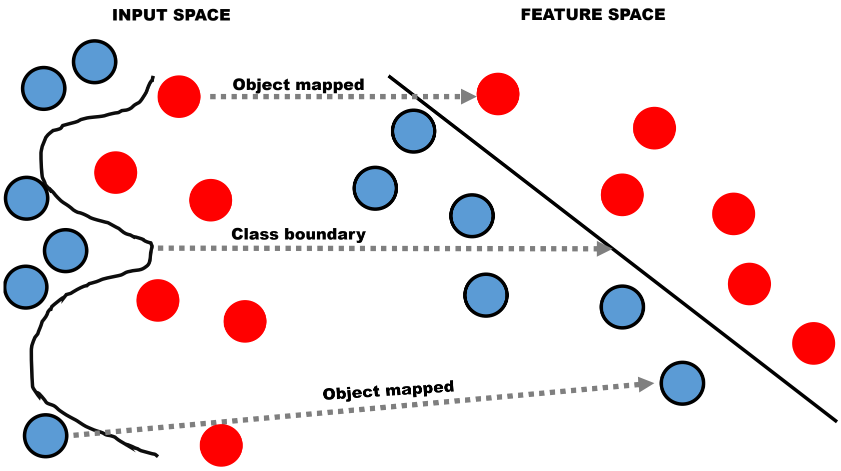


Figure 5.6: Mapping from input to feature space in an SVM

The Essential Essence of the Support Vector Machine

The SVM finds the decision hyperplane leaving the largest possible fraction of points of the same class on the same side, while maximizing the distance of either class from the hyperplane. This minimizes the risk of misclassifying not only the examples in the training data set but also the yet-to-be seen examples of the test set. The general situation is illustrated in Figure 5.7 where it can been see the support vectors are the data points that lie closest to the optimal hyperplane[90]. These are the most difficult points to classify and directly influence the location of the decision boundary.

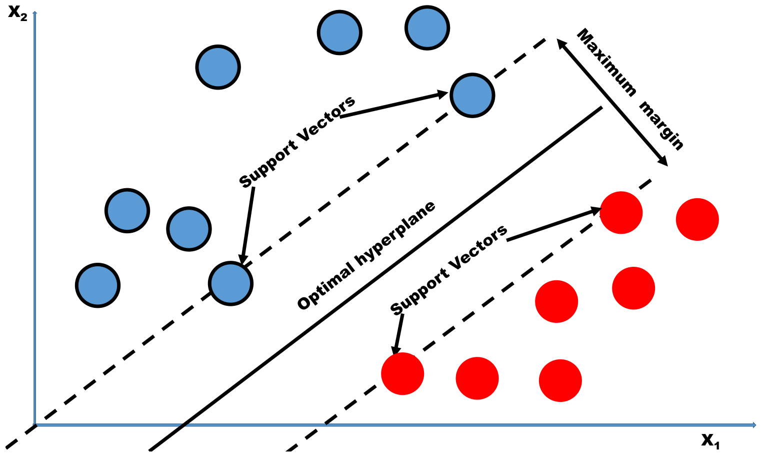


Figure 5.7: Support Vector Machine hyperplane

As we saw in Figure 2.2, in general there are many possible linear decision hyperplanes. To construct an optimal hyperplane, SVM employs an iterative training algorithm to maximize the margin around the separating hyperplane using the "*difficult*" to classify points close to the decision boundary for support.

Let's take a look at how this is achieved. Consider a training sample consisting of N patterns $\{(x_1, y_1, ..., (x_N, y_N)\}$, where x is an n dimensional feature vector, and target $y_i \in \{-1, +1\}$ with corresponding binary labels. The SVM parameters are determined by maximizing the margin hyperplane:

$$\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i \cdot x_j) \tag{5.2}$$

subject to the constraints:

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \; and \; 0 \leq \alpha_i \leq \mathcal{C} \tag{5.3}$$

where $K(x_i, x_j)$ is the kernel function used to map the data from the input space to the feature space[91];$\mathcal{C}$ is a cost/ slack parameter.
There are a large number of kernels. Three you will come across in your own work include:

- Polynomial: $K(x_i, x_j) = (x_i^T x_j + 1)^d$
- Gaussian Radial Basis: $K(x_i, x_j) = \exp\left(-\gamma \left\|x_i - x_j\right\|^2\right), \gamma \geq 0$
- Hyperbolic tangent:$K(x_i, x_j) = \tanh\left(x_i^T x_j + \kappa\right)$

Dealing with Slack

The variable $\mathcal{C}$, known as the slack parameter, serves as the cost parameter that controls the trade-off between the margin and classification error. If no slack is allowed (often known as a hard margin) and the data are linearly separable the support vectors are the points which lie along the supporting hyperplanes. In this case all of the support vectors lie exactly on the margin.

In many situations this will not yield useful results and a soft margin will be required. In this circumstance some proportion of data points are allowed to remain inside the margin.

The slack parameter is used to control this proportion[92]. A soft margin results in a wider margin and greater error on the training data set, however it often improves the generalization of data and reduces the likelihood of over fitting.

NOTE... ✍

The total number of support vectors depends on the amount of allowed slack and the distribution of the data. If a large amount of slack is permitted, there will be a larger number of support vectors than the case where very little slack is permitted. Fewer support vectors means faster classification of test points. This is because the computational complexity of the SVM is linear in the number of support vectors.

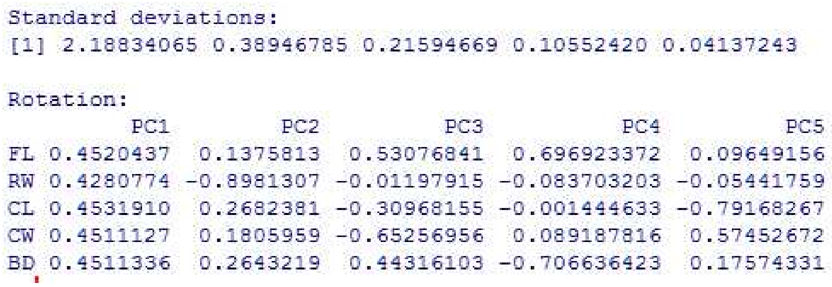
How to Build a Support Vector Machine in R...Right Now

The support vector machine can be built in R with only a few lines of R code. For this example, I will use the `crabs` dataset discussed earlier on page 64. First load the data and calculate the principal components

```
data("crabs",package="MASS")
pca <- prcomp(crabs[,4:8],
center = TRUE,
scale. = TRUE)
```

Let's take a look at the principal components. Here they are:

```
pca
```



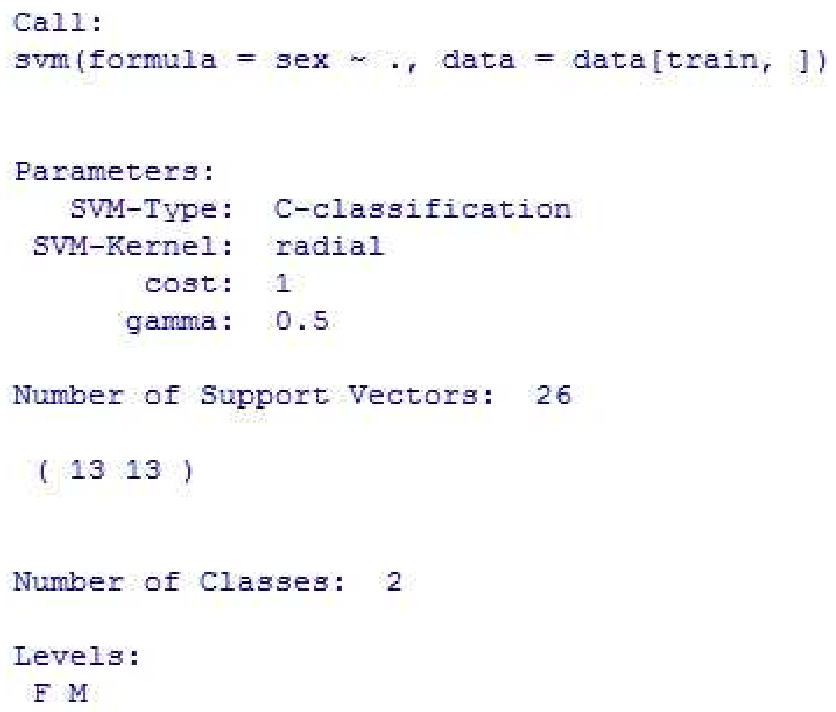
```
Standard deviations:
[1] 2.18834065 0.38946785 0.21594669 0.10552420 0.04137243

Rotation:
         PC1        PC2         PC3          PC4         PC5
FL 0.4520437  0.1375813  0.53076841  0.696923372  0.09649156
RW 0.4280774 -0.8981307 -0.01197915 -0.083703203 -0.05441759
CL 0.4531910  0.2682381 -0.30968155 -0.001444633 -0.79168267
CW 0.4511127  0.1805959 -0.65256956  0.089187816  0.57452672
BD 0.4511336  0.2643219  0.44316103 -0.706636423  0.17574331
```

As we saw previously, the first two components explain over 98% of the variation in the data. The first component represents the sum of the morphological measurements. The second component captures the difference between the rear width and the other measurements. Since these components explain most of the variation in the data I will use them to build the support vector machine.
Now we tidy up the data and create the training and test set:

```
pca1<-pca$x[,1]
pca2<-pca$x[,2]
data<-as.data.frame(crabs$sex)
colnames(data)<-c("sex")
data<-cbind(data,pca1,pca2)
set.seed(2016)
N=nrow(data)
train <- sample(1:N, 100, FALSE)
```

I use the default setting of the support vector machine function `svm` implemented in the package `e1071`:

```
require(e1071)
fit<- svm(sex ~., data = data[train,])
summary(fit)
```



```
Call:
svm(formula = sex ~ ., data = data[train, ])


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.5

Number of Support Vectors:  26

 ( 13 13 )


Number of Classes:  2

Levels:
 F M
```

The fitted model has 26 support vectors, cost parameter = 1 and , gamma = 0.5. Let's see how well this default model performs on the test set:

```
pred <- predict(fit, data[-train,])
table(pred,data$sex[-train])


pred  F  M
   F 49  5
   M  1 45
```

A total of six observations are misclassified. See if you can refine the model to improve the fit.

Notes

[82]See Hosking, Jonathan RM, Edwin PD Pednault, and Madhu Sudan. "A statistical perspective on data mining." Future Generation Computer Systems 13.2 (1997): 117-134.

[83]Add this book to your data science library. Go buy it now! It is a classic. See: Hastie, Trevor, et al. "The elements of statistical learning: data mining, inference and prediction." 2nd Edition.Springer

[84]Burges, Christopher JC. "A tutorial on support vector machines for pattern recognition." Data mining and knowledge discovery 2.2 (1998): 121-167.

[85]Hastie, Trevor, et al. "The elements of statistical learning: data mining, inference and prediction." 2nd Edition. Springer

[86]Corani, Giorgio, and Marino Gatto. "Structural risk minimization: a robust method for density-dependence detection and model selection." Ecography 30.3 (2007): 400-416.

[87]See the paper by Cortes C, Vapnik V (1995) Support-Vector Networks. Machine Learning 20: 273–297.

[88]Note that classification tasks based on drawing separating lines to distinguish between objects of different class memberships are known as hyperplane classifiers. The SVM is well suited for such tasks.

[89]A Hilbert space of finite or infinite dimension.

[90]In two dimensions, the data are separable by a straight line. In many dimensions we need separating hyperplanes.

[91]A kernel,$K(x, y)$, is essentially a continuous function that takes two arguments x and y and maps them to a real value independent of the order of the arguments, i.e.,$K(x, y) = K(y, x) \in \mathcal{R}$.

[92]The nu parameter in nu-SVM.

Chapter 6

Model Selection

"*Just as the ability to devise simple but evocative models is the signature of the great scientist so over-elaboration and over-parameterization is often the mark of mediocrity.*"

George E.P. Box

Choosing the very best model for a particular data science challenge lies at the heart of effective data science. The choice is complicated by the fact that the sample may be small and the selected model will need to perform well on new, as yet unseen observations; in other words, it will need to generalize well. Inadequate ability to generalize lies at the core of the failure of the vast majority of data driven models. Novices are often surprised by the fact that the model which performed so well during the model building process, simply failed to work when deployed in practice.

Fortunately, many of the aliments of overdeveloped models are easy to identify and remedy. This chapter presents several ideas and techniques that will assist you when confronted with this demanding task.

How I Improved My Models in One Evening

In my very first lecture at graduate school, the brilliant mathematics professor who was to share with us the inner working of measure theory, began with the words "*Lex parsimoniae. Entia non sunt multiplicanda praeter necessitatem.*" Which roughly translates to the "*The law of parsimony or economy. Entities should not be multiplied unnecessarily.*" This principal is the famous Ocams Razor; named in honor of the late middle ages scholar William of Ockham.

NOTE... ✍

Ockham is a peaceful English parish on the east side of the Wey Valley, about 20 miles from London. It is perhaps most famous for being the birthplace in 1285 of Franciscan friar William of Ockham. Friar Ockham was an intellectual rebel. He took part forcefully in the great debates of his time on the poverty of the clergy, transubstantiation and the right of the popes to exercise temporal power. He was excommunicated by Pope John XXII in 1328.

Here are several interpretations of Ocams Razor relevant to data science:

- If a smaller set of attributes fits the observations sufficiently well use those attributes. Avoid "*stacking*" additional attributes to improve the fit of a model.
- Select the modeling approach that makes the fewest assumptions.
- Only retain that subset of assumptions which make a clear difference to the predictions of the hypothesis.

- In selecting between hypotheses that explain a phenomenon equal well, it is usually best to start with the simplest one.
- If two or more models have the same prediction accuracy, choose the simplest model.

So now you see how Occam's razor works as a guiding principle to your data science practice. It suggests taking the most parsimonious option in any given situation with the hope that it will be the right choice most of the time.

The appealing thing about Ocams Razor, as a rule of thumb, is that it neatly captures the idea that in building a decision model you should try to find the smallest set of attributes which provide an adequate description of the data. This idea ties nicely into the interpretability of a model. In general, the more parameters and the greater the complexity of a model, the more difficult it is to interpret.

Interpretability, in terms of some underlying theory, can be an intoxicating appeal, especially if like me you were formally trained in disciplines with a heavy bias towards deductive theories. However, ease of interpretability may or may not be important. It depends essentially on the domain of application. For example, a model which purports to explain economic patterns might be designed to be understood through the lens of economic theory[93]. On the other hand, a real time harbor porpoise identification and recognition model has no such constraints. Model parsimony may be a worthwhile goal, but it is one that cannot always be achieved.

A Little Mistake that Cost $5 Million

Data snooping refers to unwittingly assigning meaning to spurious correlations or patterns in a sample. You may have come across the idea in your statistics 101 class when the lecturer presented an example involving the local stork population and number of babies born. Another popular example involves the late actress Elizabeth Taylor's marriages. She remarried in 1951, 1953, 1958, 1960, 1965, 1976, and 1977; all coincided with stock market gains. But would you have comfort in a model that told you to invest when the celebratory of the day gets remarried? Similarly, a model that predicts the number of babies delivered in a local hospital as a consequence of the ebb and flow of the local stork population should be viewed with suspicion[94].

Several times over my career, I have been brought in by a large investment firm to figure out what went wrong with a quantitative investment model. Here is what frequently occurred: The head of the organization, typically a administrative/ MBA type, forms a small team of freshly minted Economics/Finance PhD's to build an automatic investment fund. The goal is to apply the very latest in quantitative techniques to make money.

After several months of intensive research, the head of the team announces they have built the model, tested it extensively on "real data" and are "ready to go". The model is "seeded" and turned on. Within a matter of days (in the worst case) it has lost millions of dollars; Over the course of the next several months even more money is lost. After around 18 months the model is terminated, never recouping the original losses, and the team of "all stars" is disbanded.

What went wrong? In every single case I can recall, the problem pointed straight back to extensive data snooping and ignorance surrounding its impact on model selection, and the ability to accurately predict future observations. The challenge is that in practice data snooping is not as obvious as the textbook examples involving Elizabeth Taylor or Storks and babies.

Recall our goal in learning from data is to select a good predictor $\hat{y}(\theta) = h(x|\hat{\theta})$ from the hypothesis class $\mathcal{H}$. However, in order to identify the best predictor, say $^{*}\hat{y}$, we typically also evaluate p other predictors, $^{1}\hat{y}(\theta), ..., ^{p}\hat{y}(\theta)$ on the same data sample. The investment model development process involved running thousands of simulations over the same dataset with a high number of different parameters/models and reporting only the best result on an out-of-sample test set.

Whenever a "optimal" model is obtained by such an extensive specification search, there is always the danger that the observed good performance may be partially or completely due to chance. This is because the probability that a "*good*" result arises by chance grows with the number of combinations tested - if you sequentially flip a sufficiently large number of coins, a coin that always comes up heads will eventually emerge. Searching for the optimal model in very large model spaces can result in excessive structure in the induced models, sub-optimal model construction, and significantly inflated overestimates of accuracy.

In order to minimize the probability that apparently great performance occurred simply by chance, text books often advise us to divide the sample randomly into a training and test set. The training sample (sometimes called in-sample) contains the sample data sample that will be used to train the various models. The test sample (also known as the out-of-sample set) is used as a way to test the models selected during the training phase. I have had great success by dividing the sample into three sets, with the third set called the validation set. The use of training, test and validation samples dramatically reduce the likelihood that the optimum model suffers from severe data snooping bias[95].

Three Key Lessons of the No Free Lunch Theorem

Roughly stated, the No Free Lunch theorem[96] states that in the lack of prior knowledge (i.e. inductive bias) on average all predictive algorithms that search for the minimum classification error (or extremum over any risk metric) have identical performance according to any measure. A model that works well could also be trained by multiple algorithms for example, linear regression could be trained by gradient descent or least squares.

NOTE... ✍

I once worked for a very wise professor, who for a time, left the academic world in order to seek a fortune in the investment industry. Such moves are rarely totally satisfactory for the academic type; and within a few years the wise professor was back in the university classroom with the horrors of corporate cubical life a receding memory. In building predictive models, the wise professor clearly understood the no free lunch theorem - "*A single model just won't do.*" the wise professor would say. *"We must build an array of models, so that when one fails to work, the others might.*"

Here are three implications:

1. Don't fall in love with a specific class of models. Professor George Box warns "*Statisticians, like artists, have the bad habit of falling in love with their models.*" This is a trait of Data Scientists also. However, falling in love with a model is very dangerous. You must exhibit the qualities of Odysseus in Homer's Odyssey, no matter how beautiful a model is to you, or how much time you have invested in it, do not be seduced by the Sirens song. Always remember that there are numerous ways to achieve the same end result.

2. As my wise professor friend already knew, any learning algorithm has a limited scope of application. No learning algorithm can be guaranteed to succeed on all learnable tasks. Build you data science toolkit to contain a wide

variety of learning algorithms. Then try a variety on your learning problem. If your toolkit is empty or needs a little boost pick up a copy of my book 92 Applied Predictive Modelling Techniques in R.

3. Seek out domain experts with prior knowledge. Learning from data requires an inductive bias (prior knowledge). The No-Free-Lunch principle holds only holds if this is not the case i.e. if one believes in a uniform distribution over optimization problems). Without prior knowledge, any learning algorithm will fail on some learnable task. As Professor David Wolpert himself acknowledges[97] "*...while the NFL* [No Free Lunch]*theorems have strong implications if one believes in a uniform distribution over optimization problems, in no sense should they be interpreted as advocating such a distribution.*" In practice, you will almost always have access to prior knowledge, or can obtain it from domain experts.

Domain expertise is the crucial ingredient, but often missed, by eager data scientists. As twenty-five-year business veteran of analytics Micheal Koukounas explains in his excellent book Real-World Analytics[98]: "*At the time, I was managing a team of data scientists, really smart mathematicians and statisticians. One Monday morning in the middle of the fraud attack, I came into the office to find that one of my team members had spent the weekend building a fraud model to predict which of these transactions had a high probability of being fraudulent...He had constructed a sophisticated neural net model that performed very well...Unfortunately,...we could not use the model...*[It] *worked well in the laboratory environment but was not, to our regret, practical in the real world...In order for it to work, the petroleum company would have had to invest millions of dollars...Moreover, the company would have had to mandate all their gas stations and franchisees to invest thousands of bucks in net point-of-sales and communication equipment...*"

You can avoid this type of potentially costly blunder by seeking the assistance of domain experts as you develop your models.

What is the Bias Variance Trade-off?

The bias variance decomposition is a useful tool for understanding classifier behavior. It turns out the expected misclassification rate can be decomposed into two components, a reducible error and irreducible error:

$$\tilde{R}_{emp}(\theta) = \sum_{x} P(x) \left(\overbrace{\sigma_x^2}^{\text{irreducible error}} + \underbrace{bias_x^2 + variance_x}_{\text{Reducible error}} \right)$$

Irreducible error or inherent uncertainty is associated with the natural variability in the phenomenon under study, and is therefore beyond our control. I like to think of it as the background noise inherent in the system. For example, if you are going to build an age classification model for Emperor Penguin's using weight and height as attributes, you will expect to see some natural variation between the weight and height of all one year old birds[99].

Reducible Error

Reducible error, as the name suggests, can be minimized. It can be decomposed into error due to squared bias and error due to variance. Figure 6.1 illustrates the idea of bias and variance using Kung Fu throwing Stars. The object is to throw these lethal weapons accurately, on target, to hit the bullseye every time. A novice might throw the darts with high variance and high bias (bottom right panel); whilst the Sifu Master, who has studied the art for many years, will hit the target every time (bottom left panel), and therefore has low variance and low bias. The hope is that our model $\hat{y}$ exhibits the characteristics of the Sifu Master.

NOTE... ✍

Data Scientists Ron Kohavi and David Wolpert[100] derived the bias-variance decomposition given in the text for binary classification (where $y \in \{0,1\}$). Assume a misclassification error has a cost 1, and a correct prediction a cost of 0, then the expected misclassification rate,$\tilde{R}_{emp}(\theta)$, is given by:

$$\tilde{R}_{emp}(\theta) = \sum_x P(x) \left(\sigma_x^2 + bias_x^2 + variance_x \right)$$

where:

$$bias_x^2 = \frac{1}{2} \sum_{y \in Y} \left[P(y|x) - P(\hat{y}|x) \right]^2$$

$$variance_x = \frac{1}{2} \left(1 - \sum_{y \in Y} P(\hat{y}|x)^2 \right)$$

$$\sigma_x^2 = \frac{1}{2} \left(1 - \sum_{y \in Y} P(y|x)^2 \right)$$

Bias

Bias relates to the ability of your model $\hat{y}$ to approximate y. It results from model mismatch. It is the squared difference between $P(y|x)$, the true conditional probability of y given x, and the prediction of $\hat{y}$. It measures the amount by which the expected model prediction $\hat{y}$ differs from the true value or target y, over the training sample.

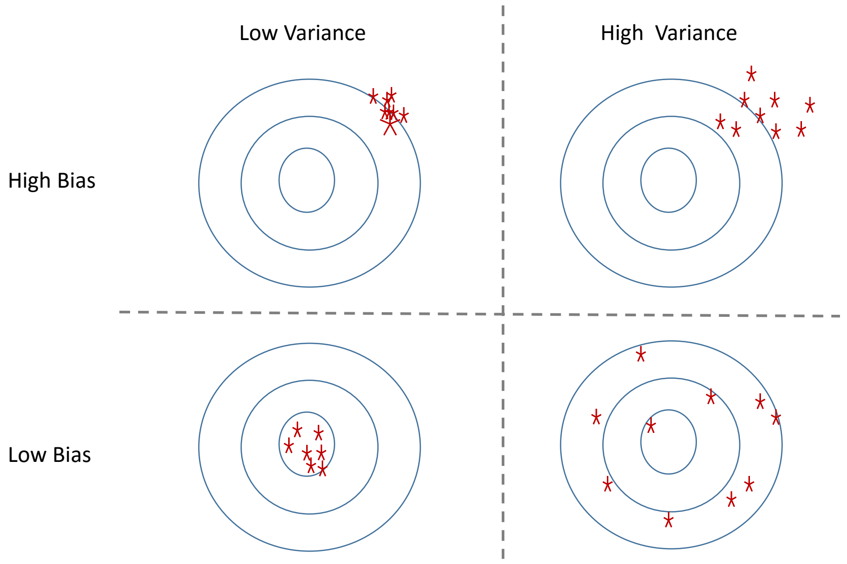


Figure 6.1: Kung Fu throwing star's illustration of bias and variance

Statisticians tend to think of bias as a form of model selection error and this is helpful, because if $\hat{y}$ perfectly represented y, the bias would be zero. Bias is large if $\hat{y}$ produces classifiers that are consistently wrong. For example, if we select a classifier such as Linear discriminant analysis or naive Bayes for the decision boundary of Figure 5.6, the bias can be expected to be high because a large number of points will be consistently misclassified. In this case $\hat{y}$ is said to under-fit the data because the classifier selected can only model a linear hyperplane.

Scholars Christopher Manning, Prabhakar Raghavan and Hinrich Schütze in their excellent book explain[101]: *"We can think of bias as resulting from our domain knowledge (or lack thereof) that we build into the classifier. If we know that the true boundary between the two classes is linear, then a learning method that produces linear classifiers is more likely to succeed than a nonlinear method. But if the true class boundary is not linear and we incorrectly bias the classifier to be linear, then classification accuracy will be low on average."*

The temptation might be to always select a non-linear classifier by default, say for example a multilayered neural network. Notice that such a strategy would violate Ocams Razor (see page 146); and in practice would introduce bias if the true decision plane was linear by overfitting y. In other words, such models would still be subject to model selection error.

Variance

Variance measures the stability of the classification predictions of $\hat{y}$. Whilst bias is a measure of classification accuracy, variance is a measure of classification consistency. Low variance implies high consistency of the classifications decisions made by $\hat{y}$. High variance implies low consistency of the predictions. Linear classification models tend to have low variance because for different training samples from the same underlying probability distribution (or population), they produce similar decision hyperplanes.

An algorithm such as kNN has low bias because it does not assume anything in particular about the distribution of the data points. However, it has high variance, because it will change its prediction in response to the composition of the training sample.

Model building is often a trade-off between bias and variance. Models say $\{^1\hat{y}(\theta), ..., {}^m\hat{y}(\theta)\}$ that exhibit low variance and high bias underfit y; whilst models that exhibit high variance and low bias overfit y. Frequently, if we select an algorithm to reduce bias we will often also increase variance. Therefore, the differences in performance between different learning algorithms can been seen as trade-off between bias and variance. This trade-off helps explain why there is no universal learning method.

NOTE... ✍

If the algorithm has high bias, the following actions might help:

- Add more features
- use an alternative (more sophisticated model).

If the model has high variance try these suggestions:

- use fewer features
- use more training samples.

Do You Make This Mistake with Your Models?

Overfitting is like attending a concert of your favorite band. Depending on the acoustics of the concert venue you will hear both music and noise from the screams of the crowd to reverberations off walls and so on. Overfitting happens when your model perfectly fits both the music and the noise when the intent is to fit the structure (the music). It is generally a result of the predictor being too complex (recall Ocams Razor) so that it fits the underlying structure as well as the noise. The consequence is a small or zero test set classification error. Alas, this super low error rate will fail to materialize on future unseen samples. One consequence of overfitting is poor generalization (prediction) on future data. The general situation is illustrated in Figure 6.2.

NOTE... ✍

Underfitting can also be a problem. It happens when the predictor is too simplistic or rigid to capture the underlying characteristics of the data. In this case the test error will be rather large.

Now here is the reality the textbooks and professors won't tell you about overfitting - It comes with personal career risk. You see, when a new predictive model which does extremely well on the test set is deployed, if it has been overfit it will disappoint not only you, your manager, but those other departments, groups and other parties who are relying on it to succeed. If you are in an organization where predictive modeling is in its infancy, it could lead key decision makers to conclude that either predictive modeling as a whole or you in particular are ineffective.

Use of Ocams Razor, crystal clarity on data snooping, and a clear headed understanding of the no free lunch concept will assist. One other critical tool to help you develop models which generalize well is the use of cross validation.

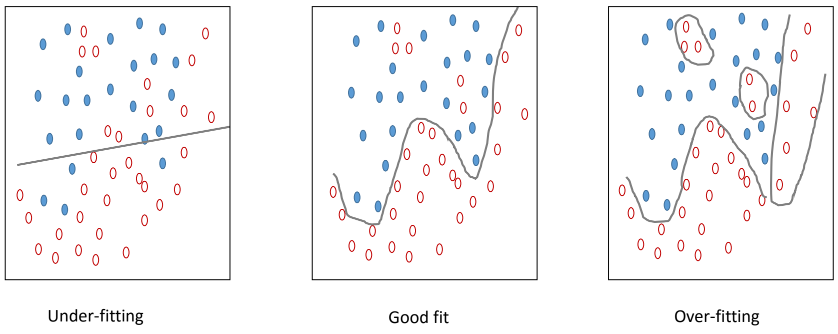


Figure 6.2: The danger of under and over fitting

The Secret of the Hold Out Technique

A handful of generations ago it was common practice to train a learning model and estimate the expected error rate on the very same sample. A researcher would collect data on multiple attributes, then run a classifier and happily observe a very low classification error rate. To the sound of much rejoicing a paper would be published highlighting the predictive power of the newly discovered model. However, researchers in several fields of scholarly activity soon observed that training an algorithm and evaluating its statistical performance on the same data tends to yield optimistic results. Today, validation techniques provide tools to better estimate the expected error rate.

In the hold out method[102] the dataset is split into two non-overlapping groups - a training set used to train the classifier, and a test set used to estimate the error rate of the trained classifier. Professor Stone captures well the idea of a hold out sample when he states[103] "*An example of controlled division* [hold out validation] *is provided by the cautious statistician who sets aside a randomly selected part of his sample without looking at it and then plays without inhibition on what is left, confident in the knowledge that the set-aside data will deliver an unbiased judgment on the efficacy of his analysis.*" Figure 6.3 illustrates the situation.

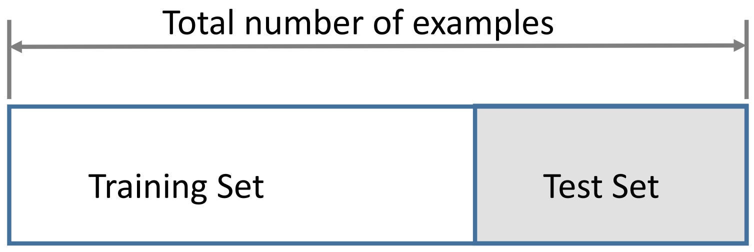


Figure 6.3: The Hold Out Method

If the test set is a representative sample from $P(x, y)$ the error will be unbiased asymptotically, so that:

$$\lim N \to \infty \, R_{test}(\theta) = R(\theta),$$

where $R(\theta)$ is the generalization error. How close the test error is to the generalization error critically depends on the sample size.

NOTE... ✍

S.C Larson[104], back in 1931, used a hold out method by dividing a sample from the Mississippi Survey into a training and test set to provide more accurate error estimates. It would be several decades before this became standard practice.

How should you divide the data between the training set and the test set for hold out validation? Since hold out validation involves a single train and test experiment, the estimate of the error rate may be dependent on how the data is split. A sub-optimal split might result in a misleading error rate. The observations selected for inclusion in the test set might be disproportionally too easy or too difficult to classify; Furthermore, in limited data situations, the test set data may be necessary to train the classifier. The limitations of holdout validation led to the development of resampling methods such as k-fold cross validation.

The Art of Effective Cross Validation

In the textbook situation you have sufficient data to train and validate your models using a training set; and have ample data for assessing the quality of the model via a test set. Implicit in this scenario is a large and diverse sample from which accurate estimates of model parameters and error rates can be obtained.

However, in practice you will frequently face data poor situations, where scarcity of samples rather than abundance is the rule, and it will not be possible to set aside a portion of the dataset purely for testing. Cross validation techniques are useful here because they seek to extract the most information possible about the expected misclassification error.

k-fold Cross Validation

The idea of k-fold cross validation was introduced by the expert witness and founder of the School of Statistics at the University of Minnesota, Professor Seymour Geisser [105]. In k-fold cross-validation the training sample is partitioned into k equally sized segments. For each of the k segments, $k-1$ folds are used for training and the remaining one fold used for testing. In this way a total of k experiments can be performed and all the examples in the dataset are eventually used for both training and testing. Figure 6.4 demonstrates an example with $k = 5$. The darker sections indicate the test sets, while the lighter sections represent data used for training.

The estimate of the classification error rate is obtained as the average of the separate k estimates:

$$\tilde{R}^{*}_{test}(\theta) = \frac{1}{k}\sum_{i=1}^{k}\tilde{R}_{test,i}(\theta)$$

where $\tilde{R}_{test,i}(\theta)$ is the classification error on the i$^{\text{th}}$ test set.

Under the assumption that training and test data are both generated from the same distribution, k-fold cross validation for functions trained on $N\left(1-\frac{N}{K}\right)$examples provides an unbiased estimate of the classification error.

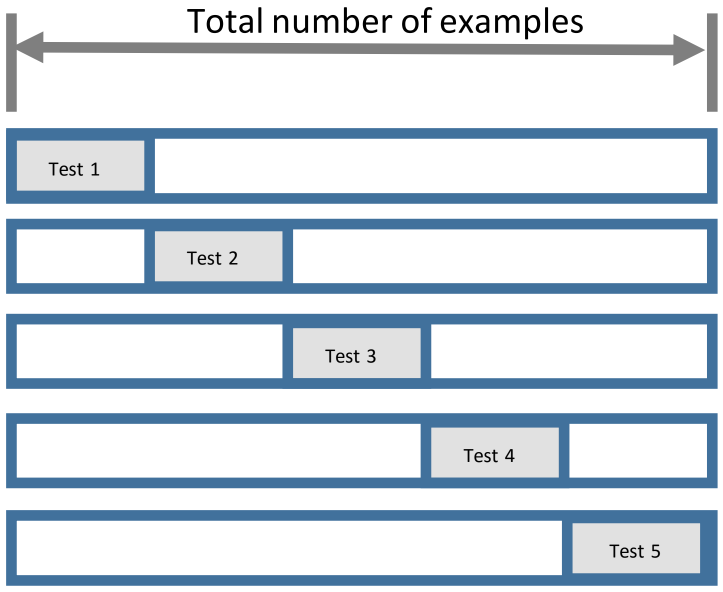


Figure 6.4: Five fold cross validation

In practice, the choice of the number of folds depends on the size of the dataset. For very sparse datasets, we may have to use leave-one-out in order to train the classifier on as many examples as possible. In general, the larger the number of folds the smaller the bias of the true error rate estimator will be. Unfortunately, the variance of the error rate estimator tends to grow as the number of folds increases. In actual practice 5 or 10-fold cross-validation are most commonly specified.

NOTE... ✍

How to split the training and validation sets? I have had some success with training and validation sets of the same size. This might a good place for you to start in your next project.

An Example in R

I illustrate the idea of k-fold cross validation using the k nearest neighbors algorithm (see page 41) with the `PimaIndiansDiabetes2` (see page 45) dataset. First load and prepare the data:

```
data("PimaIndiansDiabetes2",package="
   mlbench")
temp<-(PimaIndiansDiabetes2)
temp$insulin  <- NULL
temp$triceps <- NULL
temp<-na.omit(temp)
class<-temp[,7]
data<-temp[,-7]
data<-scale(data)
set.seed(2016)
n=nrow(temp)
train <- sample(1:n, 600, FALSE)
```

The above R code will be familiar to you, notice that the `class` object contains the classes and the observations are scaled using the `scale` method.

R contains a wide range of packages that can perform cross validation. I use the `chemometrics` package. It contains the `knnEval` method which allows for cross validation. We are interested in evaluating a range of values of `k`, say from 1 to 30 with a 10 fold cross validation. Here is how to do that:

```
library(chemometrics)
fit <- knnEval(data,
class,
train,
kfold = 10,
knnvec=seq(1,30),
legpos="bottomright")
```

The parameter `knnvec` holds the sequence of `k` values we wish to try, the `legpos` parameter controls the location of the legend.

Once the code has been executed you should see a diagram similar to Figure 6.5. For each value of `k`, the training, test and cross validation error mean are plotted. The dashed horizontal line corresponds to one standard error above the minimum cross validation error mean. This rule is used to choose the most parsimonious model whose error is no more than one standard error above the error of the best model. From the chart we see the best model has `k = 22,` and the most parsimonious model using the one standard deviation rule has `k = 13`.

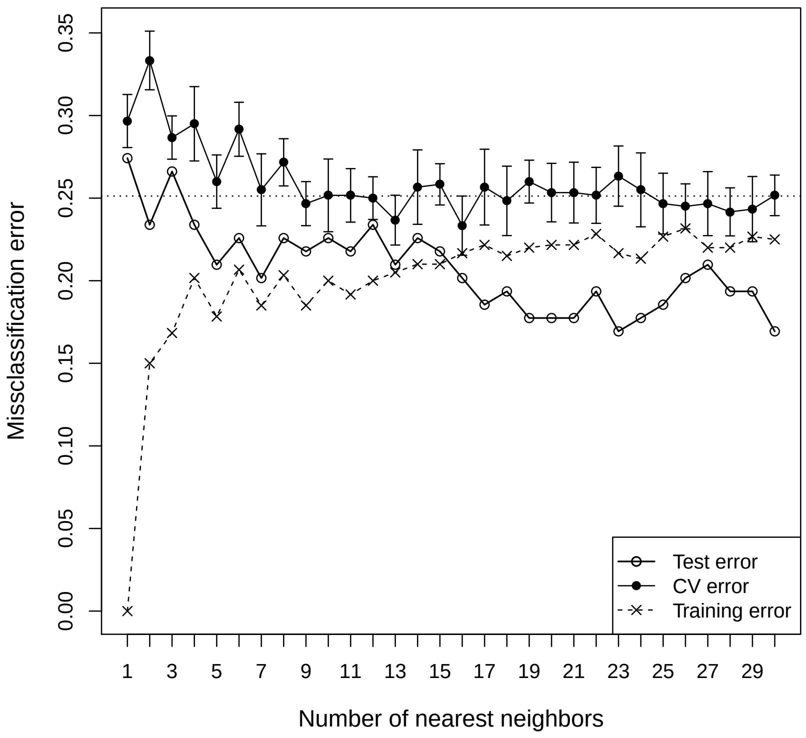


Figure 6.5: Output of `knnEval`.

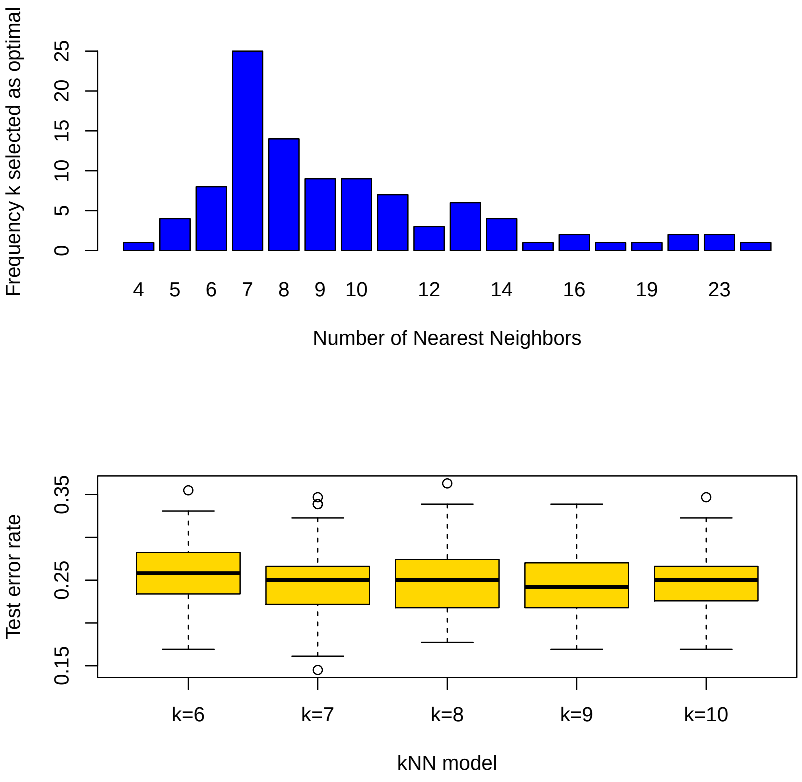


Figure 6.6: Optimal `k` (top panel) and boxplot of test error for the top five models

In practice the optimal `k` will depend on the randomly selected observations included in the training set. Figure 6.6 (top panel) shows the optimal value of k obtained by randomly choosing the training set 100 times and then estimating 10 fold cross validation. This approach is known as repeated k-fold cross validation. Notice that `k = 7` is the most frequently occurring optimal value. The bottom panel of Figure 6.6 shows boxplots of the top five models. Given the closeness of the test error between these models, Ocams rule would suggest choosing `k = 6` for evaluation on the test sample.

NOTE... ✍

Repeated k-fold cross validation runs k-fold cross-validation multiple times, using different partitions of the sample. The data is reshuffled before each k-fold round and the results averaged. It helps to avoid the dependence of the predictions on the data split.

Leave One Out Validation

Back in the early 1970's Professor Mervyn Stone of University College London suggested the use of leave-one-out cross-validation for estimating model parameters and for assessing their predictive error. In this case `k` is set equal to the number of examples in the training set. At each iteration nearly all the data except for a single observation are used for training. The model is then tested on that single observation - one or zero for success of failure, respectively. The results of the error estimates are then averaged to estimate the overall classification error.

Although leave one out cross validation typically results in a fairly accurate estimate (almost unbiased) of the error, it tends to have high variance. However, it is a particularly useful tool when the sample data are scarce, for example in bioinformatics where only a few dozen data samples might be available. Of course, given that the learning algorithm must be estimated for every data point, it comes at a larger computational cost than 5 or 10 fold cross validation.

Let's try out the idea in R. I will use the `crabs` data set we saw earlier from the `MASS` package. First load the packages and data:

```
library("MASS")
```

```
data(crabs)
```

As we did earlier, first calculate the first and second principal components for use as the attributes:

```
pca <- prcomp(crabs[,4:8],
center = TRUE,
scale. = TRUE)
pca1<-pca$x[,1]
pca2<-pca$x[,2]
data<-as.data.frame(crabs$sex)
colnames(data)<-c("sex")
data<-cbind(data,pca1,pca2)
```

The R object data contains the class labels and the first and second principal components.

Next create the training and test samples:

```
set.seed(2016)
n=nrow(crabs)
train_size=150
train <- sample(1:n, train_size, FALSE)

train_sample<-data[train,2:3]
class_train<-data$sex[train]

test_sample<-data[-train,2:3]
class_test<-data$sex[-train]
```

Let's use the naive Bayes model from the klaR package and the caret package to perform the leave one out cross validation:

```
library("klaR")
library("caret")
```

The caret package is one to keep near the top of your R toolkit. It is designed to aid data mining and has tons of awesome features. Cross validation is carried out as follows:

```
fit<- train(train_sample,
class_train,"nb",
trControl=trainControl(method="cv",
number=train_size))
```

The parameter `number` refers to the number of folds, I set it equal to the number of observations in the training set to perform leave one out cross validation.

Now predict using the test set and calculate the confusion matrix:

```
pred<-predict(fit$finalModel,
test_sample)$class
table(pred,class_test)
   class_test
pred  F  M
   F 28  0
   M  2 20
```

NOTE... ✍

Here is a quick rule of thumb useful for all resampling methods. Split your sample into three parts:

- the training set to build the classifier;
- the validation set to fine tune model parameters and pick an algorithm;
- the test set to estimate the future error rate. Remember to run the model through the test set once only or else report the results on every run.

Notes

[93]Although this is not really necessary.

[94]Have some fun and read Höfer, Thomas, and Hildegard Przyrembel. "New evidence for the theory of the stork." Paediatric and perinatal epidemiology 18.1 (2004): 88-92.

[95]See Esbensen, Kim H., and Paul Geladi. "Principles of Proper Validation: use and abuse of resampling for validation." Journal of Chemometrics 24.3:4 (2010): 168-187.

[96]Wolpert, David H., and William G. Macready. "No free lunch theorems for optimization." Evolutionary Computation, IEEE Transactions on 1.1 (1997): 67-82.

[97]Wolpert, David H. "What the no free lunch theorems really mean; how to improve search algorithms." Santa fe Institute Working Paper. 2012. 12.

[98]Koukounas,Michael.(2014). Real-World Analytics. Full Court Press. For more details see http://www.real-worldanalytics.com/

[99]Of course overall, you might expect three year old birds to be taller and heaver than one year old birds.

[100]Kohavi, Ron, and David H. Wolpert. "Bias plus variance decomposition for zero-one loss functions." ICML. Vol. 96. 1996.

[101]Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. Vol. 1. No. 1. Cambridge: Cambridge university press, 2008.

[102]Take a look at Devroye, Luc P., and Terry J. Wagner. "Distribution-free performance bounds for potential function rules." Information Theory, IEEE Transactions on 25.5 (1979): 601-604.

[103]See Stone, Mervyn. "Cross-validatory choice and assessment of statistical predictions." Journal of the royal statistical society. Series B (Methodological) (1974): 111-147.

[104]Larson, Selmer C. "The shrinkage of the coefficient of multiple correlation." Journal of Educational Psychology 22.1 (1931): 45.

[105]Geisser, Seymour. "The predictive sample reuse method with applications." Journal of the American Statistical Association 70.350 (1975): 320-328.

Congratulations!

You made it to the end. Here are three things you can do next.

1. Pick up your **FREE** copy of **12 Resources to Supercharge Your Productivity in R** at *http://www.auscov.com*

2. Gift a copy of this book to your friends, co-workers, teammates or your entire organization.

3. If you found this book useful and have a moment to spare, I would really appreciate a short review. Your help in spreading the word is gratefully received.

I've spoken to thousands of people over the past few years. I'd love to hear your experiences using the ideas in this book. Contact me with your stories, questions and suggestions at *Info@NigelDLewis.com*.

Good luck!

Dr. Nigel D. Lewis

P.S. Thanks for allowing me to partner with you on your data science journey.

Index

OTHER BOOKS YOU WILL ALSO ENJOY

Write your notes here:

Write your notes here:

Write your notes here: