Notations

x	Scalar value
$\boldsymbol{x}$	Vector
$\mathbf{X}$	Matrix
$\boldsymbol{x}^T$	Transpose
$\mathbf{X}^{-1}$	Inverse
X	Random variable
$P(X)$	Probability mass function when X is discrete
$p(X)$	Probability density function when X is continuous
$P(X\|Y)$	Conditional probability of X given Y
$E[X]$	Expected value of the random variable X
$\text{Var}(X)$	Variance of X
$\text{Cov}(X, Y)$	Covariance of X and Y
$\text{Corr}(X, Y)$	Correlation of X and Y
μ	Mean
σ^2	Variance
$\boldsymbol{\Sigma}$	Covariance matrix
m	Estimator to the mean
s^2	Estimator to the variance
$\mathbf{S}$	Estimator to the covariance matrix

$\mathcal{N}(\mu, \sigma^2)$	Univariate normal distribution with mean μ and variance σ^2
$\mathcal{Z}$	Unit normal distribution: $\mathcal{N}(0, 1)$
$\mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	d-variate normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
x	Input
d	Number of inputs (input dimensionality)
y	Output
r	Required output
K	Number of outputs (classes)
N	Number of training instances
z	Hidden value, intrinsic dimension, latent factor
k	Number of hidden dimensions, latent factors
C_i	Class i
$\mathcal{X}$	Training sample
$\{x^t\}_{t=1}^N$	Set of x with index t ranging from 1 to N
$\{x^t, r^t\}_t$	Set of ordered pairs of input and desired output with index t
$g(x\|\theta)$	Function of x defined up to a set of parameters θ
$\arg\max_\theta g(x\|\theta)$	The argument θ for which g has its maximum value
$\arg\min_\theta g(x\|\theta)$	The argument θ for which g has its minimum value
$E(\theta\|\mathcal{X})$	Error function with parameters θ on the sample $\mathcal{X}$
$l(\theta\|\mathcal{X})$	Likelihood of parameters θ on the sample $\mathcal{X}$
$\mathcal{L}(\theta\|\mathcal{X})$	Log likelihood of parameters θ on the sample $\mathcal{X}$
$1(c)$	1 if c is true, 0 otherwise
$\#\{c\}$	Number of elements for which c is true
δ_{ij}	Kronecker delta: 1 if $i = j$, 0 otherwise

1 *Introduction*

1.1 What Is Machine Learning?

TO SOLVE a problem on a computer, we need an algorithm. An algorithm is a sequence of instructions that should be carried out to transform the input to output. For example, one can devise an algorithm for sorting. The input is a set of numbers and the output is their ordered list. For the same task, there may be various algorithms and we may be interested in finding the most efficient one, requiring the least number of instructions or memory or both.

For some tasks, however, we do not have an algorithm—for example, to tell spam emails from legitimate emails. We know what the input is: an email document that in the simplest case is a file of characters. We know what the output should be: a yes/no output indicating whether the message is spam or not. We do not know how to transform the input to the output. What can be considered spam changes in time and from individual to individual.

What we lack in knowledge, we make up for in data. We can easily compile thousands of example messages some of which we know to be spam and what we want is to "learn" what consititutes spam from them. In other words, we would like the computer (machine) to extract automatically the algorithm for this task. There is no need to learn to sort numbers, we already have algorithms for that; but there are many applications for which we do not have an algorithm but do have example data.

With advances in computer technology, we currently have the ability to store and process large amounts of data, as well as to access it from physically distant locations over a computer network. Most data acquisition

devices are digital now and record reliable data. Think, for example, of a supermarket chain that has hundreds of stores all over a country selling thousands of goods to millions of customers. The point of sale terminals record the details of each transaction: date, customer identification code, goods bought and their amount, total money spent, and so forth. This typically amounts to gigabytes of data every day. What the supermarket chain wants is to be able to predict who are the likely customers for a product. Again, the algorithm for this is not evident; it changes in time and by geographic location. The stored data becomes useful only when it is analyzed and turned into information that we can make use of, for example, to make predictions.

We do not know exactly which people are likely to buy this ice cream flavor, or the next book of this author, or see this new movie, or visit this city, or click this link. If we knew, we would not need any analysis of the data; we would just go ahead and write down the code. But because we do not, we can only collect data and hope to extract the answers to these and similar questions from data.

We do believe that there is a process that explains the data we observe. Though we do not know the details of the process underlying the generation of data—for example, consumer behavior—we know that it is not completely random. People do not go to supermarkets and buy things at random. When they buy beer, they buy chips; they buy ice cream in summer and spices for Glühwein in winter. There are certain patterns in the data.

We may not be able to identify the process completely, but we believe we can construct *a good and useful approximation*. That approximation may not explain everything, but may still be able to account for some part of the data. We believe that though identifying the complete process may not be possible, we can still detect certain patterns or regularities. This is the niche of machine learning. Such patterns may help us understand the process, or we can use those patterns to make predictions: Assuming that the future, at least the near future, will not be much different from the past when the sample data was collected, the future predictions can also be expected to be right.

Application of machine learning methods to large databases is called *data mining*. The analogy is that a large volume of earth and raw material is extracted from a mine, which when processed leads to a small amount of very precious material; similarly, in data mining, a large volume of data is processed to construct a simple model with valuable use,

for example, having high predictive accuracy. Its application areas are abundant: In addition to retail, in finance banks analyze their past data to build models to use in credit applications, fraud detection, and the stock market. In manufacturing, learning models are used for optimization, control, and troubleshooting. In medicine, learning programs are used for medical diagnosis. In telecommunications, call patterns are analyzed for network optimization and maximizing the quality of service. In science, large amounts of data in physics, astronomy, and biology can only be analyzed fast enough by computers. The World Wide Web is huge; it is constantly growing, and searching for relevant information cannot be done manually.

But machine learning is not just a database problem; it is also a part of artificial intelligence. To be intelligent, a system that is in a changing environment should have the ability to learn. If the system can learn and adapt to such changes, the system designer need not foresee and provide solutions for all possible situations.

Machine learning also helps us find solutions to many problems in vision, speech recognition, and robotics. Let us take the example of recognizing faces: This is a task we do effortlessly; every day we recognize family members and friends by looking at their faces or from their photographs, despite differences in pose, lighting, hair style, and so forth. But we do it unconsciously and are unable to explain how we do it. Because we are not able to explain our expertise, we cannot write the computer program. At the same time, we know that a face image is not just a random collection of pixels; a face has structure. It is symmetric. There are the eyes, the nose, the mouth, located in certain places on the face. Each person's face is a pattern composed of a particular combination of these. By analyzing sample face images of a person, a learning program captures the pattern specific to that person and then recognizes by checking for this pattern in a given image. This is one example of *pattern recognition*.

Machine learning is programming computers to optimize a performance criterion using example data or past experience. We have a model defined up to some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data or past experience. The model may be *predictive* to make predictions in the future, or *descriptive* to gain knowledge from data, or both.

Machine learning uses the theory of statistics in building mathematical models, because the core task is making inference from a sample. The

role of computer science is twofold: First, in training, we need efficient algorithms to solve the optimization problem, as well as to store and process the massive amount of data we generally have. Second, once a model is learned, its representation and algorithmic solution for inference needs to be efficient as well. In certain applications, the efficiency of the learning or inference algorithm, namely, its space and time complexity, may be as important as its predictive accuracy.

Let us now discuss some example applications in more detail to gain more insight into the types and uses of machine learning.

1.2 Examples of Machine Learning Applications

1.2.1 Learning Associations

In the case of retail—for example, a supermarket chain—one application of machine learning is *basket analysis*, which is finding associations between products bought by customers: If people who buy X typically also buy Y, and if there is a customer who buys X and does not buy Y, he or she is a potential Y customer. Once we find such customers, we can target them for cross-selling.

ASSOCIATION RULE

In finding an *association rule*, we are interested in learning a conditional probability of the form $P(Y|X)$ where Y is the product we would like to condition on X, which is the product or the set of products which we know that the customer has already purchased.

Let us say, going over our data, we calculate that $P(\text{chips}|\text{beer}) = 0.7$. Then, we can define the rule:

70 percent of customers who buy beer also buy chips.

We may want to make a distinction among customers and toward this, estimate $P(Y|X, D)$ where D is the set of customer attributes, for example, gender, age, marital status, and so on, assuming that we have access to this information. If this is a bookseller instead of a supermarket, products can be books or authors. In the case of a Web portal, items correspond to links to Web pages, and we can estimate the links a user is likely to click and use this information to download such pages in advance for faster access.

1.2.2 Classification

A credit is an amount of money loaned by a financial institution, for example, a bank, to be paid back with interest, generally in installments. It is important for the bank to be able to predict in advance the risk associated with a loan, which is the probability that the customer will default and not pay the whole amount back. This is both to make sure that the bank will make a profit and also to not inconvenience a customer with a loan over his or her financial capacity.

In *credit scoring* (Hand 1998), the bank calculates the risk given the amount of credit and the information about the customer. The information about the customer includes data we have access to and is relevant in calculating his or her financial capacity—namely, income, savings, collaterals, profession, age, past financial history, and so forth. The bank has a record of past loans containing such customer data and whether the loan was paid back or not. From this data of particular applications, the aim is to infer a general rule coding the association between a customer's attributes and his risk. That is, the machine learning system fits a model to the past data to be able to calculate the risk for a new application and then decides to accept or refuse it accordingly.

CLASSIFICATION

This is an example of a *classification* problem where there are two classes: low-risk and high-risk customers. The information about a customer makes up the *input* to the classifier whose task is to assign the input to one of the two classes.

After training with the past data, a classification rule learned may be of the form

IF income$> \theta_1$ AND savings$> \theta_2$ THEN low-risk ELSE high-risk

for suitable values of θ_1 and θ_2 (see figure 1.1). This is an example of a *discriminant*; it is a function that separates the examples of different classes.

DISCRIMINANT

PREDICTION

Having a rule like this, the main application is *prediction*: Once we have a rule that fits the past data, if the future is similar to the past, then we can make correct predictions for novel instances. Given a new application with a certain income and savings, we can easily decide whether it is low-risk or high-risk.

In some cases, instead of making a 0/1 (low-risk/high-risk) type decision, we may want to calculate a probability, namely, $P(Y|X)$, where X are the customer attributes and Y is 0 or 1 respectively for low-risk

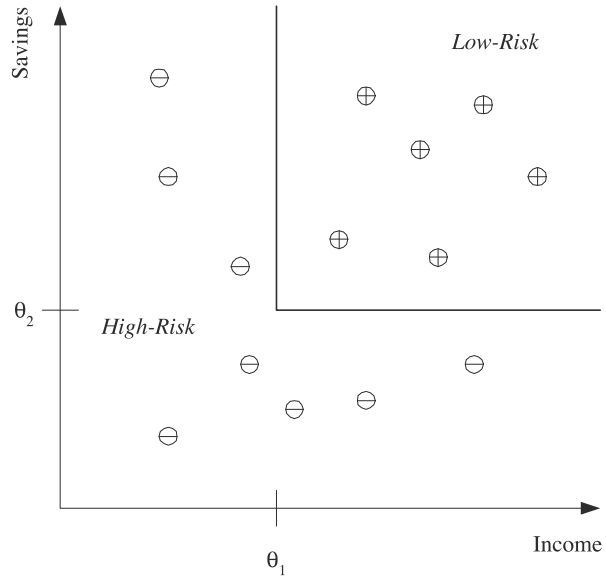

Figure 1.1 Example of a training dataset where each circle corresponds to one data instance with input values in the corresponding axes and its sign indicates the class. For simplicity, only two customer attributes, income and savings, are taken as input and the two classes are low-risk ('+') and high-risk ('−'). An example discriminant that separates the two types of examples is also shown.

and high-risk. From this perspective, we can see classification as learning an association from X to Y. Then for a given $X = x$, if we have $P(Y = 1|X = x) = 0.8$, we say that the customer has an 80 percent probability of being high-risk, or equivalently a 20 percent probability of being low-risk. We then decide whether to accept or refuse the loan depending on the possible gain and loss.

PATTERN RECOGNITION

There are many applications of machine learning in *pattern recognition*. One is *optical character recognition*, which is recognizing character codes from their images. This is an example where there are multiple classes, as many as there are characters we would like to recognize. Especially interesting is the case when the characters are handwritten—for example, to read zip codes on envelopes or amounts on checks. People have different handwriting styles; characters may be written small or large, slanted, with a pen or pencil, and there are many possible images corresponding

to the same character. Though writing is a human invention, we do not have any system that is as accurate as a human reader. We do not have a formal description of 'A' that covers all 'A's and none of the non-'A's. Not having it, we take samples from writers and learn a definition of A-ness from these examples. But though we do not know what it is that makes an image an 'A', we are certain that all those distinct 'A's have something in common, which is what we want to extract from the examples. We know that a character image is not just a collection of random dots; it is a collection of strokes and has a regularity that we can capture by a learning program.

If we are reading a text, one factor we can make use of is the redundancy in human languages. A word is a *sequence* of characters and successive characters are not independent but are constrained by the words of the language. This has the advantage that even if we cannot recognize a character, we can still read t?e word. Such contextual dependencies may also occur in higher levels, between words and sentences, through the syntax and semantics of the language. There are machine learning algorithms to learn sequences and model such dependencies.

In the case of *face recognition*, the input is an image, the classes are people to be recognized, and the learning program should learn to associate the face images to identities. This problem is more difficult than optical character recognition because there are more classes, input image is larger, and a face is three-dimensional and differences in pose and lighting cause significant changes in the image. There may also be occlusion of certain inputs; for example, glasses may hide the eyes and eyebrows, and a beard may hide the chin.

In *medical diagnosis*, the inputs are the relevant information we have about the patient and the classes are the illnesses. The inputs contain the patient's age, gender, past medical history, and current symptoms. Some tests may not have been applied to the patient, and thus these inputs would be missing. Tests take time, may be costly, and may inconvience the patient so we do not want to apply them unless we believe that they will give us valuable information. In the case of a medical diagnosis, a wrong decision may lead to a wrong or no treatment, and in cases of doubt it is preferable that the classifier reject and defer decision to a human expert.

In *speech recognition*, the input is acoustic and the classes are words that can be uttered. This time the association to be learned is from an acoustic signal to a word of some language. Different people, because

of differences in age, gender, or accent, pronounce the same word differently, which makes this task rather difficult. Another difference of speech is that the input is *temporal*; words are uttered in time as a sequence of speech phonemes and some words are longer than others.

Acoustic information only helps up to a certain point, and as in optical character recognition, the integration of a "language model" is critical in speech recognition, and the best way to come up with a language model is again by learning it from some large corpus of example data. The applications of machine learning to *natural language processing* is constantly increasing. Spam filtering is one where spam generators on one side and filters on the other side keep finding more and more ingenious ways to outdo each other. Perhaps the most impressive would be *machine translation*. After decades of research on hand-coded translation rules, it has become apparent recently that the most promising way is to provide a very large number of example pairs of translated texts and have a program figure out automatically the rules to map one string of characters to another.

Biometrics is recognition or authentication of people using their physiological and/or behavioral characteristics that requires an integration of inputs from different modalities. Examples of physiological characteristics are images of the face, fingerprint, iris, and palm; examples of behavioral characteristics are dynamics of signature, voice, gait, and key stroke. As opposed to the usual identification procedures—photo, printed signature, or password—when there are many different (uncorrelated) inputs, forgeries (spoofing) would be more difficult and the system would be more accurate, hopefully without too much inconvenience to the users. Machine learning is used both in the separate recognizers for these different modalities and in the combination of their decisions to get an overall accept/reject decision, taking into account how reliable these different sources are.

KNOWLEDGE EXTRACTION

Learning a rule from data also allows *knowledge extraction*. The rule is a simple model that explains the data, and looking at this model we have an explanation about the process underlying the data. For example, once we learn the discriminant separating low-risk and high-risk customers, we have the knowledge of the properties of low-risk customers. We can then use this information to target potential low-risk customers more efficiently, for example, through advertising.

COMPRESSION

Learning also performs *compression* in that by fitting a rule to the data, we get an explanation that is simpler than the data, requiring less mem-

ory to store and less computation to process. Once you have the rules of addition, you do not need to remember the sum of every possible pair of numbers.

OUTLIER DETECTION

Another use of machine learning is *outlier detection*, which is finding the instances that do not obey the rule and are exceptions. In this case, after learning the rule, we are not interested in the rule but the exceptions not covered by the rule, which may imply anomalies requiring attention—for example, fraud.

1.2.3 Regression

Let us say we want to have a system that can predict the price of a used car. Inputs are the car attributes—brand, year, engine capacity, mileage, and other information—that we believe affect a car's worth. The output is the price of the car. Such problems where the output is a number are *regression* problems.

REGRESSION

Let X denote the car attributes and Y be the price of the car. Again surveying the past transactions, we can collect a training data and the machine learning program fits a function to this data to learn Y as a function of X. An example is given in figure 1.2 where the fitted function is of the form

$$y = wx + w_0$$

for suitable values of w and w_0.

SUPERVISED LEARNING

Both regression and classification are *supervised learning* problems where there is an input, X, an output, Y, and the task is to learn the mapping from the input to the output. The approach in machine learning is that we assume a model defined up to a set of parameters:

$$y = g(x|\theta)$$

where $g(\cdot)$ is the model and θ are its parameters. Y is a number in regression and is a class code (e.g., 0/1) in the case of classification. $g(\cdot)$ is the regression function or in classification, it is the discriminant function separating the instances of different classes. The machine learning program optimizes the parameters, θ, such that the approximation error is minimized, that is, our estimates are as close as possible to the correct values given in the training set. For example in figure 1.2, the model is linear and w and w_0 are the parameters optimized for best fit to the

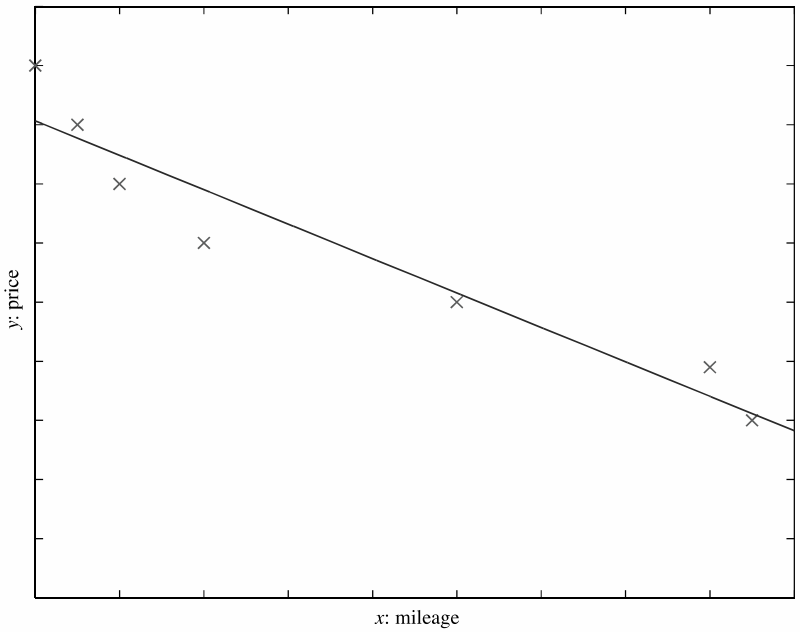

Figure 1.2 A training dataset of used cars and the function fitted. For simplicity, mileage is taken as the only input attribute and a linear model is used.

training data. In cases where the linear model is too restrictive, one can use for example a quadratic

$$y = w_2x^2 + w_1x + w_0$$

or a higher-order polynomial, or any other nonlinear function of the input, this time optimizing its parameters for best fit.

Another example of regression is navigation of a mobile robot, for example, an autonomous car, where the output is the angle by which the steering wheel should be turned at each time, to advance without hitting obstacles and deviating from the route. Inputs in such a case are provided by sensors on the car—for example, a video camera, GPS, and so forth. Training data can be collected by monitoring and recording the actions of a human driver.

One can envisage other applications of regression where one is trying

to optimize a function[1]. Let us say we want to build a machine that roasts coffee. The machine has many inputs that affect the quality: various settings of temperatures, times, coffee bean type, and so forth. We make a number of experiments and for different settings of these inputs, we measure the quality of the coffee, for example, as consumer satisfaction. To find the optimal setting, we fit a regression model linking these inputs to coffee quality and choose new points to sample near the optimum of the current model to look for a better configuration. We sample these points, check quality, and add these to the data and fit a new model. This is generally called *response surface design*.

1.2.4 Unsupervised Learning

In supervised learning, the aim is to learn a mapping from the input to an output whose correct values are provided by a supervisor. In unsupervised learning, there is no such supervisor and we only have input data. The aim is to find the regularities in the input. There is a structure to the input space such that certain patterns occur more often than others, and we want to see what generally happens and what does not. In statistics, this is called *density estimation*.

DENSITY ESTIMATION

CLUSTERING

One method for density estimation is *clustering* where the aim is to find clusters or groupings of input. In the case of a company with a data of past customers, the customer data contains the demographic information as well as the past transactions with the company, and the company may want to see the distribution of the profile of its customers, to see what type of customers frequently occur. In such a case, a clustering model allocates customers similar in their attributes to the same group, providing the company with natural groupings of its customers; this is called *customer segmentation*. Once such groups are found, the company may decide strategies, for example, services and products, specific to different groups; this is known as *customer relationship management*. Such a grouping also allows identifying those who are outliers, namely, those who are different from other customers, which may imply a niche in the market that can be further exploited by the company.

An interesting application of clustering is in *image compression*. In this case, the input instances are image pixels represented as RGB values. A clustering program groups pixels with similar colors in the same

1. I would like to thank Michael Jordan for this example.

group, and such groups correspond to the colors occurring frequently in the image. If in an image, there are only shades of a small number of colors, and if we code those belonging to the same group with one color, for example, their average, then the image is quantized. Let us say the pixels are 24 bits to represent 16 million colors, but if there are shades of only 64 main colors, for each pixel we need 6 bits instead of 24. For example, if the scene has various shades of blue in different parts of the image, and if we use the same average blue for all of them, we lose the details in the image but gain space in storage and transmission. Ideally, one would like to identify higher-level regularities by analyzing repeated image patterns, for example, texture, objects, and so forth. This allows a higher-level, simpler, and more useful description of the scene, and for example, achieves better compression than compressing at the pixel level. If we have scanned document pages, we do not have random on/off pixels but bitmap images of characters. There is structure in the data, and we make use of this redundancy by finding a shorter description of the data: 16×16 bitmap of 'A' takes 32 bytes; its ASCII code is only 1 byte.

In *document clustering*, the aim is to group similar documents. For example, news reports can be subdivided as those related to politics, sports, fashion, arts, and so on. Commonly, a document is represented as a *bag of words*, that is, we predefine a lexicon of N words and each document is an N-dimensional binary vector whose element i is 1 if word i appears in the document; suffixes "-s" and "-ing" are removed to avoid duplicates and words such as "of," "and," and so forth, which are not informative, are not used. Documents are then grouped depending on the number of shared words. It is of course here critical how the lexicon is chosen.

Machine learning methods are also used in *bioinformatics*. DNA in our genome is the "blueprint of life" and is a sequence of bases, namely, A, G, C, and T. RNA is transcribed from DNA, and proteins are translated from the RNA. Proteins are what the living body is and does. Just as a DNA is a sequence of bases, a protein is a sequence of amino acids (as defined by bases). One application area of computer science in molecular biology is *alignment*, which is matching one sequence to another. This is a difficult string matching problem because strings may be quite long, there are many template strings to match against, and there may be deletions, insertions, and substitutions. Clustering is used in learning *motifs*, which are sequences of amino acids that occur repeatedly in proteins. Motifs are of interest because they may correspond to structural or functional

elements within the sequences they characterize. The analogy is that if the amino acids are letters and proteins are sentences, motifs are like words, namely, a string of letters with a particular meaning occurring frequently in different sentences.

1.2.5 Reinforcement Learning

In some applications, the output of the system is a sequence of *actions*. In such a case, a single action is not important; what is important is the *policy* that is the sequence of correct actions to reach the goal. There is no such thing as the best action in any intermediate state; an action is good if it is part of a good policy. In such a case, the machine learning program should be able to assess the goodness of policies and learn from past good action sequences to be able to generate a policy. Such learning methods are called *reinforcement learning* algorithms.

REINFORCEMENT LEARNING

A good example is *game playing* where a single move by itself is not that important; it is the sequence of right moves that is good. A move is good if it is part of a good game playing policy. Game playing is an important research area in both artificial intelligence and machine learning. This is because games are easy to describe and at the same time, they are quite difficult to play well. A game like chess has a small number of rules but it is very complex because of the large number of possible moves at each state and the large number of moves that a game contains. Once we have good algorithms that can learn to play games well, we can also apply them to applications with more evident economic utility.

A robot navigating in an environment in search of a goal location is another application area of reinforcement learning. At any time, the robot can move in one of a number of directions. After a number of trial runs, it should learn the correct sequence of actions to reach to the goal state from an initial state, doing this as quickly as possible and without hitting any of the obstacles.

One factor that makes reinforcement learning harder is when the system has unreliable and partial sensory information. For example, a robot equipped with a video camera has incomplete information and thus at any time is in a *partially observable state* and should decide taking into account this uncertainty; for example, it may not know its exact location in a room but only that there is a wall to its left. A task may also require a concurrent operation of *multiple agents* that should interact and

cooperate to accomplish a common goal. An example is a team of robots playing soccer.

1.3 Notes

Evolution is the major force that defines our bodily shape as well as our built-in instincts and reflexes. We also learn to change our behavior during our lifetime. This helps us cope with changes in the environment that cannot be predicted by evolution. Organisms that have a short life in a well-defined environment may have all their behavior built-in, but instead of hardwiring into us all sorts of behavior for any circumstance that we could encounter in our life, evolution gave us a large brain and a mechanism to learn, such that we could update ourselves with experience and adapt to different environments. When we learn the best strategy in a certain situation, that knowledge is stored in our brain, and when the situation arises again, when we re-cognize ("cognize" means to know) the situation, we can recall the suitable strategy and act accordingly. Learning has its limits though; there may be things that we can never learn with the limited capacity of our brains, just like we can never "learn" to grow a third arm, or an eye on the back of our head, even if either would be useful. See Leahey and Harris 1997 for learning and cognition from the point of view of psychology. Note that unlike in psychology, cognitive science, or neuroscience, our aim in machine learning is not to understand the processes underlying learning in humans and animals, but to build useful systems, as in any domain of engineering.

Almost all of science is fitting models to data. Scientists design experiments and make observations and collect data. They then try to extract knowledge by finding out simple models that explain the data they observed. This is called *induction* and is the process of extracting general rules from a set of particular cases.

We are now at a point that such analysis of data can no longer be done by people, both because the amount of data is huge and because people who can do such analysis are rare and manual analysis is costly. There is thus a growing interest in computer models that can analyze data and extract information automatically from them, that is, learn.

The methods we are going to discuss in the coming chapters have their origins in different scientific domains. Sometimes the same algorithm

was independently invented in more than one field, following a different historical path.

In statistics, going from particular observations to general descriptions is called *inference* and learning is called *estimation*. Classification is called *discriminant analysis* in statistics (McLachlan 1992; Hastie, Tibshirani, and Friedman 2001). Before computers were cheap and abundant, statisticians could only work with small samples. Statisticians, being mathematicians, worked mostly with simple parametric models that could be analyzed mathematically. In engineering, classification is called *pattern recognition* and the approach is nonparametric and much more empirical (Duda, Hart, and Stork 2001; Webb 1999). Machine learning is related to *artificial intelligence* (Russell and Norvig 2002) because an intelligent system should be able to adapt to changes in its environment. Application areas like vision, speech, and robotics are also tasks that are best learned from sample data. In electrical engineering, research in *signal processing* resulted in adaptive computer vision and speech programs. Among these, the development of *hidden Markov models* (HMM) for speech recognition is especially important.

In the late 1980s with advances in VLSI technology and the possibility of building parallel hardware containing thousands of processors, the field of *artificial neural networks* was reinvented as a possible theory to distribute computation over a large number of processing units (Bishop 1995). Over time, it has been realized in the neural network community that most neural network learning algorithms have their basis in statistics—for example, the multilayer perceptron is another class of nonparametric estimator—and claims of brainlike computation have started to fade.

In recent years, kernel-based algorithms, such as support vector machines, have become popular, which, through the use of kernel functions, can be adapted to various applications, especially in bioinformatics and language processing. It is common knowledge nowadays that a good representation of data is critical for learning and kernel functions turn out to be a very good way to introduce such expert knowledge.

Recently, with the reduced cost of storage and connectivity, it has become possible to have very large datasets available over the Internet, and this, coupled with cheaper computation, have made it possible to run learning algorithms on a lot of data. In the past few decades, it was generally believed that for artificial intelligence to be possible, we needed a new paradigm, a new type of thinking, a new model of computation

or a whole new set of algorithms. Taking into account the recent successes in machine learning in various domains, it may be claimed that what we needed was not new algorithms but a lot of example data and sufficient computing power to run the algorithms on that much data. For example, the roots of support vector machines go to potential functions, linear classifiers, and neighbor-based methods, proposed in the 1950s or the 1960s; it is just that we did not have fast computers or large storage then for these algorithms to show their full potential. It may be conjectured that tasks such as machine translation, and even planning, can be solved with such relatively simple learning algorithms but trained on large amounts of example data, or through long runs of trial and error. Intelligence seems not to originate from some outlandish formula, but rather from the patient, almost brute-force use of a simple, straightforward algorithm.

Data mining is the name coined in the business world for the application of machine learning algorithms to large amounts of data (Witten and Frank 2005; Han and Kamber 2006). In computer science, it used to be called *knowledge discovery in databases* (KDD).

Research in these different communities (statistics, pattern recognition, neural networks, signal processing, control, artificial intelligence, and data mining) followed different paths in the past with different emphases. In this book, the aim is to incorporate these emphases together to give a unified treatment of the problems and the proposed solutions to them.

1.4 Relevant Resources

The latest research on machine learning is distributed over journals and conferences from different fields. Dedicated journals are *Machine Learning* and *Journal of Machine Learning Research*. Journals with a neural network emphasis are *Neural Computation*, *Neural Networks*, and the *IEEE Transactions on Neural Networks*. Statistics journals like *Annals of Statistics* and *Journal of the American Statistical Association* also publish machine learning papers. *IEEE Transactions on Pattern Analysis and Machine Intelligence* is another source.

Journals on artificial intelligence, pattern recognition, fuzzy logic, and signal processing also contain machine learning papers. Journals with an emphasis on data mining are *Data Mining and Knowledge Discovery*, *IEEE*

Transactions on Knowledge and Data Engineering, and *ACM Special Interest Group on Knowledge Discovery and Data Mining Explorations Journal*.

The major conferences on machine learning are *Neural Information Processing Systems* (NIPS), *Uncertainty in Artificial Intelligence* (UAI), *International Conference on Machine Learning* (ICML), *European Conference on Machine Learning* (ECML), and *Computational Learning Theory* (COLT). *International Joint Conference on Artificial Intelligence* (IJCAI), as well as conferences on neural networks, pattern recognition, fuzzy logic, and genetic algorithms, have sessions on machine learning and conferences on application areas like computer vision, speech technology, robotics, and data mining.

There are a number of dataset repositories on the Internet that are used frequently by machine learning researchers for benchmarking purposes:

- *UCI Repository* for machine learning is the most popular repository: http://www.ics.uci.edu/~mlearn/MLRepository.html
- *UCI KDD Archive*: http://kdd.ics.uci.edu/summary.data.application.html
- *Statlib*: http://lib.stat.cmu.edu
- *Delve*: http://www.cs.utoronto.ca/~delve/

In addition to these, there are also repositories for particular applications, for example, computional biology, face recognition, speech recognition, and so forth.

New and larger datasets are constantly being added to these repositories, especially to the UCI repository. Still, some researchers believe that such repositories do not reflect the full characteristics of real data and are of limited scope, and therefore accuracies on datasets from such repositories are not indicative of anything. It may even be claimed that when some datasets from a fixed repository are used repeatedly while tailoring a new algorithm, we are generating a new set of "UCI algorithms" specialized for those datasets.

As we will see in later chapters, different algorithms are better on different tasks anyway, and therefore it is best to keep one application in mind, to have one or a number of large datasets drawn for that and compare algorithms on those, for that specific task.

Most recent papers by machine learning researchers are accessible over the Internet, and a good place to start searching is the NEC Research In-

dex at http://citeseer.ist.psu.edu. Most authors also make codes of their algorithms available over the Web. There are also free software packages implementing various machine learning algorithms, and among these, Weka is especially noteworthy: http://www.cs.waikato.ac.nz/ml/weka/.

1.5 Exercises

1. Imagine you have two possibilities: You can fax a document, that is, send the image, or you can use an optical character reader (OCR) and send the text file. Discuss the advantage and disadvantages of the two approaches in a comparative manner. When would one be preferable over the other?
2. Let us say we are building an OCR and for each character, we store the bitmap of that character as a template that we match with the read character pixel by pixel. Explain when such a system would fail. Why are barcode readers still used?
3. Assume we are given the task to build a system that can distinguish junk e-mail. What is in a junk e-mail that lets us know that it is junk? How can the computer detect junk through a syntactic analysis? What would you like the computer to do if it detects a junk e-mail—delete it automatically, move it to a different file, or just highlight it on the screen?
4. Let us say you are given the task of building an automated taxi. Define the constraints. What are the inputs? What is the output? How can you communicate with the passenger? Do you need to communicate with the other automated taxis, that is, do you need a "language"?
5. In basket analysis, we want to find the dependence between two items X and Y. Given a database of customer transactions, how can you find these dependencies? How would you generalize this to more than two items?
6. How can you predict the next command to be typed by the user? Or the next page to be downloaded over the Web? When would such a prediction be useful? When would it be annoying?
7. In your everyday newspaper, find five sample news reports for each category of politics, sports, and the arts. Go over these reports and find words that are used frequently for each category, which may help us discriminate between different categories. For example, a news report on politics is likely to include words such as "government," "recession," "congress," and so forth, whereas a news report on the arts may include "album," "canvas," or "theater." There are also words such as "goal" that are ambiguous.
8. If a face image is a 100×100 image, written in row-major, this is a 10,000-dimensional vector. If we shift the image one pixel to the right, this will be a

very different vector in the 10,000-dimensional space. How can we build face recognizers robust to such distortions?

9. Take a word, for example, "machine." Write it ten times. Also ask a friend to write it ten times. Analyzing these twenty images, try to find features, types of strokes, curvatures, loops, how you make the dots, and so on, that discriminate your handwriting from your friend's.

10. In estimating the price of a used car, rather than estimating the absolute price it makes more sense to estimate the percent depreciation over the original price. Why?

1.6 References

Bishop, C. M. 1995. *Neural Networks for Pattern Recognition.* Oxford: Oxford University Press.

Duda, R. O., P. E. Hart, and D. G. Stork. 2001. *Pattern Classification,* 2nd ed. New York: Wiley.

Han, J., and M. Kamber. 2006. *Data Mining: Concepts and Techniques,* 2nd ed. San Francisco: Morgan Kaufmann.

Hand, D. J. 1998. "Consumer Credit and Statistics." In *Statistics in Finance,* ed. D. J. Hand and S. D. Jacka, 69–81. London: Arnold.

Hastie, T., R. Tibshirani, and J. Friedman. 2001. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* New York: Springer.

Leahey, T. H., and R. J. Harris. 1997. *Learning and Cognition,* 4th ed. New York: Prentice Hall.

McLachlan, G. J. 1992. *Discriminant Analysis and Statistical Pattern Recognition.* New York: Wiley.

Russell, S., and P. Norvig. 2002. *Artificial Intelligence: A Modern Approach,* 2nd ed. New York: Prentice Hall.

Webb, A. 1999. *Statistical Pattern Recognition.* London: Arnold.

Witten, I. H., and E. Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques,* 2nd ed. San Francisco: Morgan Kaufmann.