Part I

Basic concepts

1

Pattern analysis

Pattern analysis deals with the automatic detection of patterns in data, and plays a central role in many modern artificial intelligence and computer science problems. By patterns we understand any relations, regularities or structure inherent in some source of data. By detecting significant patterns in the available data, a system can expect to make predictions about new data coming from the same source. In this sense the system has acquired generalisation power by '*learning*' something about the source generating the data. There are many important problems that can only be solved using this approach, problems ranging from bioinformatics to text categorization, from image analysis to web retrieval. In recent years, pattern analysis has become a standard software engineering approach, and is present in many commercial products.

Early approaches were efficient in finding linear relations, while nonlinear patterns were dealt with in a less principled way. The methods described in this book combine the theoretically well-founded approach previously limited to linear systems, with the flexibility and applicability typical of nonlinear methods, hence forming a remarkably powerful and robust class of pattern analysis techniques.

There has been a distinction drawn between statistical and syntactical pattern recognition, the former dealing essentially with vectors under statistical assumptions about their distribution, and the latter dealing with structured objects such as sequences or formal languages, and relying much less on statistical analysis. The approach presented in this book reconciles these two directions, in that it is capable of dealing with general types of data such as sequences, while at the same time addressing issues typical of statistical pattern analysis such as learning from finite samples.

1.1 Patterns in data

1.1.1 Data

This book deals with data and ways to exploit it through the identification of valuable knowledge. By data we mean the output of any observation, measurement or recording apparatus. This therefore includes images in digital format; vectors describing the state of a physical system; sequences of DNA; pieces of text; time series; records of commercial transactions, etc. By knowledge we mean something more abstract, at the level of relations between and patterns within the data. Such knowledge can enable us to make predictions about the source of the data or draw inferences about the relationships inherent in the data.

Many of the most interesting problems in AI and computer science in general are extremely complex often making it difficult or even impossible to specify an explicitly programmed solution. As an example consider the problem of recognising genes in a DNA sequence. We do not know how to specify a program to pick out the subsequences of, say, human DNA that represent genes. Similarly we are not able directly to program a computer to recognise a face in a photo. Learning systems offer an alternative methodology for tackling these problems. By exploiting the knowledge extracted from a sample of data, they are often capable of adapting themselves to infer a solution to such tasks. We will call this alternative approach to software design the *learning methodology*. It is also referred to as the *data driven* or *data based* approach, in contrast to the *theory driven* approach that gives rise to precise specifications of the required algorithms.

The range of problems that have been shown to be amenable to the learning methodology has grown very rapidly in recent years. Examples include text categorization; email filtering; gene detection; protein homology detection; web retrieval; image classification; handwriting recognition; prediction of loan defaulting; determining properties of molecules, etc. These tasks are very hard or in some cases impossible to solve using a standard approach, but have all been shown to be tractable with the learning methodology. Solving these problems is not just of interest to researchers. For example, being able to predict important properties of a molecule from its structure could save millions of dollars to pharmaceutical companies that would normally have to test candidate drugs in expensive experiments, while being able to identify a combination of biomarker proteins that have high predictive power could result in an early cancer diagnosis test, potentially saving many lives.

In general, the field of pattern analysis studies systems that use the learn-

ing methodology to discover *patterns in data*. The patterns that are sought include many different types such as classification, regression, cluster analysis (sometimes referred to together as *statistical pattern recognition*), feature extraction, grammatical inference and parsing (sometimes referred to as *syntactical pattern recognition*). In this book we will draw concepts from all of these fields and at the same time use examples and case studies from some of the applications areas mentioned above: bioinformatics, machine vision, information retrieval, and text categorization.

It is worth stressing that while traditional statistics dealt mainly with data in vector form in what is known as *multivariate statistics*, the data for many of the important applications mentioned above are non-vectorial. We should also mention that pattern analysis in computer science has focussed mainly on classification and regression, to the extent that pattern analysis is synonymous with classification in the neural network literature. It is partly to avoid confusion between this more limited focus and our general setting that we have introduced the term *pattern analysis*.

1.1.2 Patterns

Imagine a dataset containing thousands of observations of planetary positions in the solar system, for example daily records of the positions of each of the nine planets. It is obvious that the position of a planet on a given day is not independent of the position of the same planet in the preceding days: it can actually be predicted rather accurately based on knowledge of these positions. The dataset therefore contains a certain amount of redundancy, that is information that can be reconstructed from other parts of the data, and hence that is not strictly necessary. In such cases the dataset is said to be *redundant*: simple laws can be extracted from the data and used to reconstruct the position of each planet on each day. The rules that govern the position of the planets are known as Kepler's laws. Johannes Kepler discovered his three laws in the seventeenth century by analysing the planetary positions recorded by Tycho Brahe in the preceding decades.

Kepler's discovery can be viewed as an early example of pattern analysis, or data-driven analysis. By assuming that the laws are invariant, they can be used to make predictions about the outcome of future observations. The laws correspond to regularities present in the planetary data and by inference therefore in the planetary motion itself. They state that the planets move in ellipses with the sun at one focus; that equal areas are swept in equal times by the line joining the planet to the sun; and that the period P (the time

	D	P	D^2	P^3
Mercury	0.24	0.39	0.058	0.059
Venus	0.62	0.72	0.38	0.39
Earth	1.00	1.00	1.00	1.00
Mars	1.88	1.53	3.53	3.58
Jupiter	11.90	5.31	142.00	141.00
Saturn	29.30	9.55	870.00	871.00

Table 1.1. *An example of a pattern in data: the quantity D^2/P^3 remains invariant for all the planets. This means that we could compress the data by simply listing one column or that we can predict one of the values for new previously unknown planets, as happened with the discovery of the outer planets.*

of one revolution around the sun) and the average distance D from the sun are related by the equation $P^3 = D^2$ for each planet.

Example 1.1 From Table 1.1 we can observe two potential properties of redundant datasets: on the one hand they are *compressible* in that we could construct the table from just one column of data with the help of Kepler's third law, while on the other hand they are *predictable* in that we can, for example, infer from the law the distances of newly discovered planets once we have measured their period. The predictive power is a direct consequence of the presence of the possibly hidden relations in the data. It is these relations once discovered that enable us to predict and therefore manipulate new data more effectively. ■

Typically we anticipate predicting one feature as a function of the remaining features: for example the distance as a function of the period. For us to be able to do this, the relation must be invertible, so that the desired feature can be expressed as a function of the other values. Indeed we will seek relations that have such an explicit form whenever this is our intention. Other more general relations can also exist within data, can be detected and can be exploited. For example, if we find a general relation that is expressed as an invariant function f that satisfies

$$f(\mathbf{x}) = 0, \tag{1.1}$$

where $\mathbf{x}$ is a data item, we can use it to identify novel or faulty data items for which the relation fails, that is for which $f(\mathbf{x}) \neq 0$. In such cases it is, however, harder to realise the potential for compressibility since it would require us to define a lower-dimensional coordinate system on the manifold defined by equation (1.1).

Kepler's laws are accurate and hold for all planets of a given solar system. We refer to such relations as *exact*. The examples that we gave above included problems such as loan defaulting, that is the prediction of which borrowers will fail to repay their loans based on information available at the time the loan is processed. It is clear that we cannot hope to find an exact prediction in this case since there will be factors beyond those available to the system, which may prove crucial. For example, the borrower may lose his job soon after taking out the loan and hence find himself unable to fulfil the repayments. In such cases the most the system can hope to do is find relations that hold with a certain probability. Learning systems have succeeded in finding such relations. The two properties of compressibility and predictability are again in evidence. We can specify the relation that holds for much of the data and then simply append a list of the exceptional cases. Provided the description of the relation is succinct and there are not too many exceptions, this will result in a reduction in the size of the dataset. Similarly, we can use the relation to make predictions, for example whether the borrower will repay his or her loan. Since the relation holds with a certain probability we will have a good chance that the prediction will be fulfilled. We will call relations that hold with a certain probability *statistical*.

Predicting properties of a substance based on its molecular structure is hindered by a further problem. In this case, for properties such as boiling point that take real number values, the relations sought will necessarily have to be approximate in the sense that we cannot expect an exact prediction. Typically we may hope that the expected error in the prediction will be small, or that with high probability the true value will be within a certain margin of the prediction, but our search for patterns must necessarily seek a relation that is approximate. One could claim that Kepler's laws are approximate if for no other reason because they fail to take general relativity into account. In the cases of interest to learning systems, however, the approximations will be much looser than those affecting Kepler's laws. Relations that involve some inaccuracy in the values accepted are known as *approximate*. For approximate relations we can still talk about prediction, though we must qualify the accuracy of the estimate and quite possibly the probability with which it applies. Compressibility can again be demonstrated if we accept that specifying the error corrections between the value output by the rule and the true value, take less space if they are small.

The relations that make a dataset redundant, that is the laws that we extract by mining it, are called *patterns* throughout this book. Patterns can be deterministic relations like Kepler's exact laws. As indicated above

other relations are approximate or only holds with a certain probability. We are interested in situations where exact laws, especially ones that can be described as simply as Kepler's, may not exist. For this reason we will understand a *pattern* to be any relation present in the data, whether it be exact, approximate or statistical.

Example 1.2 Consider the following artificial example, describing some observations of planetary positions in a two dimensional orthogonal coordinate system. Note that this is certainly not what Kepler had in Tycho's data.

x	y	x^2	y^2	xy
0.8415	0.5403	0.7081	0.2919	0.4546
0.9093	−0.4161	0.8268	0.1732	−0.3784
0.1411	−0.99	0.0199	0.9801	−0.1397
−0.7568	−0.6536	0.5728	0.4272	0.4947
−0.9589	0.2837	0.9195	0.0805	−0.272
−0.2794	0.9602	0.0781	0.9219	−0.2683
0.657	0.7539	0.4316	0.5684	0.4953
0.9894	−0.1455	0.9788	0.0212	−0.144
0.4121	−0.9111	0.1698	0.8302	−0.3755
−0.544	−0.8391	0.296	0.704	0.4565

The left plot of Figure 1.1 shows the data in the (x, y) plane. We can make many assumptions about the law underlying such positions. However if we consider the quantity $c_1x^2+c_2y^2+c_3xy+c_4x+c_5y+c_6$ we will see that it is constant for some choice of the parameters, indeed as shown in the left plot of Figure 1.1 we obtain a linear relation with just two features, x^2 and y^2. This would not generally the case if the data were random, or even if the trajectory was following a curve different from a quadratic. In fact this invariance in the data means that the planet follows an elliptic trajectory. By changing the coordinate system the relation has become linear. ■

In the example we saw how applying a change of coordinates to the data leads to the representation of a pattern changing. Using the initial coordinate system the pattern was expressed as a quadratic form, while in the coordinate system using monomials it appeared as a linear function. The possibility of transforming the representation of a pattern by changing the coordinate system in which the data is described will be a recurrent theme in this book.

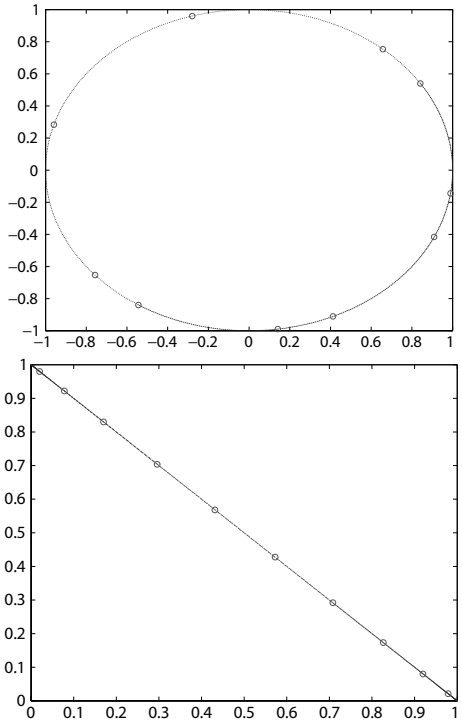

Fig. 1.1. The artificial planetary data lying on an ellipse in two dimensions and the same data represented using the features x^2 and y^2 showing a linear relation

The pattern in the example had the form of a function f that satisfied

$$f(\mathbf{x}) = 0,$$

for all the data points $\mathbf{x}$. We can also express the pattern described by Kepler's third law in this form

$$f(D, P) = D^2 - P^3 = 0.$$

Alternatively

$$g(D, P) = 2\log D - 3\log P = 0.$$

Similarly, if we have a function g that for each data item $(\mathbf{x}, \mathbf{y})$ predicts some output values $\mathbf{y}$ as a function of the input features $\mathbf{x}$, we can express the pattern in the form

$$f(\mathbf{x}, \mathbf{y}) = \mathcal{L}(g(\mathbf{x}), \mathbf{y}) = 0,$$

where $\mathcal{L} : Y \times Y \to \mathbb{R}^+$ is a so-called *loss function* that measures the

disagreement between its two arguments outputting 0 if and only if the two arguments are the same and outputs a positive discrepancy if they differ.

Definition 1.3 A general exact pattern for a data source is a non-trivial function f that satisfies

$$f(\mathbf{x}) = 0,$$

for all of the data, $\mathbf{x}$, that can arise from the source. ■

The definition only covers exact patterns. We first consider the relaxation required to cover the case of approximate patterns. Taking the example of a function g that predicts the values $\mathbf{y}$ as a function of the input features $\mathbf{x}$ for a data item $(\mathbf{x}, \mathbf{y})$, if we cannot expect to obtain an exact equality between $g(\mathbf{x})$ and $\mathbf{y}$, we use the loss function $\mathcal{L}$ to measure the amount of mismatch. This can be done by allowing the function to output 0 when the two arguments are similar, but not necessarily identical, or by allowing the function f to output small, non-zero positive values. We will adopt the second approach since when combined with probabilistic patterns it gives a distinct and useful notion of probabilistic matching.

Definition 1.4 A general approximate pattern for a data source is a non-trivial function f that satisfies

$$f(\mathbf{x}) \approx 0$$

for all of the data $\mathbf{x}$, that can arise from the source. ■

We have deliberately left vague what approximately equal to zero might mean in a particular context.

Finally, we consider statistical patterns. In this case there is a probability distribution that generates the data. In many cases the individual data items can be assumed to be generate independently and identically, a case often referred to as independently and identically distributed or i.i.d. for short. We will use the symbol $\mathbb{E}$ to denote the *expectation* of some quantity under a distribution. If we wish to indicate the distribution over which the expectation is taken we add either the distribution or the variable as an index.

Note that our definitions of patterns hold for each individual data item in the case of exact and approximate patterns, but for the case of a statistical pattern we will consider the expectation of a function according to the underlying distribution. In this case we require the pattern function to be positive to ensure that a small expectation arises from small function values

and not through the averaging of large positive and negative outputs. This can always be achieved by taking the absolute value of a pattern function that can output negative values.

Definition 1.5 A general statistical pattern for a data source generated i.i.d. according to a distribution $\mathcal{D}$ is a non-trivial non-negative function f that satisfies

$$\mathbb{E}_{\mathcal{D}} f(\mathbf{x}) = \mathbb{E}_{\mathbf{x}} f(\mathbf{x}) \approx 0.$$

■

If the distribution does not satisfy the i.i.d. requirement this is usually as a result of dependencies between data items generated in sequence or because of slow changes in the underlying distribution. A typical example of the first case is time series data. In this case we can usually assume that the source generating the data is ergodic, that is, the dependency decays over time to a probability that is i.i.d. It is possible to develop an analysis that approximates i.i.d. for this type of data. Handling changes in the underlying distribution has also been analysed theoretically but will also be beyond the scope of this book.

Remark 1.6 [Information theory] It is worth mentioning how the patterns we are considering and the corresponding compressibility are related to the traditional study of statistical information theory. Information theory defines the entropy of a (not necessarily i.i.d.) source of data and limits the compressibility of the data as a function of its entropy. For the i.i.d. case it relies on knowledge of the exact probabilities of the finite set of possible items.

Algorithmic information theory provides a more general framework for defining redundancies and regularities in datasets, and for connecting them with the compressibility of the data. The framework considers all computable functions, something that for finite sets of data becomes too rich a class. For in general we do not have access to all of the data and certainly not an exact knowledge of the distribution that generates it. ■

Our information about the data source must rather be gleaned from a finite set of observations generated according to the same underlying distribution. Using only this information a pattern analysis algorithm must be able to identify patterns. Hence, we give the following general definition of a pattern analysis algorithm.