1

The Learning Methodology

The construction of machines capable of learning from experience has for a long time been the object of both philosophical and technical debate. The technical aspect of the debate has received an enormous impetus from the advent of electronic computers. They have demonstrated that machines can display a significant level of learning ability, though the boundaries of this ability are far from being clearly defined.

The availability of reliable learning systems is of strategic importance, as there are many tasks that cannot be solved by classical programming techniques, since no mathematical model of the problem is available. So for example it is not known how to write a computer program to perform hand-written character recognition, though there are plenty of examples available. It is therefore natural to ask if a computer could be trained to recognise the letter 'A' from examples – after all this is the way humans learn to read. We will refer to this approach to problem solving as the learning methodology

The same reasoning applies to the problem of finding genes in a DNA sequence, filtering email, detecting or recognising objects in machine vision, and so on. Solving each of these problems has the potential to revolutionise some aspect of our life, and for each of them machine learning algorithms could provide the key to its solution.

In this chapter we will introduce the important components of the learning methodology, give an overview of the different kinds of learning and discuss why this approach has such a strategic importance. After the framework of the learning methodology has been introduced, the chapter ends with a roadmap for the rest of the book, anticipating the key themes, and indicating why Support Vector Machines meet many of the challenges confronting machine learning systems. As this roadmap will descibe the role of the different chapters, we urge our readers to refer to it before delving further into the book.

1.1 Supervised Learning

When computers are applied to solve a practical problem it is usually the case that the method of deriving the required output from a set of inputs can be described explicitly. The task of the system designer and eventually the programmer implementing the specifications will be to translate that method

into a sequence of instructions which the computer will follow to achieve the desired effect.

As computers are applied to solve more complex problems, however, situations can arise in which there is no known method for computing the desired output from a set of inputs, or where that computation may be very expensive. Examples of this type of situation might be modelling a complex chemical reaction, where the precise interactions of the different reactants are not known, or classification of protein types based on the DNA sequence from which they are generated, or the classification of credit applications into those who will default and those who will repay the loan.

These tasks cannot be solved by a traditional programming approach since the system designer cannot precisely specify the method by which the correct output can be computed from the input data. An alternative strategy for solving this type of problem is for the computer to attempt to learn the input/output functionality from examples, in the same way that children learn which are sports cars simply by being told which of a large number of cars are sporty rather than by being given a precise specification of sportiness. The approach of using examples to synthesise programs is known as the *learning methodology*, and in the particular case when the examples are input/output pairs it is called *supervised learning*. The examples of input/output functionality are referred to as the *training data*.

The input/output pairings typically reflect a functional relationship mapping inputs to outputs, though this is not always the case as for example when the outputs are corrupted by noise. When an underlying function from inputs to outputs exists it is referred to as the *target function*. The estimate of the target function which is learnt or output by the learning algorithm is known as the *solution* of the learning problem. In the case of classification this function is sometimes referred to as the *decision function*. The solution is chosen from a set of candidate functions which map from the input space to the output domain. Usually we will choose a particular set or class of candidate functions known as *hypotheses* before we begin trying to learn the correct function. For example, so-called *decision trees* are hypotheses created by constructing a binary tree with simple decision functions at the internal nodes and output values at the leaves. Hence, we can view the choice of the set of hypotheses (or *hypothesis space*) as one of the key ingredients of the learning strategy. The algorithm which takes the training data as input and selects a hypothesis from the hypothesis space is the second important ingredient. It is referred to as the *learning algorithm*.

In the case of learning to distinguish sports cars the output is a simple yes/no tag which we can think of as a binary output value. For the problem of recognising protein types, the output value will be one of a finite number of categories, while the output values when modelling a chemical reaction might be the concentrations of the reactants given as real values. A learning problem with binary outputs is referred to as a *binary classification* problem, one with a finite number of categories as *multi-class classification*, while for real-valued outputs the problem becomes known as *regression*. This book will consider all of these

types of learning, though binary classification is always considered first as it is often the simplest case.

There are other types of learning that will not be considered in this book. For example *unsupervised learning* considers the case where there are no output values and the learning task is to gain some understanding of the process that generated the data. This type of learning includes density estimation, learning the support of a distribution, clustering, and so on. There are also models of learning which consider more complex interactions between a learner and their environment. Perhaps the simplest case is when the learner is allowed to query the environment about the output associated with a particular input. The study of how this affects the learner's ability to learn different tasks is known as *query learning*. Further complexities of interaction are considered in *reinforcement learning*, where the learner has a range of actions at their disposal which they can take to attempt to move towards states where they can expect high rewards. The learning methodology can play a part in reinforcement learning if we treat the optimal action as the output of a function of the current state of the learner. There are, however, significant complications since the quality of the output can only be assessed indirectly as the consequences of an action become clear.

Another type of variation in learning models is the way in which the training data are generated and how they are presented to the learner. For example, there is a distinction made between *batch* learning in which all the data are given to the learner at the start of learning, and *on-line* learning in which the learner receives one example at a time, and gives their estimate of the output, before receiving the correct value. In on-line learning they update their current hypothesis in response to each new example and the quality of learning is assessed by the total number of mistakes made during learning.

The subject of this book is a family of techniques for learning to perform input/output mappings from labelled examples for the most part in the batch setting, that is for applying the supervised learning methodology from batch training data.

1.2 Learning and Generalisation

We discussed how the quality of an on-line learning algorithm can be assessed in terms of the number of mistakes it makes during the training phase. It is not immediately clear, however, how we can assess the quality of a hypothesis generated during batch learning. Early machine learning algorithms aimed to learn representations of simple symbolic functions that could be understood and verified by experts. Hence, the goal of learning in this paradigm was to output a hypothesis that performed the correct classification of the training data and early learning algorithms were designed to find such an accurate fit to the data. Such a hypothesis is said to be *consistent*. There are two problems with the goal of generating a verifiable consistent hypothesis.

The first is that the function we are trying to learn may not have a simple representation and hence may not be easily verified in this way. An example

of this situation is the identification of genes within a DNA sequence. Certain subsequences are genes and others are not, but there is no simple way to categorise which are which.

The second problem is that frequently training data are noisy and so there is no guarantee that there is an underlying function which correctly maps the training data. The example of credit checking is clearly in this category since the decision to default may be a result of factors simply not available to the system. A second example would be the classification of web pages into categories, which again can never be an exact science.

The type of data that is of interest to machine learning practitioners is increasingly of these two types, hence rendering the proposed measure of quality difficult to implement. There is, however, a more fundamental problem with this approach in that even when we can find a hypothesis that is consistent with the training data, it may not make correct classifications of unseen data. The ability of a hypothesis to correctly classify data not in the training set is known as its *generalisation*, and it is this property that we shall aim to optimise.

Shifting our goal to generalisation removes the need to view our hypothesis as a correct representation of the true function. If the hypothesis gives the right output it satisfies the generalisation criterion, which in this sense has now become a functional measure rather than a descriptional one. In this sense the criterion places no constraints on the size or on the 'meaning' of the hypothesis – for the time being these can be considered to be arbitrary.

This change of emphasis will be somewhat counteracted when we later search for compact representations (that is short descriptions) of hypotheses, as these can be shown to have good generalisation properties, but for the time being the change can be regarded as a move from symbolic to subsymbolic representations.

A precise definition of these concepts will be given in Chapter 4, when we will motivate the particular models we shall be using.

1.3 Improving Generalisation

The generalisation criterion places an altogether different constraint on the learning algorithm. This is most amply illustrated by the extreme case of rote learning. Many classical algorithms of machine learning are capable of representing any function and for difficult training sets will give a hypothesis that behaves like a rote learner. By a rote learner we mean one that correctly classifies the data in the training set, but makes essentially uncorrelated predictions on unseen data. For example, decision trees can grow so large that there is a leaf for each training example. Hypotheses that become too complex in order to become consistent are said to *overfit*. One way of trying to control this difficulty is to restrict the size of the hypothesis, for example pruning the size of the decision tree. Ockham's razor is a principle that motivates this approach, suggesting that unnecessary complications are not helpful, or perhaps more accurately complications must pay for themselves by giving significant improvements in the classification rate on the training data.

These ideas have a long history as the mention of Ockham suggests. They can be used to motivate heuristic trade-offs between complexity and accuracy and various principles have been proposed for choosing the optimal compromise between the two. As an example the *Minimum Description Length* (MDL) principle proposes to use the set of hypotheses for which the description of the chosen function together with the list of training errors is shortest.

The approach that we will adopt is to motivate the trade-off by reference to statistical bounds on the generalisation error. These bounds will typically depend on certain quantities such as the margin of the classifier, and hence motivate algorithms which optimise the particular measure. The drawback of such an approach is that the algorithm is only as good as the result that motivates it. On the other hand the strength is that the statistical result provides a well-founded basis for the approach, hence avoiding the danger of a heuristic that may be based on a misleading intuition.

The fact that the algorithm design is based on a statistical result does not mean that we ignore the computational complexity of solving the particular optimisation problem. We are interested in techniques that will scale from toy problems to large realistic datasets of hundreds of thousands of examples. It is only by performing a principled analysis of the computational complexity that we can avoid settling for heuristics that work well on small examples, but break down once larger training sets are used. The theory of computational complexity identifies two classes of problems. For the first class there exist algorithms that run in time polynomial in the size of the input, while for the second the existence of such an algorithm would imply that any problem for which we can check a solution in polynomial time can also be solved in polynomial time. This second class of problems is known as the NP-complete problems and it is generally believed that these problems cannot be solved efficiently.

In Chapter 4 we will describe in more detail the type of statistical result that will motivate our algorithms. We distinguish between results that measure generalisation performance that can be obtained with a given finite number of training examples, and asymptotic results, which study how the generalisation behaves as the number of examples tends to infinity. The results we will introduce are of the former type, an approach that was pioneered by Vapnik and Chervonenkis.

We should emphasise that alternative algorithms to those we will describe can be motivated by other approaches to analysing the learning methodology. We will point to relevant literature describing some other approaches within the text. We would, however, like to mention the Bayesian viewpoint in a little more detail at this stage.

The starting point for Bayesian analysis is a prior distribution over the set of hypotheses that describes the learner's prior belief of the likelihood of a particular hypothesis generating the data. Once such a prior has been assumed together with a model of how the data have been corrupted by noise, it is possible in principle to estimate the most likely hypothesis given the particular training set, and even to perform a weighted average over the set of likely hypotheses.

If no restriction is placed over the set of all possible hypotheses (that is all

possible functions from the input space to the output domain), then learning is impossible since no amount of training data will tell us how to classify unseen examples. Problems also arise if we allow ourselves the freedom of choosing the set of hypotheses after seeing the data, since we can simply assume all of the prior probability on the correct hypotheses. In this sense it is true that all learning systems have to make some prior assumption of a Bayesian type often called the *learning bias*. We will place the approach we adopt in this context in Chapter 4.

1.4 Attractions and Drawbacks of Learning

It is not surprising that the promise of the learning methodology should be so tantalising. Firstly, the range of applications that can potentially be solved by such an approach is very large. Secondly, it appears that we can also avoid much of the laborious design and programming inherent in the traditional solution methodology, at the expense of collecting some labelled data and running an off-the-shelf algorithm for learning the input/output mapping. Finally, there is the attraction of discovering insights into the way that humans function, an attraction that so inspired early work in neural networks, occasionally to the detriment of scientific objectivity.

There are, however, many difficulties inherent in the learning methodology, difficulties that deserve careful study and analysis. One example is the choice of the class of functions from which the input/output mapping must be sought. The class must be chosen to be sufficiently rich so that the required mapping or an approximation to it can be found, but if the class is too large the complexity of learning from examples can become prohibitive, particularly when taking into account the number of examples required to make statistically reliable inferences in a large function class. Hence, learning in three-node neural networks is known to be NP-complete, while the problem of minimising the number of training errors of a thresholded linear function is also NP-hard. In view of these difficulties it is immediately apparent that there are severe restrictions on the applicability of the approach, and any suggestions of a panacea are highly misleading.

In practice these problems manifest themselves in specific learning difficulties. The first is that the learning algorithm may prove inefficient as for example in the case of local minima. The second is that the size of the output hypothesis can frequently become very large and impractical. The third problem is that if there are only a limited number of training examples too rich a hypothesis class will lead to overfitting and hence poor generalisation. The fourth problem is that frequently the learning algorithm is controlled by a large number of parameters that are often chosen by tuning heuristics, making the system difficult and unreliable to use.

Despite the drawbacks, there have been notable successes in the application of the learning methodology to problems of practical interest. Unfortunately, however, there is frequently a lack of understanding about the conditions that

will render an application successful, in both algorithmic and statistical inference terms. We will see in the next section that Support Vector Machines address all of these problems.

1.5 Support Vector Machines for Learning

Support Vector Machines (SVM) are learning systems that use a hypothesis space of linear functions in a high dimensional feature space, trained with a learning algorithm from optimisation theory that implements a learning bias derived from statistical learning theory. This learning strategy introduced by Vapnik and co-workers is a principled and very powerful method that in the few years since its introduction has already outperformed most other systems in a wide variety of applications.

This book gives an introduction to Support Vector Machines by describing the hypothesis space and its representation in Chapters 2 and 3, the learning bias in Chapter 4, and the learning algorithm in Chapters 5 and 7. Chapter 6 is the key chapter in which all these components are brought together, while Chapter 8 gives an overview of some applications that have been made to real-world problems. The book is written in a modular fashion so that readers familiar with the material covered in a particular chapter can safely bypass that section. In particular, if the reader wishes to get a direct introduction to what an SVM is and how to implement one, they should move straight to Chapters 6 and 7.

Chapter 2 introduces linear learning machines one of the main building blocks of the system. Chapter 3 deals with kernel functions which are used to define the implicit feature space in which the linear learning machines operate. The use of kernel functions is the key to the efficient use of high dimensional feature spaces. The danger of overfitting inherent in high dimensions requires a sophisticated learning bias provided by the statistical learning theory covered in Chapter 4. Optimisation theory covered in Chapter 5 gives a precise characterisation of the properties of the solution which guide the implementation of efficient learning algorithms described in Chapter 7, and ensure that the output hypothesis has a compact representation. The particular choice of a convex learning bias also results in the absence of local minima so that solutions can always be found efficiently even for training sets with hundreds of thousands of examples, while the compact representation of the hypothesis means that evaluation on new inputs is very fast. Hence, the four problems of efficiency of training, efficiency of testing, overfitting and algorithm parameter tuning are all avoided in the SVM design.

1.6 Exercises

1. Describe the process of distinguishing between reptiles and mammals as a binary classification problem. What is the input space? How might the inputs be represented for computer processing of such data? Give an example of a function realising the classification rule.

2. Repeat Exercise 1 for the following categories of animals: birds/fishes; fishes/mammals; mammals/birds. Write down the list of decision functions used in the four decision rules.

3. You are given a set of correctly labelled mammals and fishes:

$$\{\text{dog, cat, dolphin}\},\ \{\text{goldfish, shark, tuna}\}.$$

Taking the hypothesis space as the set of four functions obtained in Exercises 1 and 2, how would you pick the correct decision rule?

1.7 Further Reading and Advanced Topics

The problem of learning from data has been investigated by philosophers throughout history, under the name of 'inductive inference'. Although this might seem surprising today, it was not until the 20th century that pure induction was recognised as impossible unless one assumes some prior knowledge. This conceptual achievement is essentially due to the fundamental work of Karl Popper [119].

There is a long history of studying this problem within the statistical framework. Gauss proposed the idea of least squares regression in the 18th century, while Fisher's approach [40] to classification in the 1930s still provides the starting point for most analysis and methods.

Researchers in the area of artificial intelligence started to consider the problem of learning from its very beginning. Alan Turing [154] proposed the idea of learning machines in 1950, contesting the belief of Lady Lovelace that 'the machine can only do what we know how to order it to do'. Also there is a foresight of subsymbolic learning in that paper, when Turing comments: 'An important feature of a learning machine is that its teacher will often be very largely ignorant of quite what is going on inside, although he may still be able to some extent to predict his pupil's behaviour.' Just a few years later the first examples of learning machines were developed, for example Arthur Samuel's draughts player [124] was an early example of reinforcement learning, while Frank Rosenblatt's perceptron [122] contained many of the features of the systems discussed in the next chapter. In particular, the idea of modelling learning problems as problems of search in a suitable hypothesis space is characteristic of the artificial intelligence approach. Solomonoff also formally studied the problem of learning as inductive inference in the famous papers [151] and [152].

The development of learning algorithms became an important sub field of artificial intelligence, eventually forming the separate subject area of *machine learning*. A very readable 'first introduction' to many problems in machine learning is provided by Tom Mitchell's book *Machine learning* [99]. Support Vector Machines were introduced by Vapnik and his co-workers in the COLT paper [19] and is described in more detail in Vapnik's book [159].

These references are also given on the website **www.support-vector.net**, which will be kept up to date with new work, pointers to software and papers that are available on-line.