Chapter 1

Introduction

In this book we will be concerned with supervised learning, which is the problem of learning input-output mappings from empirical data (the training dataset). Depending on the characteristics of the output, this problem is known as either regression, for continuous outputs, or classification, when outputs are discrete.

digit classification

A well known example is the classification of images of handwritten digits. The training set consists of small digitized images, together with a classification from $0, \ldots, 9$, normally provided by a human. The goal is to learn a mapping from image to classification label, which can then be used on new, unseen images. Supervised learning is an attractive way to attempt to tackle this problem, since it is not easy to specify accurately the characteristics of, say, the handwritten digit 4.

robotic control

An example of a regression problem can be found in robotics, where we wish to learn the inverse dynamics of a robot arm. Here the task is to map from the state of the arm (given by the positions, velocities and accelerations of the joints) to the corresponding torques on the joints. Such a model can then be used to compute the torques needed to move the arm along a given trajectory. Another example would be in a chemical plant, where we might wish to predict the yield as a function of process parameters such as temperature, pressure, amount of catalyst etc.

the dataset

In general we denote the input as $\mathbf{x}$, and the output (or target) as y. The input is usually represented as a vector $\mathbf{x}$ as there are in general many input variables—in the handwritten digit recognition example one may have a 256-dimensional input obtained from a raster scan of a 16×16 image, and in the robot arm example there are three input measurements for each joint in the arm. The target y may either be continuous (as in the regression case) or discrete (as in the classification case). We have a dataset $\mathcal{D}$ of n observations, $\mathcal{D} = \{(\mathbf{x}_i, y_i) | i = 1, \ldots, n\}$.

training is inductive

Given this training data we wish to make predictions for new inputs $\mathbf{x}_*$ that we have not seen in the training set. Thus it is clear that the problem at hand is *inductive*; we need to move from the finite training data $\mathcal{D}$ to a

function f that makes predictions for all possible input values. To do this we must make assumptions about the characteristics of the underlying function, as otherwise any function which is consistent with the training data would be equally valid. A wide variety of methods have been proposed to deal with the supervised learning problem; here we describe two common approaches. The first is to restrict the class of functions that we consider, for example by only considering linear functions of the input. The second approach is (speaking rather loosely) to give a prior probability to every possible function, where higher probabilities are given to functions that we consider to be more likely, for example because they are smoother than other functions.[1] The first approach has an obvious problem in that we have to decide upon the richness of the class of functions considered; if we are using a model based on a certain class of functions (e.g. linear functions) and the target function is not well modelled by this class, then the predictions will be poor. One may be tempted to increase the flexibility of the class of functions, but this runs into the danger of overfitting, where we can obtain a good fit to the training data, but perform badly when making test predictions.

two approaches

The second approach appears to have a serious problem, in that surely there are an uncountably infinite set of possible functions, and how are we going to compute with this set in finite time? This is where the Gaussian process comes to our rescue. A Gaussian *process* is a generalization of the Gaussian probability *distribution*. Whereas a probability distribution describes random variables which are scalars or vectors (for multivariate distributions), a stochastic *process* governs the properties of functions. Leaving mathematical sophistication aside, one can loosely think of a function as a very long vector, each entry in the vector specifying the function value $f(x)$ at a particular input x. It turns out, that although this idea is a little naïve, it is surprisingly close what we need. Indeed, the question of how we deal computationally with these infinite dimensional objects has the most pleasant resolution imaginable: if you ask only for the properties of the function at a finite number of points, then inference in the Gaussian process will give you the same answer if you ignore the infinitely many other points, as if you would have taken them all into account! And these answers are consistent with answers to any other finite queries you may have. One of the main attractions of the Gaussian process framework is precisely that it unites a sophisticated and consistent view with computational tractability.

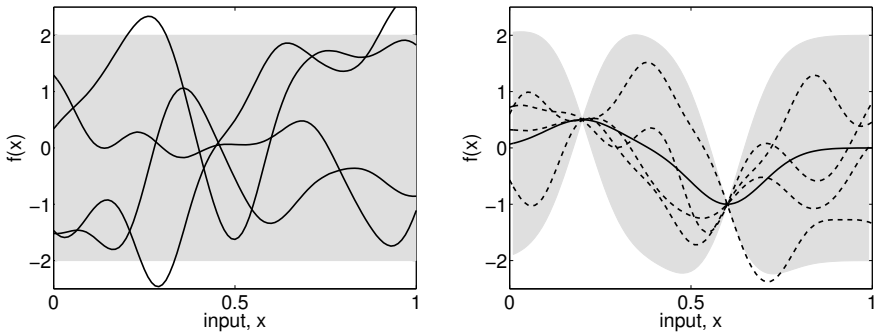
Gaussian process

consistency

tractability

It should come as no surprise that these ideas have been around for some time, although they are perhaps not as well known as they might be. Indeed, many models that are commonly employed in both machine learning and statistics are in fact special cases of, or restricted kinds of Gaussian processes. In this volume, we aim to give a systematic and unified treatment of the area, showing connections to related models.

[1] These two approaches may be regarded as imposing a *restriction* bias and a *preference* bias respectively; see e.g. Mitchell [1997].



(a), prior (b), posterior

Figure 1.1: Panel (a) shows four samples drawn from the prior distribution. Panel (b) shows the situation after two datapoints have been observed. The mean prediction is shown as the solid line and four samples from the posterior are shown as dashed lines. In both plots the shaded region denotes twice the standard deviation at each input value x.

1.1 A Pictorial Introduction to Bayesian Modelling

In this section we give graphical illustrations of how the second (Bayesian) method works on some simple regression and classification examples.

We first consider a simple 1-d *regression* problem, mapping from an input x to an output $f(x)$. In Figure 1.1(a) we show a number of sample functions drawn at random from the *prior* distribution over functions specified by a particular Gaussian process which favours smooth functions. This prior is taken to represent our prior beliefs over the kinds of functions we expect to observe, before seeing any data. In the absence of knowledge to the contrary we have assumed that the average value over the sample functions at each x is zero. Although the specific random functions drawn in Figure 1.1(a) do not have a mean of zero, the mean of $f(x)$ values for any fixed x would become zero, independent of x as we kept on drawing more functions. At any value of x we can also characterize the variability of the sample functions by computing the variance at that point. The shaded region denotes twice the pointwise standard deviation; in this case we used a Gaussian process which specifies that the prior variance does not depend on x.

regression

random functions

mean function

pointwise variance

Suppose that we are then given a dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2)\}$ consisting of two observations, and we wish now to only consider functions that pass though these two data points exactly. (It is also possible to give higher preference to functions that merely pass "close" to the datapoints.) This situation is illustrated in Figure 1.1(b). The dashed lines show sample functions which are consistent with $\mathcal{D}$, and the solid line depicts the mean value of such functions. Notice how the uncertainty is reduced close to the observations. The combination of the prior and the data leads to the *posterior* distribution over functions.

functions that agree with observations

posterior over functions

If more datapoints were added one would see the mean function adjust itself to pass through these points, and that the posterior uncertainty would reduce close to the observations. Notice, that since the Gaussian process is not a parametric model, we do not have to worry about whether it is possible for the model to fit the data (as would be the case if e.g. you tried a linear model on strongly non-linear data). Even when a lot of observations have been added, there may still be some flexibility left in the functions. One way to imagine the reduction of flexibility in the distribution of functions as the data arrives is to draw many random functions from the prior, and reject the ones which do not agree with the observations. While this is a perfectly valid way to do inference, it is impractical for most purposes—the exact analytical computations required to quantify these properties will be detailed in the next chapter.

non-parametric

inference

prior specification

The specification of the prior is important, because it fixes the properties of the functions considered for inference. Above we briefly touched on the mean and pointwise variance of the functions. However, other characteristics can also be specified and manipulated. Note that the functions in Figure 1.1(a) are smooth and stationary (informally, stationarity means that the functions look similar at all x locations). These are properties which are induced by the *covariance function* of the Gaussian process; many other covariance functions are possible. Suppose, that for a particular application, we think that the functions in Figure 1.1(a) vary too rapidly (i.e. that their characteristic length-scale is too short). Slower variation is achieved by simply adjusting parameters of the covariance function. The problem of *learning* in Gaussian processes is exactly the problem of finding suitable properties for the covariance function. Note, that this gives us a model of the data, and characteristics (such a smoothness, characteristic length-scale, etc.) which we can *interpret*.

covariance function

modelling and interpreting

classification

We now turn to the *classification* case, and consider the binary (or two-class) classification problem. An example of this is classifying objects detected in astronomical sky surveys into stars or galaxies. Our data has the label $+1$ for stars and -1 for galaxies, and our task will be to predict $\pi(\mathbf{x})$, the probability that an example with input vector $\mathbf{x}$ is a star, using as inputs some features that describe each object. Obviously $\pi(\mathbf{x})$ should lie in the interval $[0, 1]$. A Gaussian process prior over functions does not restrict the output to lie in this interval, as can be seen from Figure 1.1(a). The approach that we shall adopt is to squash the prior function f pointwise through a response function which restricts the output to lie in $[0, 1]$. A common choice for this function is the logistic function $\lambda(z) = (1 + \exp(-z))^{-1}$, illustrated in Figure 1.2(b). Thus the prior over f induces a prior over probabilistic classifications π.

squashing function

This set up is illustrated in Figure 1.2 for a 2-d input space. In panel (a) we see a sample drawn from the prior over functions f which is squashed through the logistic function (panel (b)). A dataset is shown in panel (c), where the white and black circles denote classes $+1$ and -1 respectively. As in the regression case the effect of the data is to downweight in the posterior those functions that are incompatible with the data. A contour plot of the posterior mean for $\pi(\mathbf{x})$ is shown in panel (d). In this example we have chosen a short characteristic length-scale for the process so that it can vary fairly rapidly; in

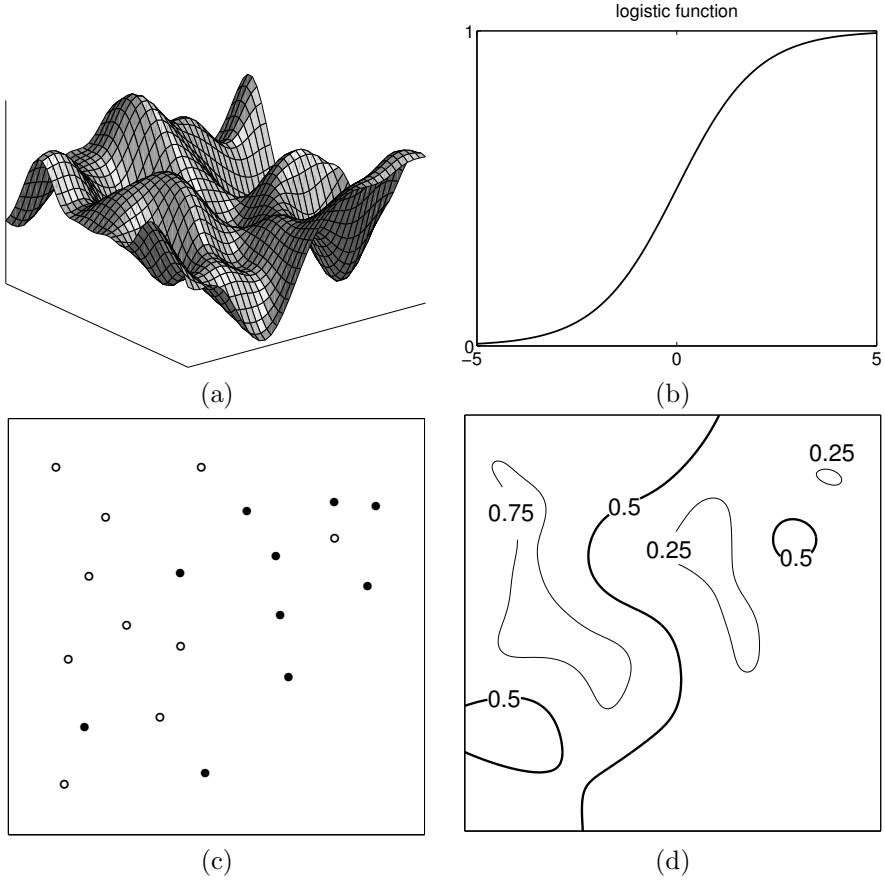


Figure 1.2: Panel (a) shows a sample from prior distribution on f in a 2-d input space. Panel (b) is a plot of the logistic function $\lambda(z)$. Panel (c) shows the location of the data points, where the open circles denote the class label $+1$, and closed circles denote the class label -1. Panel (d) shows a contour plot of the mean predictive probability as a function of $\mathbf{x}$; the decision boundaries between the two classes are shown by the thicker lines.

this case notice that all of the training points are correctly classified, including the two "outliers" in the NE and SW corners. By choosing a different length-scale we can change this behaviour, as illustrated in section 3.7.1.

1.2 Roadmap

The book has a natural split into two parts, with the chapters up to and including chapter 5 covering core material, and the remaining chapters covering the connections to other methods, fast approximations, and more specialized properties. Some sections are marked by an asterisk. These sections may be omitted on a first reading, and are not pre-requisites for later (un-starred) material.

regression

Chapter 2 contains the definition of Gaussian processes, in particular for the use in regression. It also discusses the computations needed to make predictions for regression. Under the assumption of Gaussian observation noise the computations needed to make predictions are tractable and are dominated by the inversion of a $n \times n$ matrix. In a short experimental section, the Gaussian process model is applied to a robotics task.

classification

Chapter 3 considers the classification problem for both binary and multi-class cases. The use of a non-linear response function means that exact computation of the predictions is no longer possible analytically. We discuss a number of approximation schemes, include detailed algorithms for their implementation and discuss some experimental comparisons.

covariance functions

As discussed above, the key factor that controls the properties of a Gaussian process is the covariance function. Much of the work on machine learning so far, has used a very limited set of covariance functions, possibly limiting the power of the resulting models. In chapter 4 we discuss a number of valid covariance functions and their properties and provide some guidelines on how to combine covariance functions into new ones, tailored to specific needs.

learning

Many covariance functions have adjustable parameters, such as the characteristic length-scale and variance illustrated in Figure 1.1. Chapter 5 describes how such parameters can be inferred or learned from the data, based on either Bayesian methods (using the marginal likelihood) or methods of cross-validation. Explicit algorithms are provided for some schemes, and some simple practical examples are demonstrated.

connections

Gaussian process predictors are an example of a class of methods known as kernel machines; they are distinguished by the probabilistic viewpoint taken. In chapter 6 we discuss other kernel machines such as support vector machines (SVMs), splines, least-squares classifiers and relevance vector machines (RVMs), and their relationships to Gaussian process prediction.

theory

In chapter 7 we discuss a number of more theoretical issues relating to Gaussian process methods including asymptotic analysis, average-case learning curves and the PAC-Bayesian framework.

fast approximations

One issue with Gaussian process prediction methods is that their basic complexity is $\mathcal{O}(n^3)$, due to the inversion of a $n \times n$ matrix. For large datasets this is prohibitive (in both time and space) and so a number of approximation methods have been developed, as described in chapter 8.

The main focus of the book is on the core supervised learning problems of regression and classification. In chapter 9 we discuss some rather less standard settings that GPs have been used in, and complete the main part of the book with some conclusions.

Appendix A gives some mathematical background, while Appendix B deals specifically with Gaussian Markov processes. Appendix C gives details of how to access the data and programs that were used to make the some of the figures and run the experiments described in the book.