Chapter 1

Introduction

"All models are wrong, but yours are stupid too."

@ML_Hipster (2013)

Prediction, extrapolation, and induction are all examples of learning a function from data. There are many ways to learn functions, but one particularly elegant way is by probabilistic inference. This type of method takes a group of hypotheses (a *model*), then weights those hypotheses based on how well their predictions match the data. Keeping around all the hypotheses that match the data helps guard against over-fitting. Comparing different models is a way to find which sorts of structure are present in a dataset.

To be able to learn a wide variety of structures, we would like to have an expressive language of models of functions. We would like to be able to represent simple kinds of functions, such as linear functions or polynomials. We would also like to have models of arbitrarily complex functions, specified in terms of high-level properties such as how smooth they are, whether they repeat over time, or which symmetries they have.

The beginning of thesis will show how to build such a language using Gaussian processes (GPs), a tractable class of models of functions. This chapter will introduce the basic properties of GPs. The next chapter will describe how to model different types of structure using GPs.

1.1 Gaussian Process models

Gaussian processes are a simple and general class of models of functions. To be precise, a GP is any distribution over functions such that any finite subset of function values
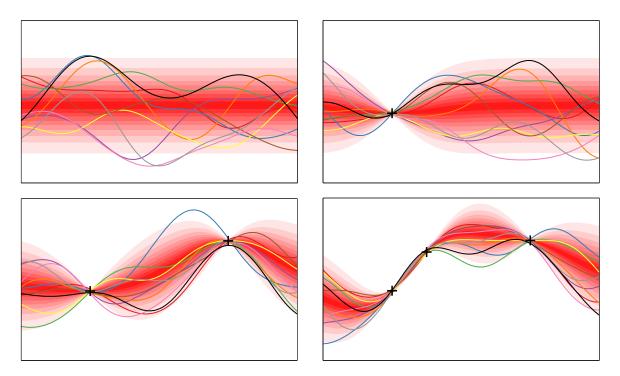
Figure 1.1: A visual representation of a Gaussian process modeling a one-dimensional function. Different shades of red correspond to deciles of the predictive density at each input location. Coloured lines show samples from the process – examples of some of the hypotheses included in the model. *Top left:* A GP not conditioned on any datapoints. *Remaining plots:* The posterior after conditioning on different amounts of data. All plots have the same axes.

$f(\mathbf{x}_1), f(\mathbf{x}_2), \dots f(\mathbf{x}_N)$ have a joint Gaussian distribution (Rasmussen and Williams, 2006, chapter 2). A GP model, before conditioning on data, is completely specified by its mean function,

$$\mathbb{E}\left[f(\mathbf{x})\right] = \mu(\mathbf{x}) \tag{1.1}$$

and its covariance function, also called the *kernel*:

$$\mathrm{Cov}\left[f(\mathbf{x}), f(\mathbf{x}')\right] = k(\mathbf{x}, \mathbf{x}') \tag{1.2}$$

It is common practice to assume that the mean function is simply zero everywhere, since uncertainty about the mean function can be taken into account by adding an extra term to the kernel.

After accounting for the mean, the kind of structure which can be captured by a

GP model is entirely determined by its kernel. The kernel determines how the model generalizes, or extrapolates to new data.

There are many possible choices of covariance function, and we can specify a wide range of models just by specifying the kernel of a GP. For example, linear regression, splines, and Kalman filters are all examples of GPs with particular kernels. However, these are just a few familiar examples out of a wide range of possibilities. One of the main difficulties in using GPs is constructing a kernel which represents the particular structure present in the data being modelled.

1.1.1   Model selection

The crucial property of GPs that allows us to automatically construct models is that we can compute the *marginal likelihood* of a dataset given a particular model, also known as the *evidence* (MacKay, 1992). The marginal likelihood allows us to compare models and automatically discover the appropriate amount of detail to use, due to Bayesian Occam's razor (MacKay, 2003; Rasmussen and Ghahramani, 2001).

Choosing a kernel, or kernel parameters, by maximizing the marginal likelihood will typically result in selecting the *least* flexible model which still captures all the structure in the data. For example, if a kernel has a parameter which controls the smoothness of the functions it models, the maximum-likelihood estimate of that parameter will usually correspond to the smoothest family of functions consistent with the observed function values.

Here is the marginal likelihood under a GP prior of observing a set of function values $[f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots f(\mathbf{x}_N)] := \boldsymbol{f}(\mathbf{X})$ at locations $\mathbf{X}$:

$$
\begin{aligned}
p(\boldsymbol{f}(\mathbf{X})|\mathbf{X}, \mu(\cdot), k(\cdot, \cdot)) &= \mathcal{N}(\boldsymbol{f}(\mathbf{X})|\mu(\mathbf{X}), k(\mathbf{X}, \mathbf{X})) \\
&= (2\pi)^{-\frac{N}{2}} \times \underbrace{|k(\mathbf{X}, \mathbf{X})|^{-\frac{1}{2}}}_{\text{discourages flexibility}} \\
&\quad \times \underbrace{\exp\left\{-\frac{1}{2}\left(\boldsymbol{f}(\mathbf{X}) - \boldsymbol{\mu}(\mathbf{X})\right)^{\mathsf{T}} k(\mathbf{X}, \mathbf{X})^{-1} \left(\boldsymbol{f}(\mathbf{X}) - \boldsymbol{\mu}(\mathbf{X})\right)\right\}}_{\text{encourages fit with data}}
\end{aligned}
\tag{1.3}
$$

This multivariate Gaussian density is referred to as the *marginal* likelihood because it implicitly integrates (marginalizes) over all possible functions values $\boldsymbol{f}(\bar{\mathbf{X}})$, where $\bar{\mathbf{X}}$ is the set of all locations where we have not observed the function.

1.1.2   Prediction

We can ask the model which function values are likely to occur at any location, given the observations seen so far. By the formula for Gaussian conditionals (given in appendix **??**), the predictive distribution of a function value $f(\mathbf{x}^\star)$ at a test point $\mathbf{x}^\star$ has the form:

$$p(f(\mathbf{x}^\star)|\boldsymbol{f}(\mathbf{X}),\mathbf{X},\mu(\cdot),k(\cdot,\cdot)) = \mathcal{N}\Big(f(\mathbf{x}^\star)\,|\,\underbrace{\mu(\mathbf{x}^\star) + k(\mathbf{x}^\star,\mathbf{X})k(\mathbf{X},\mathbf{X})^{-1}\left(\boldsymbol{f}(\mathbf{X}) - \mu(\mathbf{X})\right)}_{\text{predictive mean follows observations}},$$
$$\underbrace{k(\mathbf{x}^\star,\mathbf{x}^\star) - k(\mathbf{x}^\star,\mathbf{X})k(\mathbf{X},\mathbf{X})^{-1}k(\mathbf{X},\mathbf{x}^\star)}_{\text{predictive variance shrinks given more data}}\Big)$$

$$(1.4)$$

These expressions may look complex, but only require a few matrix operations to evaluate.

Sampling a function from a GP is also straightforward: a sample from a GP at a finite set of locations is just a single sample from a single multivariate Gaussian distribution, given by equation (1.4). Figure 1.1 shows prior and posterior samples from a GP, as well as contours of the predictive density.

Our use of probabilities does not mean that we are assuming the function being learned is stochastic or random in any way; it is simply a consistent method of keeping track of uncertainty.

1.1.3   Useful properties of Gaussian processes

There are several reasons why GPs in particular are well-suited for building a language of regression models:

- **Analytic inference.** Given a kernel function and some observations, the predictive posterior distribution can be computed exactly in closed form. This is a rare property for nonparametric models to have.

- **Expressivity.** Through the choice of covariance function, we can express a wide range of modeling assumptions. Some examples will be shown in section 1.2.

- **Integration over hypotheses.** The fact that a GP posterior, given a fixed kernel, lets us integrate exactly over a wide range of hypotheses means that overfitting is less of an issue than in comparable model classes. For example, compared to

neural networks, relatively few parameters need to be estimated, which lessens the need for the complex optimization or regularization schemes.

- **Model Selection.** A side benefit of being able to integrate over all hypotheses is that we can compute the marginal likelihood of the data given a model. This gives us a principled way of comparing different models.

- **Closed-form predictive distribution.** The predictive distribution of a GP at a set of test points is simply a multivariate Gaussian distribution. This means that GPs can easily be composed with other models or decision procedures.

- **Easy to analyze.** It may seem unsatisfying to restrict ourselves to a limited model class, as opposed to trying to do inference in the set of all computable functions. However, simple models can be used as well-understood building blocks for constructing more interesting models.

  For example, consider linear models. Although they form an extremely limited model class, they are simple, easy to analyze, and easy to incorporate into other models or procedures. Gaussian processes can be seen as an extension of linear models which retain these attractive properties (Rasmussen and Williams, 2006, chapter 2).

1.1.4 Limitations of Gaussian processes

There are several issues which make GPs sometimes difficult to use:

- **Slow inference.** Computing the matrix inverse in equations (1.3) and (1.4) takes $\mathcal{O}(N^3)$ time, making exact inference prohibitively slow for more than a few thousand datapoints. However, this problem can be addressed by approximate inference schemes (Hensman et al., 2013; Quiñonero-Candela and Rasmussen, 2005; Snelson and Ghahramani, 2006).

- **Light tails of the predictive distribution.** The predictive distribution of a standard GP model is Gaussian. We may sometimes with to use non-Gaussian predictive likelihoods, for example in order to be robust to outliers, or to perform classification. Using non-Gaussian likelihoods requires approximate inference. Fortunately, mature software packages exist (Hensman et al., 2014; Rasmussen and

Nickisch, 2010; Vanhatalo et al., 2013) which can automatically perform approximate inference for a wide variety of non-Gaussian likelihoods, as well as using sparse approximations.

- **The need to choose a kernel.** The flexibility of GP models raises the question of which kernel to use for a given problem. Choosing a useful kernel is equivalent to learning a useful representation of the input. Kernel parameters can be set automatically by maximizing the marginal likelihood, but until recently, human experts were required to choose the parametric form of the kernel. Section 1.2 will show a way in which kernels can be automatically constructed for a given dataset.

1.2   Outline and contributions of thesis

The main contribution of this thesis is to develop a method to automatically model, visualize, and describe structure in data, by searching through an open-ended language of regression models. This thesis also includes a set of related results showing how Gaussian processes can be extended or composed with other models.

Chapter 1.2 is a tutorial showing how to build a wide variety of structured models of functions by constructing appropriate covariance functions. We will also show how GPs can produce nonparametric models of manifolds with diverse topological structures, such as cylinders, toruses and Möbius strips.

Chapter 1.2 shows how to search over an open-ended language of models, built by adding and multiplyin different kernels. Since we can evaluate each model by the marginal likelihood, we can automatically construct custom models for each dataset by a simple breadth-first search. The nature of GPs allow the resulting models to be visualized by decomposing them into diverse, interpretable components, each capturing a different type of structure. Capturing this high-level structure sometimes allows us to extrapolate beyond the range of the data.

One benefit of using a compositional model class is that the resulting models are relatively interpretable. Chapter 1.2 demonstrates a system which automatically describes the structure implied by a given kernel on a given dataset, generating reports with graphs and English-language text describing the resulting model. Combined with the automatic model search developed in chapter 1.2, this system represents the beginnings of what could be called an "automatic statistician".

Chapter 1.2 analyzes deep neural network models by characterizing the prior over

functions obtained by composing GP priors to form *deep Gaussian processes*. We show that, as the number of layers increase, the amount of information retained about the original input diminishes to a single degree of freedom. A simple change to the network architecture fixes this pathology. We relate these models to neural networks, and as a side effect derive several forms of *infinitely deep kernels*.

Chapter 1.2 examines a more limited, but much faster way of discovering structure using GPs. Specifying a kernel with many different types of structure, we use kernel parameters to discard whichever types of structure are *not* found in the current dataset. The model class we examine is called *additive Gaussian processes*, a model summing over exponentially-many GPs, each depending on a different subset of the input variables. We give a polynomial-time inference algorithm for this model, and relate it to other model classes. For example, additive GPs are shown to have the same covariance as a GP that uses *dropout*, a recently developed regularization technique for neural networks.

Chapter 1.2 develops a Bayesian clustering model in which the clusters have non-parametric shapes, called the infinite warped mixture model. The density manifolds learned by this model follow the contours of the data density, and have interpretable, parametric forms in the latent space. The marginal likelihood lets us infer the effective dimension and shape of each cluster separately, as well as the number of clusters.

References

James Hensman, Nicolo Fusi, and Neil D. Lawrence. Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*, 2013. (page 5)

James Hensman, Nicolo Fusi, Ricardo Andrade, Nicolas Durrande, Alan Saul, Max Zwiessele, and Neil D. Lawrence. GPy: A Gaussian process framework in Python, 2014. https://github.com/SheffieldML/GPy. (page 5)

David J.C. MacKay. *Bayesian methods for adaptive models*. PhD thesis, California Institute of Technology, 1992. (page 3)

David J.C. MacKay. *Information theory, inference, and learning algorithms*. Cambridge University press, 2003. (page 3)

@ML_Hipster. "...essentially, all models are wrong, but yours are stupid too." – G.E.P. Box in a less than magnanimous mood., 2013. URL https://twitter.com/ML_Hipster/status/394577463990181888. (page 1)

Joaquin Quiñonero-Candela and Carl E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6: 1939–1959, 2005. (page 5)

Carl E. Rasmussen and Zoubin Ghahramani. Occam's razor. *Advances in Neural Information Processing Systems*, pages 294–300, 2001. (page 3)

Carl E. Rasmussen and Hannes Nickisch. Gaussian processes for machine learning (GPML) toolbox. *Journal of Machine Learning Research*, 11:3011–3015, December 2010. (page 5)

Carl E. Rasmussen and Christopher K.I. Williams. *Gaussian Processes for Machine Learning*, volume 38. The MIT Press, Cambridge, MA, USA, 2006. (pages 2 and 5)

Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. *Advances in Neural Information Processing Systems*, 2006. (page 5)

Jarno Vanhatalo, Jaakko Riihimäki, Jouni Hartikainen, Pasi Jylänki, Ville Tolvanen, and Aki Vehtari. GPstuff: Bayesian modeling with Gaussian processes. *Journal of Machine Learning Research*, 14(1):1175–1179, 2013. (page 6)