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 nice way is by *inference*. Inference procedures set up a group of hypotheses – a *model*, then weight 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. We can also compare models to find what sorts of structure are present in a dataset.

To be able to learn a wide variety of types of structure, we'd like to have an expressive language of models of functions. We'd like to be able to represent simple kinds of functions, such as linear or polynomial ones. We'd 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.

This thesis will show how to build such a language using Gaussian processes (GPs), a tractable set of models of very different types of functions. This chapter will introduce the basic properties of GPs. The next chapter will describe the many types of functions that we know how to model 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 $f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots f(\mathbf{x}_N)$ have a joint Gaussian distribution (Rasmussen and Williams,

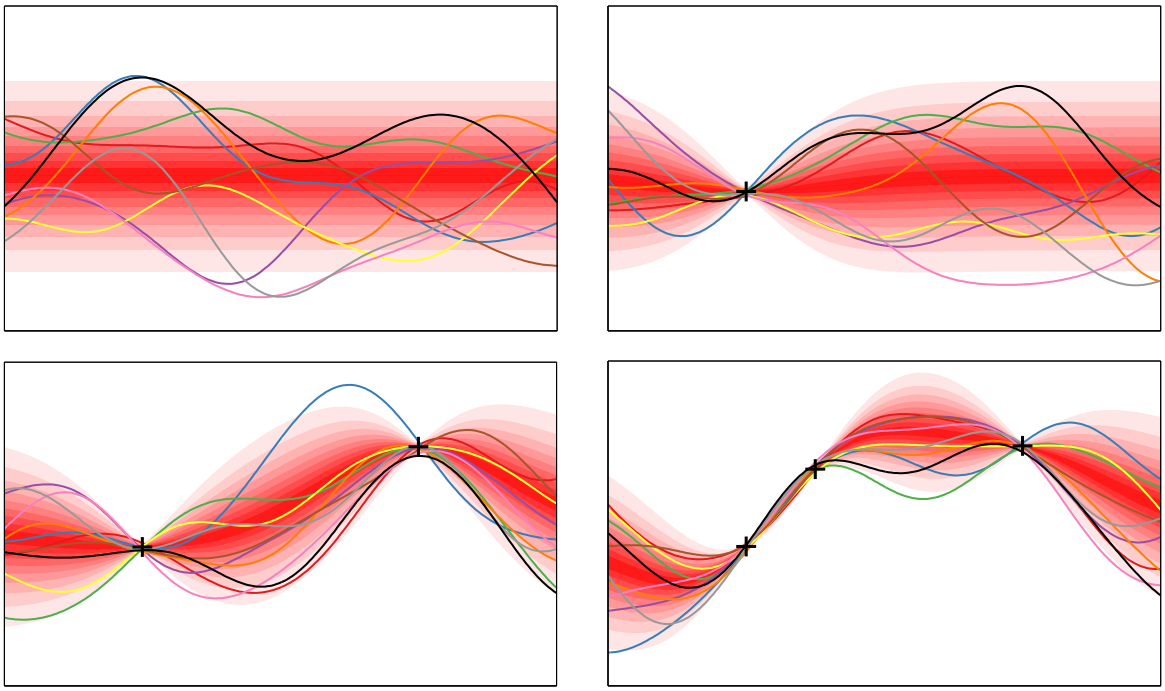

Fig. 1.1 A visual representation of a one-dimensional Gaussian process posterior. Different shades of red correspond to deciles of the predictive density at each input location. Coloured lines show samples from the process. *Top left:* A GP not conditioned on any datapoints. *Remaining plots:* The posterior after conditioning on different amounts of data.

2006). 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*:

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

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 model classes barely scratch the surface of the many possible models we can express through choosing a kernel. 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 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 select the *least* flexible model class 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 possible family of functions which still go through all the observed function values.

Here's the marginal likelihood under a GP of observing a set of function values $[f(\mathbf{x}_1), f(\mathbf{x}_2), \dots f(\mathbf{x}_N)] = \boldsymbol{f}(\mathbf{X})$ at locations $[\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]^\mathsf{T} = \mathbf{X}$:

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

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 haven't observed the function.

1.1.2 Prediction

Even though we don't need to consider locations other than at the data when computing the marginal likelihood, we can still ask the model which function values are likely to occur at any location, given the observations we've seen. By the formula for Gaussian conditionals (described in appendix **??**), the predictive distribution of a function value $f(\mathbf{x}^\star)$ at a test point $\mathbf{x}^\star$ has a simple form:

$$p(f(\mathbf{x}^\star)|\boldsymbol{f}(\mathbf{X}), \mathbf{X}, \mu(\cdot), k(\cdot,\cdot)) = \mathcal{N}\Big(f(\mathbf{x}^\star) \,|\, \underbrace{\mu(\mathbf{x}^\star) + k(\mathbf{x}^\star, \mathbf{X})k(\mathbf{X}, \mathbf{X})^{-1}\left(\boldsymbol{f}(\mathbf{X}) - \mu(\mathbf{X})\right)}_{\text{predictive mean goes through observations}}, \underbrace{k(\mathbf{x}^\star, \mathbf{x}^\star) - k(\mathbf{x}^\star, \mathbf{X})k(\mathbf{X}, \mathbf{X})^{-1}k(\mathbf{X}, \mathbf{x}^\star)}_{\text{predictive variance shrinks given more data}}\Big) \tag{1.4}$$

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 representation of uncertainty through 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 our 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.** By choosing different covariance functions, we can express a very wide range of modeling assumptions.
- **Integration over hypotheses.** The fact that a GP posterior lets us exactly integrate over a wide range of hypotheses means that overfitting is less of an

issue than in comparable model classes, such as neural networks. It also removes the need for sophisticated optimization schemes. In contrast, much of the neural network literature is devoted to techniques for regularization and optimization.

- **Marginal likelihood.** A side benefit of being able to integrate over all hypotheses is that we can compute the *marginal likelihood* of the data given the 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 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 fast, simple, and easy to analyze, and easy to incorporate into other models or procedures. Gaussian processes can be seen as an extension of linear models (Rasmussen and Williams, 2006) which retain these attractive properties.

1.1.4 Limitations of Gaussian Processes

There are, unfortunately, several issues which make usage of GPs sometimes difficult:

- **Slow inference.** Computing the matrix inverse in equations (1.3) and (1.4) takes $\mathcal{O}(N^3)$ time, making inference slow for more than about 1000 datapoints. However, this problem can be addressed by approximate inference schemes (Hensman et al., 2013; Quiñonero-Candela and Rasmussen, 2005; Snelson and Ghahramani, 2006). Most GP software packages implement several of these methods (Rasmussen and Nickisch, 2010; Vanhatalo et al., 2014).

- **Light tails of the predictive distribution.** The predictive distribution of a standard GP model is Gaussian. In order to be robust to outliers, or to perform classification, we may wish to use non-Gaussian noise models. Fortunately, mature software packages exist which can automatically perform approximate inference for a wide variety of non-Gaussian likelihoods.

- **The need to choose a kernel.** In practice, the extreme flexibility of GP models means that we are also faced with the difficult task of choosing a kernel. In fact, choosing a useful kernel is equivalent to the problem of learning a good representation of the input. Kernel parameters are usually set automatically by maximizing the marginal likelihood. However, until recently, human experts were still required to choose the parametric form of the kernel from a small set of standard kernels. Section 1.2 will show how the entire construction of kernels can be automated.

1.2 Outline and Contributions of Thesis

The main contribution of this thesis is to show how to automate the discovery and explanation of structure in functions, simply by searching an open-ended language of regression models. It 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'll also show how GPs can produce nonparametric models of manifolds, diverse topological structures, such as cylinders, toruses and Möbius strips.

Chapter 1.2 shows how to search over a general, open-ended language of models, built by composing together different kernels. Since we can evaluate each model by its marginal likelihood, we can automatically construct custom models for each dataset by a simple 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 even allows us to extrapolate beyond the range of the data.

One benefit of using a compositional model class is that the resulting models are 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. We'll show several automatic analyses of time-series. Combined with the automatic model search developed in chapter 1.2, this system represents the beginnings of an "automatic statistician".

Chapter 1.2 analyzes deep 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 in such models increases, 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 different 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 *aren't* 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 class, 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 discovered regularization technique for neural networks.

Chapter 1.2 develops a Bayesian clustering model in which the clusters have non-parametric shapes - 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

David Duvenaud, Hannes Nickisch, and Carl Edward Rasmussen. Additive Gaussian processes. In *Advances in Neural Information Processing Systems 24*, pages 226–234, Granada, Spain, 2011.

David Duvenaud, James Robert Lloyd, Roger Grosse, Joshua B. Tenenbaum, and Zoubin Ghahramani. Structure discovery in nonparametric regression through compositional kernel search. In *Proceedings of the 30th International Conference on Machine Learning*, June 2013.

David Duvenaud, Oren Rippel, Ryan P. Adams, and Zoubin Ghahramani. Avoiding pathologies in very deep networks. Reykjavik, Iceland, April 2014. URL `http://arxiv.org/pdf/1402.5836.pdf`.

Roger B. Grosse, Ruslan Salakhutdinov, William T. Freeman, and Joshua B. Tenenbaum. Exploiting compositionality to explore a large space of model structures. In *Uncertainty in Artificial Intelligence*, 2012.

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

Tomoharu Iwata, David Duvenaud, and Zoubin Ghahramani. Warped mixtures for nonparametric cluster shapes. Bellevue, Washington, July 2013. URL `http://arxiv.org/pdf/1206.1846`.

James Robert Lloyd, David Duvenaud, Roger Grosse, Joshua B. Tenenbaum, and Zoubin Ghahramani. Automatic construction and natural-language description of nonparametric regression models. Technical Report arXiv:1402.4304 [stat.ML], 2014.

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

David JC 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)

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

Carl Edward Rasmussen and Zoubin Ghahramani. Occam's razor. *Advances in neural information processing systems*, pages 294–300, 2001. (page 3)

Carl Edward Rasmussen and Hannes Nickisch. Gaussian processes for machine learning (gpml) toolbox. *J. Mach. Learn. Res.*, 11:3011–3015, December 2010. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1756006.1953029. (page 5)

C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*, volume 38. The MIT Press, Cambridge, MA, USA, 2006. (pages 1 and 5)

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

Jarno Vanhatalo, Jaakko Riihimaki, Jouni Hartikainen, Pasi Jylanki, Ville Tolvanen, and Aki Vehtari. Gpstuff, 2014. http://mloss.org/software/view/451/. (page 5)