Chapter 1

Introduction

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

@ML_Hipster (2013)

Prediction, extrapolation, and induction can all be expressed as learning a function from data. A general recipe for learning from data is to perform *inference* - to choose a hyopthesis or to weight a set of hypotheses, based on how compatible they are with the data. To do inference, we start with a weighted set of hypotheses - a *model.* To be able to learn a wide variety of types of functions, we'd like to have an expressive language of models of functions, which can represent both simple parametric functions, such as linear or polynomial, and also complex nonparametric functions specified in terms of properties such as smoothness or periodicity. Fortunately, Gaussian processes (GPs) provide a very general and analytically tractable way of learning many different classes of functions. This chapter will introduce the basic properties of GPs.

Once we have a rich enough language for expressing functions, the remaining question becomes: Which particular model will work well on my problem? What sort of structure should I put in my model? 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 nonparametric models of functions. To be precise, a GP is any distribution over functions such that any finite subset of function evaluations $f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots f(\mathbf{x}_N)$, have a joint Gaussian distribution (Rasmussen and Williams, 2006). A GP model, before conditioning on data, is completely specified

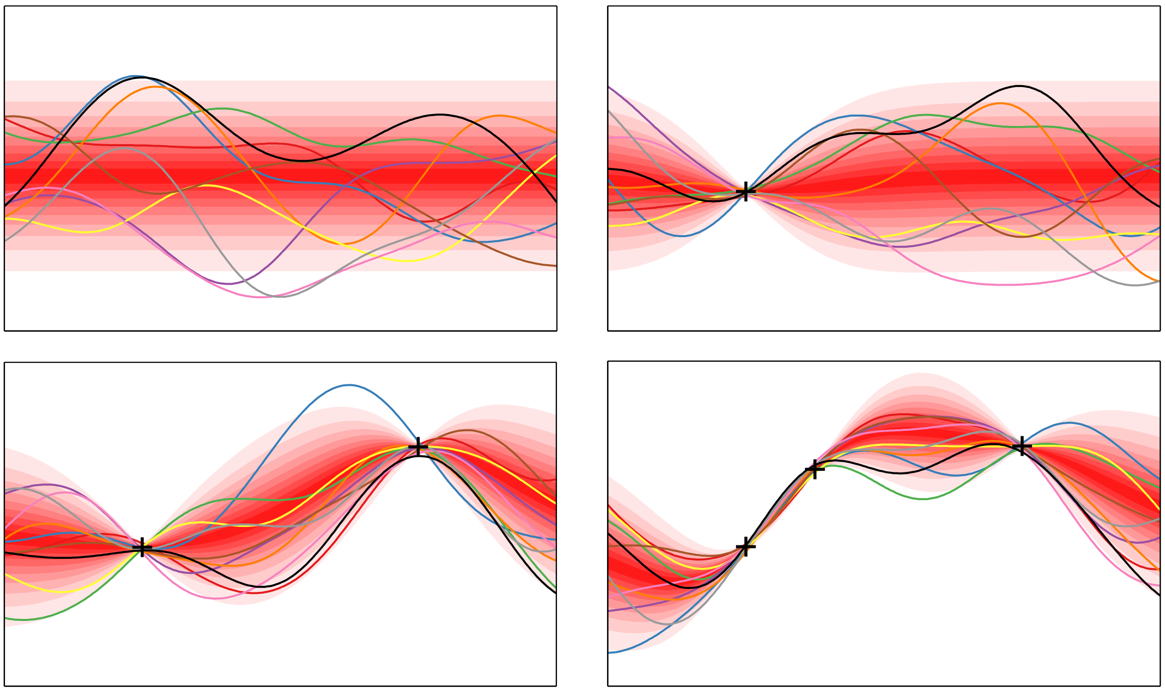

Fig. 1.1 A visual representation of a one-dimensional Gaussian process posterior. Different shades of red correspond to deciles of the predictive density at each input location. Coloured lines show samples from the process. Top left: A GP not conditioned on any datapoints. The remaining plots show the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

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 mean function is simply zero everywhere, since uncertainty about the mean function can be equivalently expressed by adding an extra term to the kernel. Also equivalent is to subtract the mean function from the observations, and then add it back when making predictions.

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 GPs with particular kernels. However, these model classes barely scratch the surface of the wide variety of possibilites. One of the main difficulties in using GPs is constructing a kernel which represents the particular structure present in the data being modeled.

1.1.1 Model Selection

The crucial property of GPs that allows us to automatically construct models is that we can compute their *marginal likelihood.* 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.

To be concrete, here's the marginal likelihood under a GP 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 given by the rows of $\mathbf{X}$:

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

This Gaussian likelihood is referred to as the *marginal* likelihood because it implicitly integrates over all possible functions values $\boldsymbol{f}(\bar{\mathbf{X}})$, where $\bar{\mathbf{X}}$ is the set of all locations where we don't have any observations.

1.1.2 Prediction

Even though we don't need to consider any other locations when computing the likelihood, we can still ask the model which function values are likely to occur at any location, given the observations we've seen. The predictive distribution at a test point $\mathbf{x}^\star$ has a simple form:

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

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

Sampling from a GP is also straightforward: a sample from a GP 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.

Our representation of uncertainty through probabilites 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

- **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, without the need for sampling procedures.

- **Easy to Analyze** It may seem unsatisfying to restrict ourselves to a limited model class, instead of some more flexible model class, such as the set of all computible functions. However, simple models can be used as well-understood building blocks for constructing more interesting models in diverse settings.

  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 which retain these attractive properties.

1.1.4 Limitations of Gaussian Processes

- **Slow inference** Computing the matrix inverse in (1.3) and (1.4) takes $\mathcal{O}(N^3)$ time. This problem can now be addressed by approximate inference schemes, [TODO: Citations] and most GP software packages implement several of these.

- **Light tails** We may wish to use non-Gaussian noise models, for instance in order to be robust to outliers, to perform classification, or for some other form of structured prediction. Using non-Gaussian noise models requires approximate inference schemes. Fortunately, mature software packages exist which can automatically perform approximate inference for a wide variety of 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 reprentation of the input. Typically, human experts choose from among a small set of standard kernels. In this thesis, we show how the construction and selection of useful kernels can be done automatically.

1.2 Outline and Contributions of Thesis

This thesis presents a set of related results showing how the probabilistic nature of Gaussian process models allows them to be easily extended or composed with other models. Furthermore, the fact that the marginal likelihood is available means that we can evaluate how much evidence the data provides for one structure over another, allowing an automatic search to construct models for us.

Chapter 1.3 contains an overview of many types of structured priors on functions that can be easily expressed by constructing appropriate covariance functions. We'll also see how GPs can be combined with latent variable models to produce models of nonparametric manifolds. By introducing structure into the kernels of those GPs, we can create manifolds with diverse topological structures, such as cylinders, torii and Möbius strips.

Chapter 1.3 shows how to construct a general, open-ended language over kernels - which implies a corresponding language over models. Given a wide variety of models, plus the ability to evaluate the suitability of each one, it's straightforward to automatically construct and search over models. Because of the structure of GPs, the resulting models can be decomposed into diverse, interpretable components, each capturing a different type of structure. Capturing high-level structure also allows us to extrapolate, rather than simply interpolating.

One benefit of using a relatively simple model class is that the resulting models are interpretable. **Chapter 1.3** 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 1.3, this system represents the beginnings of an "automatic statistician". We discuss the advantages and potential pitfalls of automating the modeling process in this way.

Chapter ?? 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 diminshes 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 1.3 examines a more limited, but much faster way of discovering structure using GPs, starting with many different types of structure in the kernel, and using kernel parameters to discard whichever types of structure *aren't* present in the current dataset. This model class is called *additive Gaussian processes*, a model summing over exponentially-many GPs, each depending on a different subset of the input variables. An polynomial-time algorithm for doing inference in this model class is given, and the resulting model class is characterized and related to existing model classes. This model class is also shown to have identical covariance to the model obtained by performing *dropout* in GPs, a recently discovered regularization technique for neural networks.

Chapter 1.3 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, but have interpretable, parametric forms in the latent space. The availability of the marginal likelihood allows us to infer the effective dimension of each manifold separately, as well as the warping function and the number of clusters.

1.3 Attribution

This thesis was made possible by the substantial contributions of the many co-authors I was fortunate to work with. In this section, I attempt to give proper credit to my tireless co-authors, who made this research enjoyable to produce.

Structure through kernels Section **??** of chapter 1.3, describing how kernel symmetries give rise to priors on manifolds with interesting topologies, is based on a collaboration with David Reshef, Roger Grosse, Josh Tenenbaum, and Zoubin Ghahramani.

Structure Search The research upon which Chapter 1.3 is based was done in collaboration with James Robert Lloyd, Roger Grosse, Joshua B. Tenenbaum, and Zoubin Ghahramani, and was published in (Duvenaud et al., 2013), where James Lloyd was joint first author. Myself, James Lloyd and Roger Grosse jointly developed the idea of searching through a grammar-based language of GP models, inspired by Grosse et al. (2012), and wrote the first versions of the code together. James Lloyd ran most of the experiments. I produced all of the figures. Carl Rasmussen, Zoubin Ghahramani and Josh Tenenbaum provided many conceptual insights, as well as suggestions about how the resulting procedure could be most fruitfully applied.

Automatic Statistician The work appearing in chapter 1.3 was written in collaboration with James Robert Lloyd, Roger Grosse, Joshua B. Tenenbaum, Zoubin Ghahramani, and was published in (Lloyd et al., 2014). The idea of the correspondence between kernels and adjectives grew out of discussions between James and myself. James Lloyd wrote most of the code to automatically generate reports, and ran all of the experiments. The text was written mainly by myself, James Lloyd, and Zoubin Ghahramani, with many helpful contributions and suggestions from Roger Grosse and Josh Tenenbaum.

Deep Gaussian Processes The ideas contained in chapter **??** were developed through discussions with Oren Rippel, Ryan Adams and Zoubin Ghahramani, and appear in (Duvenaud et al., 2014). The derivations, experiments and writing were done mainly by myself, with many helpful suggestions by my co-authors.

Additive Gaussian processes The work in chapter 1.3 was done in collaboration with Hannes Nickisch and Carl Rasmussen, who derived and coded up the initial model. My role in the project was to examine the properties of the resulting model, clarify the connections to existing methods, to create all figures and run all experiments. This work was published in (Duvenaud et al., 2011). The connection to dropout regularization was my own contribution.

Warped Mixtures The work comprising the bulk of chapter 1.3 was done in collaboration with Tomoharu Iwata and Zoubin Ghahramani, and appeared in (Iwata et al., 2013). Specifically, the main idea was borne out of a conversation between Tomo and myself, and together we wrote almost all of the code together as well as the paper. Tomo ran most of the experiments. Zoubin Ghahramani provided guidance and many helpful suggestions throughout the project.

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. (page 8)

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. (page 7)

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`. (page 8)

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. (page 7)

Tomoharu Iwata, David Duvenaud, and Zoubin Ghahramani. Warped mixtures for nonparametric cluster shapes. Bellevue, Washington, July 2013. URL `http://arxiv.org/pdf/1206.1846`. (page 8)

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. (page 8)

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)

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

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