Gaussian Processes for Regression: A Quick Introduction

M. Ebden, August 2008
Comments to mebden@gmail.com

1 Motivation

Figure 1 illustrates a typical example of a prediction problem: given some noisy observations of a dependent variable at certain values of the independent variable x, what is our best estimate of the dependent variable at a new value, x_*?

If we expect the underlying function $f(x)$ to be linear, and can make some assumptions about the input data, we might use a least-squares method to fit a straight line (linear regression). Moreover, if we suspect $f(x)$ may also be quadratic, cubic, or even nonpolynomial, we can use the principles of model selection to choose among the various possibilities.

Gaussian process regression (GPR) is an even finer approach than this. Rather than claiming $f(x)$ relates to some specific models (e.g. $f(x) = mx + c$), a Gaussian process can represent $f(x)$ obliquely, but rigorously, by letting the data 'speak' more clearly for themselves. GPR is still a form of supervised learning, but the training data are harnessed in a subtler way.

As such, GPR is a less 'parametric' tool. However, it's not completely free-form, and if we're unwilling to make even basic assumptions about $f(x)$, then more general techniques should be considered, including those underpinned by the principle of maximum entropy; Chapter 6 of Sivia and Skilling (2006) offers an introduction.

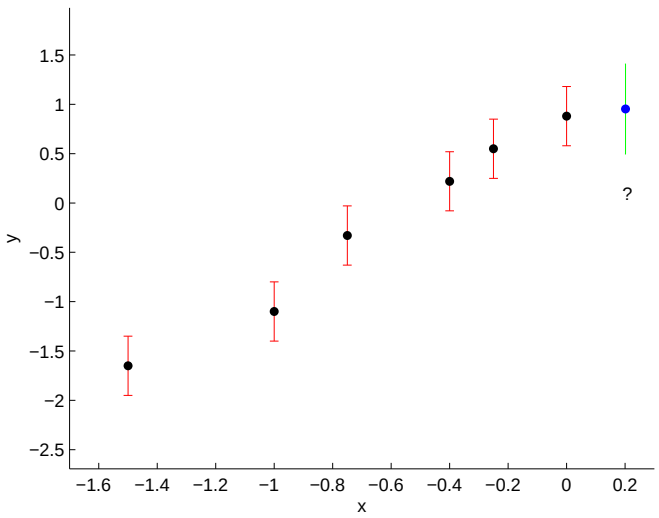

Figure 1: Given six noisy data points (error bars are indicated with vertical lines), we are interested in estimating a seventh at $x_* = 0.2$.

2 Definition of a Gaussian Process

Gaussian processes (GPs) extend multivariate Gaussian distributions to infinite dimensionality. Formally, a Gaussian process generates data located throughout some domain such that any finite subset of the range follows a multivariate Gaussian distribution. Now, the n observations in an arbitrary data set, $\mathbf{y} = \{y_1, \ldots, y_n\}$, can always be imagined as a single point sampled from some multivariate (n-variate) Gaussian distribution, after enough thought. Hence, working backwards, this data set can be partnered with a GP. Thus GPs are as universal as they are simple.

Very often, it's assumed that the mean of this partner GP is zero everywhere. What relates one observation to another in such cases is just the *covariance function*, $k(x, x')$. A popular choice is the 'squared exponential',

$$k(x, x') = \sigma_f^2 \exp\left[\frac{-(x - x')^2}{2l^2}\right], \tag{1}$$

where the maximum allowable covariance is defined as σ_f^2 — this should be high for functions which cover a broad range on the y axis. If $x \approx x'$, then $k(x, x')$ approaches this maximum, meaning $f(x)$ is nearly perfectly correlated with $f(x')$. This is good: for our function to look smooth, neighbours must be alike. Now if x is distant from x', we have instead $k(x, x') \approx 0$, i.e. the two points cannot 'see' each other. So, for example, during interpolation at new x values, distant observations will have negligible effect. How much effect this separation has will depend on the length parameter, l, so there is much flexibility built into (1).

Not quite enough flexibility though: the data are often noisy as well, from measurement errors and so on. Each observation y can be thought of as related to an underlying function $f(x)$ through a Gaussian noise model:

$$y = f(x) + \mathcal{N}(0, \sigma_n^2), \tag{2}$$

something which should look familiar to those who've done regression before. Regression is the search for $f(x)$. Purely for simplicity of exposition in the next page, we take the novel approach of folding the noise into $k(x, x')$, by writing

$$k(x, x') = \sigma_f^2 \exp\left[\frac{-(x - x')^2}{2l^2}\right] + \sigma_n^2 \delta(x, x'), \tag{3}$$

where $\delta(x, x')$ is the Kronecker delta function. (When most people use Gaussian processes, they keep σ_n separate from $k(x, x')$. However, our redefinition of $k(x, x')$ is equally suitable for working with problems of the sort posed in Figure 1. So, given n observations $\mathbf{y}$, our objective is to predict y_*, not the 'actual' f_*; their expected values are identical according to (2), but their variances differ owing to the observational noise process. e.g. in Figure 1, the expected value of y_*, and of f_*, is the dot at x_*.)

To prepare for GPR, we calculate the covariance function, (3), among all possible combinations of these points, summarizing our findings in three matrices:

$$K = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_n) \\ k(x_2, x_1) & k(x_2, x_2) & \cdots & k(x_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n, x_1) & k(x_n, x_2) & \cdots & k(x_n, x_n) \end{bmatrix} \tag{4}$$

$$K_* = \begin{bmatrix} k(x_*, x_1) & k(x_*, x_2) & \cdots & k(x_*, x_n) \end{bmatrix} \qquad K_{**} = k(x_*, x_*). \tag{5}$$

Confirm for yourself that the diagonal elements of K are $\sigma_f^2 + \sigma_n^2$, and that its extreme off-diagonal elements tend to zero when x spans a large enough domain.

3 How to Regress using Gaussian Processes

Since the key assumption in GP modelling is that our data can be represented as a sample from a multivariate Gaussian distribution, we have that

$$\begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K & K_*^{\mathrm{T}} \\ K_* & K_{**} \end{bmatrix}\right), \tag{6}$$

where T indicates matrix transposition. We are of course interested in the conditional probability $p(y_*|\mathbf{y})$: "given the data, how likely is a certain prediction for y_*?". As explained more slowly in the Appendix, the probability follows a Gaussian distribution:

$$y_*|\mathbf{y} \sim \mathcal{N}(K_* K^{-1}\mathbf{y},\ K_{**} - K_* K^{-1} K_*^{\mathrm{T}}). \tag{7}$$

Our best estimate for y_* is the mean of this distribution:

$$\overline{y}_* = K_* K^{-1}\mathbf{y}, \tag{8}$$

and the uncertainty in our estimate is captured in its variance:

$$\mathrm{var}(y_*) = K_{**} - K_* K^{-1} K_*^{\mathrm{T}}. \tag{9}$$

We're now ready to tackle the data in Figure 1.

1. There are $n = 6$ observations $\mathbf{y}$, at

   $$\mathbf{x} = \begin{bmatrix} -1.50 & -1.00 & -0.75 & -0.40 & -0.25 & 0.00 \end{bmatrix}.$$

   We know $\sigma_n = 0.3$ from the error bars. With judicious choices of σ_f and l (more on this later), we have enough to calculate a covariance matrix using (4):

   $$K = \begin{bmatrix} 1.70 & 1.42 & 1.21 & 0.87 & 0.72 & 0.51 \\ 1.42 & 1.70 & 1.56 & 1.34 & 1.21 & 0.97 \\ 1.21 & 1.56 & 1.70 & 1.51 & 1.42 & 1.21 \\ 0.87 & 1.34 & 1.51 & 1.70 & 1.59 & 1.48 \\ 0.72 & 1.21 & 1.42 & 1.59 & 1.70 & 1.56 \\ 0.51 & 0.97 & 1.21 & 1.48 & 1.56 & 1.70 \end{bmatrix}.$$

   From (5) we also have $K_{**} = 1.70$ and

   $$K_* = \begin{bmatrix} 0.38 & 0.79 & 1.03 & 1.35 & 1.46 & 1.58 \end{bmatrix}.$$

2. From (8) and (9), $\overline{y}_* = 0.95$ and $\mathrm{var}(y_*) = 0.21$.

3. Figure 1 shows a data point with a question mark underneath, representing the estimation of the dependent variable at $x_* = 0.2$.

We can repeat the above procedure for various other points spread over some portion of the x axis, as shown in Figure 2. (In fact, equivalently, we could avoid the repetition by performing the above procedure once with suitably larger K_* and K_{**} matrices. In this case, since there are 1,000 test points spread over the x axis, K_{**} would be of size 1,000 × 1,000.) Rather than plotting simple error bars, we've decided to plot $\overline{y}_* \pm 1.96\sqrt{\mathrm{var}(y_*)}$, giving a 95% confidence interval.

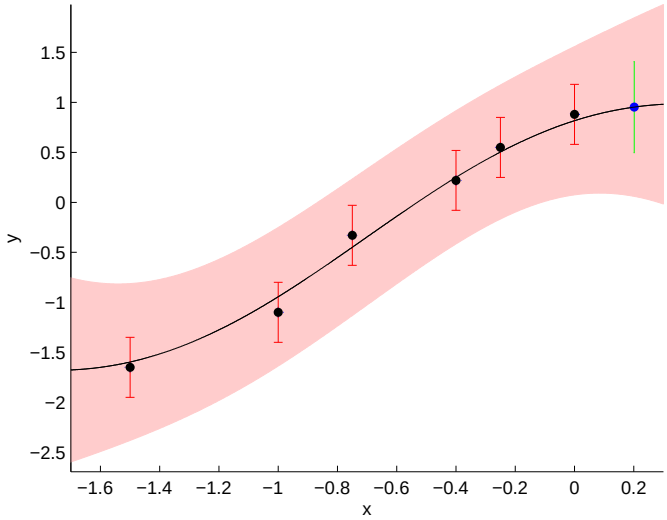

Figure 2: The solid line indicates an estimation of y_* for 1,000 values of x_*. Pointwise 95% confidence intervals are shaded.

4 GPR IN THE REAL WORLD

The reliability of our regression is dependent on how well we select the covariance function. Clearly if its parameters — call them $\boldsymbol{\theta} = \{l, \sigma_f, \sigma_n\}$ — are not chosen sensibly, the result is nonsense. Our maximum *a posteriori* estimate of $\boldsymbol{\theta}$ occurs when $p(\boldsymbol{\theta}|\mathbf{x}, \mathbf{y})$ is at its greatest. Bayes' theorem tells us that, assuming we have little prior knowledge about what $\boldsymbol{\theta}$ should be, this corresponds to maximizing $\log p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$, given by

$$\log p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^{\mathrm{T}} K^{-1}\mathbf{y} - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi. \tag{10}$$

Simply run your favourite multivariate optimization algorithm (e.g. conjugate gradients, Nelder-Mead simplex, etc.) on this equation and you've found a pretty good choice for $\boldsymbol{\theta}$; in our example, $l = 1$ and $\sigma_f = 1.27$.

It's only "pretty good" because, of course, Thomas Bayes is rolling in his grave. Why commend just one answer for $\boldsymbol{\theta}$, when you can integrate everything over the many different possible choices for $\boldsymbol{\theta}$? Chapter 5 of Rasmussen and Williams (2006) presents the equations necessary in this case.

Finally, if you feel you've grasped the toy problem in Figure 2, the next two examples handle more complicated cases. Figure 3(a), in addition to a long-term downward trend, has some fluctuations, so we might use a more sophisticated covariance function:

$$k(x, x') = \sigma_{f_1}^2 \exp\left[\frac{-(x-x')^2}{2l_1^2}\right] + \sigma_{f_2}^2 \exp\left[\frac{-(x-x')^2}{2l_2^2}\right] + \sigma_n^2 \delta(x, x'). \tag{11}$$

The first term takes into account the small vicissitudes of the dependent variable, and the second term has a longer length parameter ($l_2 \approx 6l_1$) to represent its long-term trend. Covariance functions can be grown in this way *ad infinitum*, to suit the complexity of your particular data.

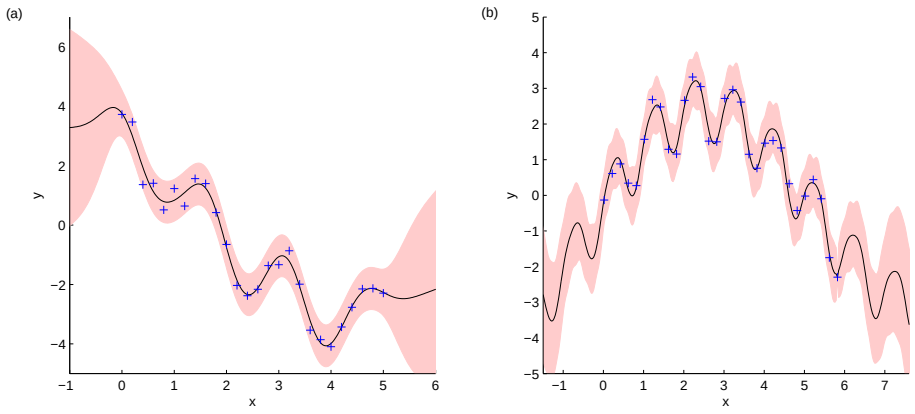

Figure 3: Estimation of y_* (solid line) for a function with **(a)** short-term and long-term dynamics, and **(b)** long-term dynamics and a periodic element. Observations are shown as crosses.

The function looks as if it might contain a periodic element, but it's difficult to be sure. Let's consider another function, which we're told has a periodic element. The solid line in Figure 3(b) was regressed with the following covariance function:

$$k(x, x') = \sigma_f^2 \exp\left[\frac{-(x - x')^2}{2l^2}\right] + \exp\{-2\sin^2[\nu\pi(x - x')]\} + \sigma_n^2\delta(x, x'). \quad (12)$$

The first term represents the hill-like trend over the long term, and the second term gives periodicity with frequency ν. This is the first time we've encountered a case where x and x' can be distant and yet still 'see' each other (that is, $k(x, x') \not\approx 0$ for $x \gg x'$).

What if the dependent variable has other dynamics which, *a priori*, you expect to appear? There's no limit to how complicated $k(x, x')$ can be, provided K is positive definite. Chapter 4 of Rasmussen and Williams (2006) offers a good outline of the range of covariance functions you should keep in your toolkit.

"Hang on a minute," you ask, "isn't choosing a covariance function from a toolkit a lot like choosing a model type, such as linear versus cubic — which we discussed at the outset?" Well, there are indeed similarities. In fact, there is no way to perform regression without imposing at least a modicum of structure on the data set; such is the nature of generative modelling. However, it's worth repeating that Gaussian processes do allow the data to speak very clearly. For example, there exists excellent theoretical justification for the use of (1) in many settings (Rasmussen and Williams (2006), Section 4.3). You will still want to investigate carefully which covariance functions are appropriate for your data set. Essentially, choosing among alternative functions is a way of reflecting various forms of prior knowledge about the *physical process* under investigation.

5 DISCUSSION

We've presented a brief outline of the mathematics of GPR, but practical implementation of the above ideas requires the solution of a few algorithmic hurdles as opposed to those of data analysis. If you aren't a good computer programmer, then the code for Figures 1 and 2 is at *github.com/mebden/GPtutorial*, and more general code can be found at *www.gaussianprocess.org/gpml*.

We've merely scratched the surface of a powerful technique (MacKay, 1998). First, although the focus has been on one-dimensional inputs, it's simple to accept those of higher dimension. Whereas x would then change from a scalar to a vector, $k(x, x')$ would remain a scalar and so the maths overall would be virtually unchanged. Second, the zero vector representing the mean of the multivariate Gaussian distribution in (6) can be replaced with functions of x. Third, in addition to their use in regression, GPs are applicable to integration, global optimization, mixture-of-experts models, unsupervised learning models, and more — see Chapter 9 of Rasmussen and Williams (2006). The next tutorial will focus on their use in *classification*.

REFERENCES

MacKay, D. (1998). In C.M. Bishop (Ed.), Neural networks and machine learning. (NATO ASI Series, Series F, Computer and Systems Sciences, Vol. 168, pp. 133-166.) Dordrecht: Kluwer Academic Press.

Rasmussen, C. and C. Williams (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Sivia, D. and J. Skilling (2006). *Data Analysis: A Bayesian Tutorial* (second ed.). Oxford Science Publications.

APPENDIX

Imagine a data sample $\mathbf{d}$ taken from some multivariate Gaussian distribution with zero mean and a covariance given by matrix D. Now decompose $\mathbf{d}$ arbitrarily into two consecutive subvectors $\mathbf{a}$ and $\mathbf{b}$ — in other words, writing $\mathbf{d} \sim \mathcal{N}(\mathbf{0}, D)$ would be the same as writing

$$\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}A & C^{\mathrm{T}}\\ C & B\end{bmatrix}\right), \tag{13}$$

where A, B, and C are the corresponding bits and pieces that make up D.

Interestingly, the conditional distribution of $\mathbf{b}$ given $\mathbf{a}$ is itself Gaussian-distributed. If the covariance matrix D were diagonal or even block diagonal, then knowing $\mathbf{a}$ wouldn't tell us anything about $\mathbf{b}$: specifically, $\mathbf{b}|\mathbf{a} \sim \mathcal{N}(\mathbf{0}, B)$. On the other hand, if C were nonzero, then some matrix algebra leads us to

$$\mathbf{b}|\mathbf{a} \sim \mathcal{N}(CA^{-1}\mathbf{a},\ B - CA^{-1}C^{\mathrm{T}}). \tag{14}$$

The mean, $CA^{-1}\mathbf{a}$, is known as the 'matrix of regression coefficients', and the variance, $B - CA^{-1}C^{\mathrm{T}}$, is the 'Schur complement of A in D'.

In summary, if we know some of $\mathbf{d}$, we can use that to inform our estimate of what the rest of $\mathbf{d}$ might be, thanks to the revealing off-diagonal elements of D.

Gaussian Processes for Classification: A Quick Introduction

M. Ebden, August 2008
Prerequisite reading: *Gaussian Processes for Regression*

1 Overview

As mentioned in the previous document, GPs can be applied to problems other than regression. For example, if the output of a GP is squashed onto the range $[0, 1]$, it can represent the *probability* of a data point belonging to one of say two types, and voilà, we can ascertain classifications. This is the subject of the current document.

The big difference between GPR and GPC is how the output data, $\mathbf{y}$, are linked to the underlying function outputs, $\mathbf{f}$. They are no longer connected simply via a noise process as in (2) in the previous document, but are instead now discrete: say $y = 1$ precisely for one class and $y = -1$ for the other. In principle, we could try fitting a GP that produces an output of approximately 1 for some values of x and approximately -1 for others, simulating this discretization. Instead, we interpose the GP between the data and a squashing function; then, classification of a new data point x_* involves two steps instead of one:

1. Evaluate a 'latent function' f which models qualitatively how the likelihood of one class versus the other changes over the x axis. This is the GP.
2. Squash the output of this latent function onto $[0, 1]$ using any sigmoidal function, $\pi(f) = \text{prob}(y = 1|f)$.

Writing these two steps schematically,

$$\boxed{\text{data, } x_*} \xrightarrow{\text{GP}} \boxed{\text{latent function, } f_*|x_*} \xrightarrow{\text{sigmoid}} \boxed{\text{class probability, } \pi(f_*)}.$$

The next section will walk you through more slowly how such a classifier operates. Section 3 explains how to train the classifier, so perhaps we're presenting things in reverse order! Section 4 handles classification when there are more than two classes.

Before we get started, a quick note on $\pi(f)$. Although other forms will do, here we will prescribe it to be the cumulative Gaussian distribution, $\Phi(f)$. This S-shaped function satisfies our needs, mapping high f into $\pi(f) \approx 1$, and low f into $\pi(f) \approx 0$.

A second quick note, revisiting (6) and (7) in the first document: confirm for yourself that, if there were no noise ($\sigma_n = 0$), the two equations could be rewritten as

$$\begin{bmatrix} \mathbf{f} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K & K_*^{\mathrm{T}} \\ K_* & K_{**} \end{bmatrix}\right) \tag{1}$$

and

$$f_*|\mathbf{f} \sim \mathcal{N}(K_* K^{-1}\mathbf{f},\ K_{**} - K_* K^{-1} K_*^{\mathrm{T}}). \tag{2}$$

2 USING THE CLASSIFIER

Suppose we've trained a classifier from n input data, $\mathbf{x}$, and their corresponding expert-labelled output data, $\mathbf{y}$. And suppose that in the process we formed some GP outputs $\mathbf{f}$ corresponding to these data, which have some uncertainty but mean values given by $\hat{\mathbf{f}}$. We're now ready to input a new data point, x_*, in the left side of our schematic, in order to determine at the other end the probability π_* of its class membership.

In the first step, finding the probability $p(f_*|\mathbf{f})$ is similar to GPR, i.e. we adapt (2):

$$p(f_*|\mathbf{f}) = \mathcal{N}(K_* K^{-1}\hat{\mathbf{f}},\ K_{**} - K_*(K')^{-1}K_*^{\mathrm{T}}). \tag{3}$$

(K' will be explained soon, but for now consider it to be very similar to K.) In the second step, we squash f_* to find the probability of class membership, $\pi_* = \pi(f_*) = \Phi(f_*)$. The expected value is

$$\overline{\pi}_* = \int \pi(f_*)p(f_*|\mathbf{f})df_*. \tag{4}$$

This is the integral of a cumulative Gaussian times a Gaussian, which can be solved analytically. By Section 3.9 of Rasmussen and Williams (2006), the solution is:

$$\overline{\pi}_* = \Phi\left(\frac{\overline{f}_*}{\sqrt{1+\mathrm{var}(f_*)}}\right). \tag{5}$$

An example is depicted in Figure 1.

3 TRAINING THE GP IN THE CLASSIFIER

Our objective now is to find $\hat{\mathbf{f}}$ and K', so that we know everything about the GP producing (3), the first step of the classifier. The second step of the classifier does not require training as it's a fixed sigmoidal function.

Among the many GPs which could be partnered with our data set, naturally we'd like to compare their usefulness quantitatively. Considering the outputs $\mathbf{f}$ of a certain GP, how likely they are to be appropriate for the training data can be decomposed using Bayes' theorem:

$$p(\mathbf{f}|\mathbf{x},\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{x})}{p(\mathbf{y}|\mathbf{x})}. \tag{6}$$

Let's focus on the two factors in the numerator. Assuming the data set is i.i.d.,

$$p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{n} p(y_i|f_i). \tag{7}$$

Dropping the subscripts in the product, $p(y|f)$ is informed by our sigmoid function, $\pi(f)$. Specifically, $p(y=1|f)$ is $\pi(f)$ by definition, and to complete the picture, $p(y=-1|f) = 1-\pi(f)$. A terse way of combining these two cases is to write $p(y|f) = \Phi(yf)$.

The second factor in the numerator is $p(\mathbf{f}|\mathbf{x})$. This is related to the output of the first step of our schematic drawing, but first we're interested in the value of $p(\mathbf{f}|\mathbf{x})$ which maximizes the posterior probability $p(\mathbf{f}|\mathbf{x},\mathbf{y})$. This occurs when the derivative of (6) with respect to $\mathbf{f}$ is zero, or equivalently and more simply, when the derivative

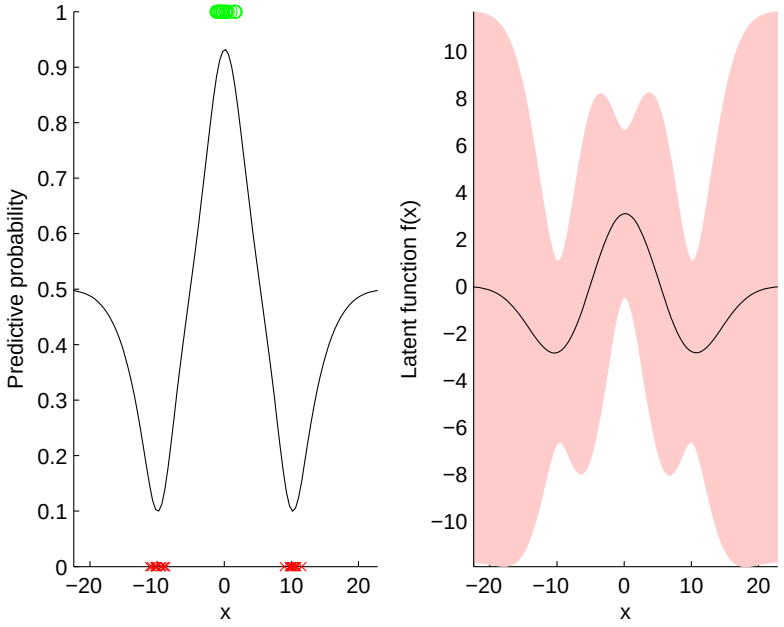

Figure 1: **(a)** Toy classification dataset, where circles and crosses indicate class membership of the input (training) data $\mathbf{y}$, at locations $\mathbf{x}$. $\mathbf{y}$ is 1 for one class and -1 for another, but for illustrative purposes we pretend $y = 0$ instead of -1 in this figure. The solid line is the (mean) probability $\overline{\pi}_* = \text{prob}(y_* = 1|x_*)$, i.e. the 'answer' to our problem after successfully performing GPC. **(b)** The corresponding distribution of the latent function f, not constrained to lie between 0 and 1.

of its logarithm is zero. Doing this, and using the same logic that produced (10) in the previous document, we find that

$$\hat{\mathbf{f}} = K\nabla \log p(\mathbf{y}|\hat{\mathbf{f}}), \tag{8}$$

where $\hat{\mathbf{f}}$ is the best $\mathbf{f}$ for our problem. Unfortunately, $\hat{\mathbf{f}}$ appears on both sides of the equation, so we make an initial guess (zero is fine) and go through a few iterations. The answer to (8) can be used directly in (3), so we've found one of the two quantities we seek therein.

The variance of $\mathbf{f}$ is given by the negative second derivative of the logarithm of (6), which turns out to be $(K^{-1} + W)^{-1}$, with $W = -\nabla\nabla \log p(\mathbf{y}|\mathbf{f})$. Making a *Laplace approximation*, we pretend $p(\mathbf{f}|\mathbf{x}, \mathbf{y})$ is Gaussian distributed, i.e.

$$p(\mathbf{f}|\mathbf{x}, \mathbf{y}) \sim q(\mathbf{f}|\mathbf{x}, \mathbf{y}) = \mathcal{N}(\hat{\mathbf{f}}, (K^{-1} + W)^{-1}). \tag{9}$$

(This assumption is occasionally inaccurate, so if it yields poor classifications, better ways of characterizing the uncertainty in $\mathbf{f}$ should be considered, for example via expectation propagation.)

Now for a subtle point. The fact that $\mathbf{f}$ can vary means that using (2) directly is inappropriate: in particular, its mean is correct but its variance no longer tells the whole

story. This is why we use the adapted version, (3), with K' instead of K. Since the varying quantity in (2), $\mathbf{f}$, is being multiplied by $K_* K^{-1}$, we add $K_* K^{-1} \text{cov}(\mathbf{f}) K^{-1} K_*^{\mathrm{T}}$ to the variance in (2). Simplification leads to (3), in which $K' = K + W^{-1}$.

With the GP now completely specified, we're ready to use the classifier as described in the previous section.

GPC in the Real World

As with GPR, the reliability of our classification is dependent on how well we select the covariance function in the GP in our first step. The parameters are $\boldsymbol{\theta} = \{l, \sigma_f\}$, one fewer now because $\sigma_n = 0$. However, as usual, $\boldsymbol{\theta}$ is optimized by maximizing $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$, or (omitting $\boldsymbol{\theta}$ on the righthand side of the equation),

$$p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}) = \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|\mathbf{x}) d\mathbf{f}. \tag{10}$$

This can be simplified, using a Laplace approximation, to yield

$$p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}) = -\frac{1}{2} \hat{\mathbf{f}}^{\mathrm{T}} K^{-1} \hat{\mathbf{f}} + \log p(\mathbf{y}|\hat{\mathbf{f}}) - \frac{1}{2} \log(|K| \cdot |K^{-1} + W|). \tag{11}$$

This is the equation to run your favourite optimizer on, as performed in GPR.

4 MULTI-CLASS GPC

We've described binary classification, where the number of possible classes, C, is just two. In the case of $C > 2$ classes, one approach is to fit an f for each class. In the first of the two steps of classification, our GP values are concatenated as

$$\mathbf{f} = (f_1^1, \ldots, f_n^1, f_1^2, \ldots, f_n^2, \ldots, f_1^C, \ldots, f_n^C)^{\mathrm{T}}. \tag{12}$$

Let $\mathbf{y}$ be a vector of the same length as $\mathbf{f}$ which, for each $i = 1, \ldots, n$, is 1 for the class which is the label and 0 for the other $C-1$ entries. Let K grow to being block diagonal in the matrices $K^1, \ldots, K^C$. So the first change we see for $C > 2$ is a lengthening of the GP. Section 3.5 of Rasmussen and Williams (2006) offers hints on keeping the computations manageable.

The second change is that the (merely one-dimensional) cumulative Gaussian distribution is no longer sufficient to describe the squashing function in our classifier; instead we use the softmax function. For the ith data point,

$$p(y_i^c|\mathbf{f}_i) = \pi_i^c = \frac{\exp(f_i^c)}{\sum_{c'} \exp(f_i^{c'})} \tag{13}$$

where $\mathbf{f}_i$ is a nonconsecutive subset of $\mathbf{f}$, viz. $\mathbf{f}_i = \{f_i^1, f_i^2, \ldots, f_i^C\}$. We can summarize our results with $\boldsymbol{\pi} = \{\pi_1^1, \ldots, \pi_n^1, \pi_1^2, \ldots, \pi_n^2, \ldots, \pi_1^C, \ldots, \pi_n^C\}$.

Now that we've presented the two big changes needed to go from binary- to multi-class GPC, we continue as before. Setting to zero the derivative of the logarithm of the components in (6), we replace (8) with

$$\hat{\mathbf{f}} = K(\mathbf{y} - \hat{\boldsymbol{\pi}}). \tag{14}$$

The corresponding variance is $(K^{-1} + W)^{-1}$ as before, but now $W = \text{diag}(\boldsymbol{\pi}) - \Pi\Pi^{\mathrm{T}}$, where Π is a $Cn \times n$ matrix obtained by stacking vertically the diagonal matrices $\text{diag}(\boldsymbol{\pi}^c)$, if $\boldsymbol{\pi}^c$ is the subvector of $\boldsymbol{\pi}$ pertaining to class c.

With these quantities estimated, we have enough to generalize (3) to

$$p(f_*^c|\mathbf{f}) = \mathcal{N}\big(K_*^c(K^c)^{-1}\hat{\mathbf{f}}^c,\ \mathrm{diag}(K_{**}) - (K_*^c)^{\mathrm{T}}(K^c + (W^c)^{-1})^{-1}(K_*^c)^{\mathrm{T}}\big), \quad (15)$$

where f_*^c, K_*^c, and W^c represent the class-relevant information only. Finally, (11) is replaced with

$$p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}) = -\frac{1}{2}\hat{\mathbf{f}}^{\mathrm{T}}K^{-1}\hat{\mathbf{f}} + \mathbf{y}^{\mathrm{T}}\hat{\mathbf{f}} - \sum_{i=1}^{n}\log\left[\sum_{c=1}^{C}\exp\hat{f}_i^c\right] - \frac{1}{2}\log\big(|K|\cdot|K^{-1}+W|\big). \quad (16)$$

We won't present an example of multi-class GPC, but hopefully you get the idea.

5 Discussion

As with GPR, classification can be extended to accept x values with multiple dimensions, while keeping most of the mathematics unchanged. Other possible extensions include using the expectation propagation method in lieu of the Laplace approximation as mentioned previously, putting confidence intervals on the classification probabilities, calculating the derivatives of (16) to aid the optimizer, or using the variational Gaussian process classifiers described in MacKay (1998), to name but four extensions.

Second, we repeat the Bayesian call from the previous document to integrate over a range of possible covariance function parameters. This should be done regardless of how much prior knowledge is available — see for example Chapter 5 of Sivia and Skilling (2006) on how to choose priors in the most opaque situations.

Third, we've again spared you a few practical algorithmic details; computer code is available at *www.gaussianprocess.org/gpml*, with examples.

Acknowledgments

Thanks are due to Prof Stephen Roberts and members of the Pattern Analysis Research Group, as well as the ALADDIN project (www.aladdinproject.org).

References

MacKay, D. (1998). In C.M. Bishop (Ed.), Neural networks and machine learning. (NATO ASI Series, Series F, Computer and Systems Sciences, Vol. 168, pp. 133-166.) Dordrecht: Kluwer Academic Press.

Rasmussen, C. and C. Williams (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Sivia, D. and J. Skilling (2006). *Data Analysis: A Bayesian Tutorial* (second ed.). Oxford Science Publications.

Gaussian Processes for Dimensionality Reduction: A Quick Introduction

M. Ebden, August 2015

Prerequisite reading: *Gaussian processes for regression*

Suppose you wanted to learn a low-dimensional representation of a dataset for ease of interpretation, sort of like how your shadow is a simplified, two-dimensional representation of a three-dimensional object. Let the original data be matrix $\mathbf{Y}$ with n rows (observations) of d dimensions each, and let the new representation be $\mathbf{X}$ with n rows of q dimensions each ($q < d$).

To learn this low-dimensional representation, we'll assume that for the ith dimension of $\mathbf{Y}$ the n elements in $\mathbf{y}_{:,i}$ are a sample from a Gaussian process based on the low-dimensional space. Specifically, we'll use a zero-mean Gaussian process, $\mathcal{GP}\left(\mathbf{0}_{n\times 1}, k(\mathbf{x}, \mathbf{x}')\right)$, keeping the same model for each dimension of $\mathbf{Y}$. We'll choose for $k(\cdot)$ the squared exponential kernel (a.ka. RBF kernel) in order to ensure that points close in $\mathbf{X}$ will be close in $\mathbf{Y}$. Renaming the noise variance from σ_n^2 (in the first report) to $1/\beta$, the kernel is:

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left[\frac{-|\mathbf{x} - \mathbf{x}'|^2}{2l^2}\right] + \frac{\delta(x, x')}{\beta}.$$

The covariance matrix $\mathbf{K}$ for this GP is constructed as per (4) in the first report, and we have no need for a $\mathbf{K}_*$ or $\mathbf{K}_{**}$ from (5) because there are no points in $\mathbf{Y}$ to predict. The tuple of kernel parameters this time is $\boldsymbol{\theta} = \{\sigma, l, \beta\}$.

We'll further assume that the GPs behind each of the d dimensions have been sampled independently. Therefore, the likelihood of the observed values in $\mathbf{Y}$ is the product of d independent GPs:

$$\mathrm{p}(\mathbf{Y}|\mathbf{X}, \boldsymbol{\theta}) = \prod_{i=1}^{d} \frac{\exp\left[-\frac{1}{2}\mathbf{y}_{:,i}^{\mathrm{T}}\mathbf{K}^{-1}\mathbf{y}_{:,i}\right]}{(2\pi)^{n/2}|\mathbf{K}|^{1/2}}.$$

Then we can adjust $\mathbf{X}$ and $\boldsymbol{\theta}$ to maximize this likelihood. If we were only adjusting $\mathbf{X}$, it would be akin to determining the shadow which is most likely to correspond to your body's shape. Because we're adjusting $\boldsymbol{\theta}$ as well, imagine manipulating the light source and the surface for the shadow to appear on as well.

Figure 1 gives an example outcome of the most likely $\mathbf{X}$ for a certain $\mathbf{Y}$, with $n = 17$, $d = 2$, and $q = 1$. $\mathbf{Y}$ was preprocessed to have zero mean and the same variance in each dimension. Using an optimizer based on scaled conjugate gradients, the optimal $\boldsymbol{\theta}$ and $\mathbf{X}$ were then found. The former is $\{\sigma, l, \beta\} = \{1.05, 3.3\times 10^{-4}, 93\}$ and the latter is given in Figure 1(b). In this example, $\mathbf{X}$ is more than just a 'shadow' (simple projection) of $\mathbf{Y}$: the above method of dimensionality reduction is powerful enough to have learned that the points in the left curve in Figure 1(a) should be grouped together while the points in the right curve form a separate group; no simple lamplight projection can achieve this. The price we pay for this flexibility is in the high number of parameters being fit: the total here is $3+17\times 1 = 20$, i.e. $\boldsymbol{\theta}$ plus the one-dimensional values in $\mathbf{X}$. Usually $\mathbf{X}$ are referred to as latent variables rather than parameters, and

our overall setup is called the 'Gaussian process latent variable model' (GP-LVM). The technique was first presented by Neil Lawrence in 2004; he examined a set of oil-flow data in which $n = 1000$ and $d = 12$, which we reduce to $q = 2$ in Figure 2.

This tutorial on the GP-LVM is provided by Mind Foundry, a technology spin-out from the University of Oxford.

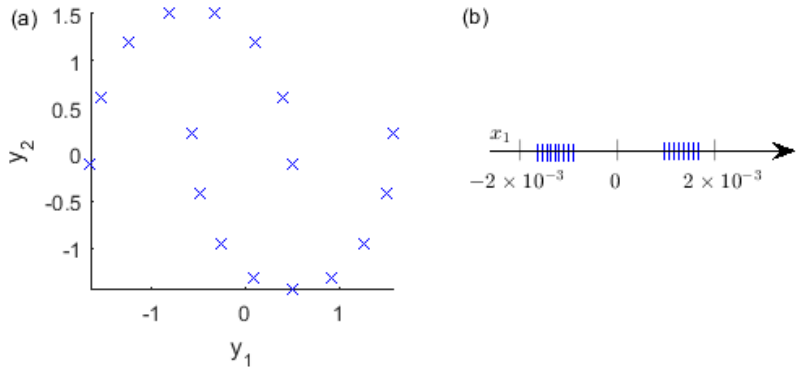

Figure 1: **(a)** Seventeen data points, $\mathbf{Y}$, each of two dimensions.
(b) A one-dimensional projection, $\mathbf{X}$, of those points.

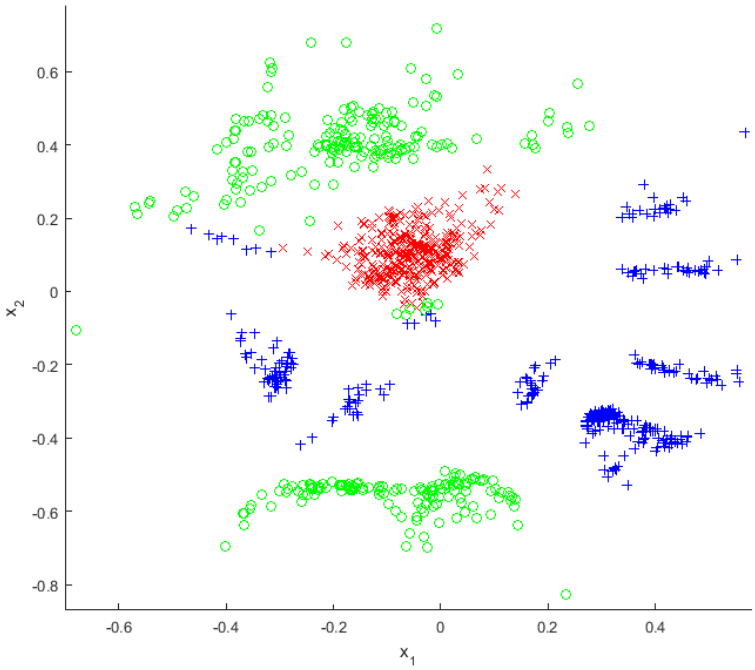

Figure 2: Although the GP-LVM algorithm uses unsupervised learning, the three classes of data in this originally 12-D dataset generally occupy different regions.