STA561: Probabilistic machine learning

Gaussian Processes (10/16/13)

Lecturer: Barbara Engelhardt *Scribes: Changwei Hu, Di Jin, Mengdi Wang*

1 Introduction

In supervised learning, we observe some inputs $\mathbf{x_i}$ and some outputs y_i. We assume that $y_i = f(\mathbf{x}_i)$, for some unknown f, possibly corrupted by noise: $y_i = f(\mathbf{x_i}) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. One approach to modeling this relationship is to compute a posterior *distribution over functions* given the data, $p(f|\mathbf{X}, \mathbf{y})$, and then to use this posterior distribution to make predictions given new inputs, marginalizing over all possible functions, i.e., to compute:

$$p(y^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y}) = \int_{\mathcal{F}} p(y^*|f, \mathbf{x}^*)p(f|\mathbf{X}, \mathbf{y})df.$$

Up until now, we have focused on parametric representations for the function f, so that instead of inferring $p(f|\mathcal{D})$, we infer $p(\theta|\mathcal{D})$. In this lecture, we discuss a way to perform Bayesian inference on a distribution of functions.

Our approach will be based on **Gaussian processes** or **GPs**. A GP defines a prior distribution over functions; when we use it as a prior, we can build a posterior distribution over functions given data observations. Although it might seem difficult to represent a distribution over functions, a key point is that, in GPs, we will define a posterior distribution over $f(\mathbf{x})$ at all values $\mathbf{X} \in \mathcal{X}$ given a finite set of observed points, say $\mathbf{x_1}, ..., \mathbf{x_n}$. In order to infer the function at an infinite number of points given a finite number of points, a GP assumes that $p(f(\mathbf{x}_1), f(\mathbf{x}_2), ...f(\mathbf{x}_n))$ is jointly Gaussian, with a mean $\mu(\mathbf{x})$ and covariance $\mathbf{\Sigma}(\mathbf{x})$ given by $\sum_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$, where κ is a positive definite kernel function. We use the kernel to enforce smoothness of the random function across time points: if $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar according to our kernel function κ, then this constrains the distribution of our function $f(\mathbf{x_i})$ and $f(\mathbf{x_j})$ so that they are (in probability) similar:

$$cov(f(\mathbf{x_i}), \mathbf{f}(\mathbf{x_j})) = \kappa(\mathbf{x_i}, \mathbf{x_j}).$$

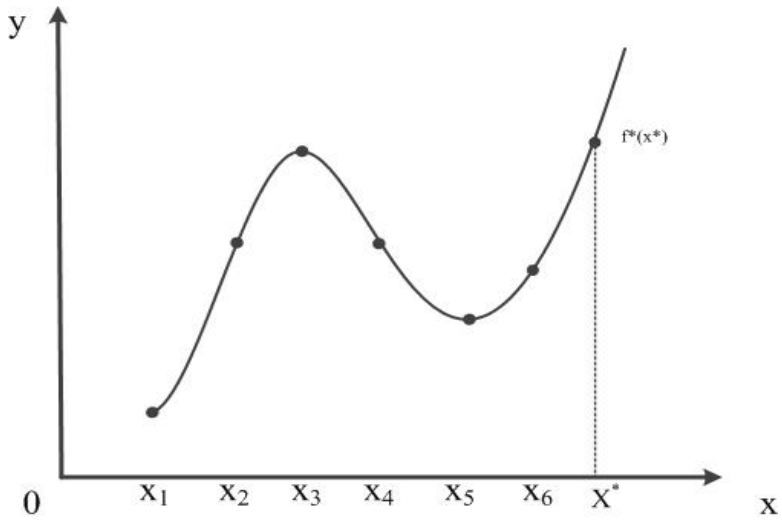

Figure 1: The x-axis is $\mathbf{x}$, the y-axis is $f(\mathbf{x})$, and this shows one instance of a function in this GP space.

2 Gaussian Processes

2.1 Prediction

We observe some inputs $\mathbf{x_i}$ and some outputs y_i. We assume that $y_i = f(\mathbf{x}_i) + \epsilon$ (Figure 1), for some unknown $f(\cdot)$ corrupted by Gaussian noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^{\mathbf{2}})$.

$$\mathbf{y_i} = \mathbf{f}(\mathbf{x_i}) + \epsilon_\mathbf{i}$$

A simple example can be illustrated in Figure 1, $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5), (x_6, y_6)\}$ are observed data, the curved line is one possible function of $\mathbf{x}$, i.e. $f(\mathbf{x})$. To make a prediction for a x^*, can find the expected value y^* from the expected value of the posterior probability of $f(\mathbf{x}^*)$.

2.2 Distribution over functions

Our approach will be to define a GP prior over functions. We need to be able to define a distribution over the function's values at a finite set of points, say $\mathbf{x_1}, ..., \mathbf{x_n}$. Recall that a GP assumes that $p(f(\mathbf{x}_1), f(\mathbf{x}_2), ...f(\mathbf{x}_n))$ is jointly Gaussian, with some mean $\mu(\mathbf{x}) = \mathbf{E}[\mathbf{f}(\mathbf{x})]$ and covariance $\mathbf{\Sigma}(\mathbf{x})$ given by $\Sigma_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$, where κ is a positive definite kernel function. In particular, $f(x) \sim GP(\mu(x), K(x, x'))$. Often, $\mu(x) = 0$ for simplicity.

The posterior distribution of the function given the data is: $p(f|X, y) = \mathcal{N}(f|\mu, K)$ where $\mu = (\mu(x_1), \ldots, \mu(x_n))$ and $\Sigma_{ij} = \kappa(\mathbf{x_i}, \mathbf{x_j})$. We can produce predictions from this posterior

distribution as follows:

$$p(y^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y}) = \int_{\mathcal{F}} p(y_*|f, \mathbf{x}_*)p(f|\mathbf{X}, \mathbf{y})df.$$

Because this is a Gaussian distribution, it turns out that this integral is straightforward. We will write this out below.

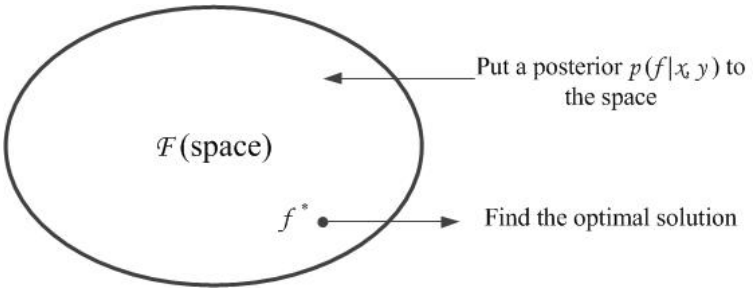

Figure 2: Gaussian process puts a distribution on function space $\mathcal{F}$.

- **Gaussian Process** is a distribution over functions, used as a prior, we can define a posterior distribution over functions given data.
- **Key point:** data do not need to define function for all values of $\mathbf{x}$, but only at a finite number of values.

In Figure 2 we assume that $p(f(\mathbf{x}_1), f(\mathbf{x}_2), ...f(\mathbf{x}_N))$ follows some distribution (i.e., there is a probability $p(f|\mu, \kappa)$ over function in function space $\mathcal{F}$). Given a new data x^*, we predict y^* by using statistical machinery (i.e., the posterior predictive distribution).

2.3 Predictions for noise-free observations

Suppose we observed a training set $\mathcal{D} = \{(\mathrm{x}_i, f(x_i)), i = 1 : n\}$, where $f_i = f(\mathrm{x}_i)$ is the noise-free observation of the function evaluated at x_i. Given a test set $\mathbf{X}_*$ of size $n_* \times p$, we want to predict the function outputs $\mathbf{f}_*$.

In the setting of no uncertainty (i.e., $\epsilon = 0$), if we ask the GP to predict $f(\mathrm{x})$ for a value of x that it has already seen, we want the GP to return the answer $f(\mathrm{x})$ with no uncertainty. In other words, it should memorize the training data, and perform noiseless *interpolation* of points $\mathbf{x}^*$ not in the training data set. This will only happen if the observations are noiseless, i.e., every time $\mathbf{x}$ appears in the data set, the same $f(\mathbf{x})$ appears as the label. We will consider the case of noisy observations later.

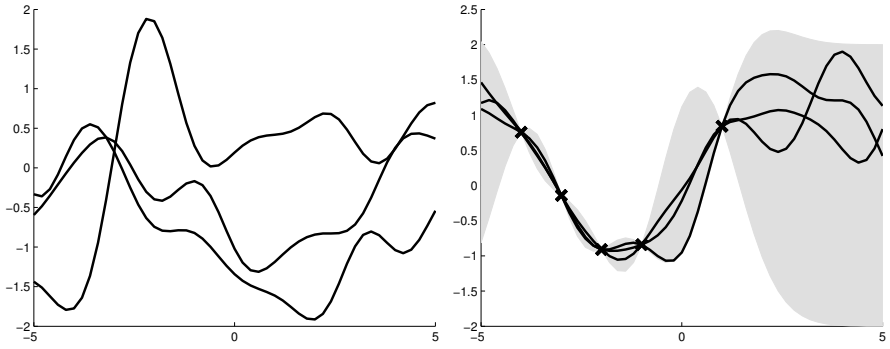

Figure 3: Left: functions sampled from a GP prior with SE kernel. Right: samples from a GP posterior, after conditioning on five noise-free observations. The shaded area represents $\mathbb{E}[f(\mathrm{x})] \pm 2\mathrm{std}(\mathrm{f(x)})$. (Figure 15.22 from Murphy, 2012)

Now we return to the prediction problem. By the definition of the GP, the joint distribution has the following form

$$\begin{pmatrix} \mathbf{f} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu \\ \mu_* \end{pmatrix}, \begin{pmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^{\mathbf{T}} & \mathbf{K}_{**} \end{pmatrix} \right)$$

where $\mathbf{K} = \kappa(\mathrm{X}, \mathrm{X})$ is $n \times n$, $\mathbf{K}_* = \kappa(\mathrm{X}, \mathrm{X}_*)$ is $n \times n_*$, and $\mathbf{K}_{**} = \kappa(\mathrm{X}_*, \mathrm{X}_*)$ is $n_* \times n_*$. By the standard rules for conditioning Gaussians (see previous lecture), the posterior has the following form

$$\mathbf{f}_* | \mathbf{f}, \boldsymbol{X}, \mathbf{X}_* \sim N\left(m_*, \boldsymbol{\Sigma}_*\right)$$

where $m_* = \mu\left(\mathbf{X}_*\right) + \mathbf{K}_*^{\mathbf{T}} \mathbf{K}^{-\mathbf{1}}\left(\mathbf{f} - \mu\left(\mathbf{X}\right)\right)$, and $\boldsymbol{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_*^{\mathbf{T}} \mathbf{K}^{-\mathbf{1}} \mathbf{K}_*$. This allows us to compute the posterior prediction for noiseless function $f(\mathbf{x}^*)$ given our data and new sample $\mathbf{x}^*$.

This process is illustrated in Figure 5. On the left we show sample samples from the prior, $p(\mathbf{f}|\mathbf{X})$, where we use a squared exponential kernel (i.e., a Gaussian kernel or RBF kernel). On the right we show samples from the posterior, $p(\mathbf{f}_*|\mathbf{f}, \boldsymbol{X}, \mathbf{X}_*)$. We see that the model interpolates the training data according to the posterior probability of the points along X, and that the predictive uncertainty increases as we move further away from the observed data. This uncertainty is strongly regulated by the choice of the kernel. If we changed the length scale of the kernel, on one hand we may have much less uncertainty (i.e., a smoother function) or, changing the length scale the other way, we may have much greater uncertainty in our predictions (i.e., a less smooth function). This is how the kernel controls the smoothness of the functions in the underlying function space.

2.4 Relationship between kernel and smoothness

When we use GPs for prediciton, we infer the distribution of functions in between observed data points $\mathbf{x}$ using some notion of smoothness of the functions in function space (see Figure 3). Let us assume $p(f(\mathbf{x}_1), ..., f(\mathbf{x}_N))$ is jointly Gaussian:

$$f(\mathbf{x}_1), ..., f(\mathbf{x}_N) \sim \mathcal{N}_n(\mu(\mathbf{x}), \mathbf{\Sigma}(\mathbf{x}))$$

where $\mathbf{\Sigma}(\mathbf{x}) = \mathbf{K}$, $\mathbf{K_{i,j}} = \kappa(\mathbf{x_i}, \mathbf{x_j})$, and κ is our *Mercer* kernel function.
The idea is: the quantification of similarity among x_i and x_j is actually a measure of the similarity between $f(x_i)$ and $f(x_j)$, i.e., $\text{cov}\left(f(x_i), f(x_j)\right) = \kappa(x_i, x_j)$. If neighboring points are similar, then correspondingly the conditional probability of one of the points $f(\mathbf{x}')$ given $\mathbf{x}, \mathbf{x}', \mathbf{f}(\mathbf{x})$ is going to have less uncertainty; whereas if those points have no similarity, then the same conditional probability will simply revert to the prior probability.

As the example of Figure 4:

- at each point x, $f(\text{x})$ is marginally Gaussian.
- between $\mathbf{x}$ points, we see the smoothness of functions.
- the smoothness is dictated by the kernel. For example, if we use RBF kernels, then, under small length scale, $\kappa(\text{x}, \text{x}')$ is near zero, so the function is non-smooth (the dashed line). Under large length scale, $\kappa(\text{x}, \text{x}')$ is larger, so the function is smoother between noiseless observations (the solid line).

2.5 Predictions using noisy observations

Now let us consider the case where what we observe is a noisy version of the underlying function, $y = f(\text{x}) + \epsilon$, where $\epsilon \sim N\left(0, \sigma_y^2\right)$. Now at each point $\mathbf{x}$, the corresponding $f(\mathbf{x})$ also has noise because we observe $y = f(\mathbf{x}) + \epsilon$. In other words, there is now conditional uncertainty in the function space, conditioning on a specific $\mathbf{x}$. In this case, predictions now must incorporate this uncertainty. The covariance of the observed noisy responses y_i and y_j changes only along the diagonal (i.e., the variance at a specific point $\mathbf{x}$):

$$\text{cov}\left(y_i, y_j\right) = \kappa\left(x_i, x_j\right) + \sigma_y^2 \delta\left(i = j\right)$$

From the above equation, we know that if $i \neq j$

$$\text{cov}\left(y_i, y_j\right) = \kappa\left(x_i, x_j\right)$$

If $i = j$, then

$$\text{cov}\left(y_i, y_j\right) = \kappa\left(x_i, x_i\right) + \sigma_y^2$$

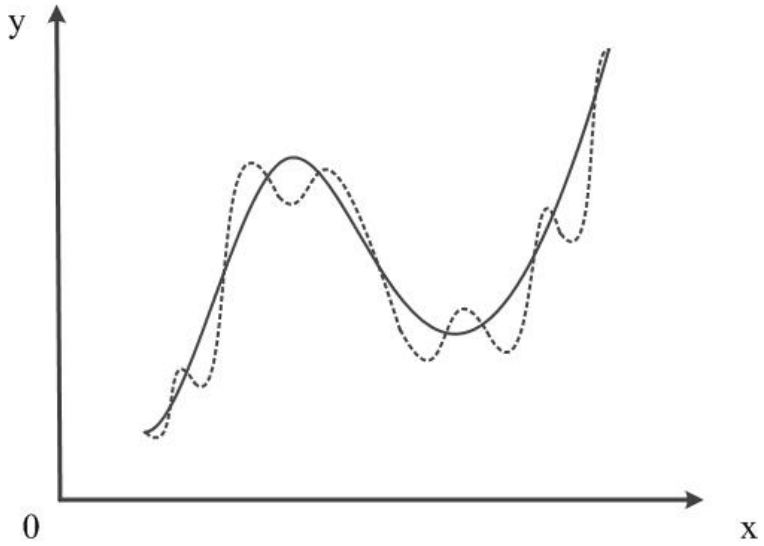

Figure 4: Two draws from a noiseless GP posterior with different length scales in an RBF kernel.

Let us define

$$\mathrm{cov}\left(\mathbf{y}|\mathbf{x}\right) = \mathbf{K} + \sigma_{\mathbf{y}}^{\mathbf{2}}\mathbf{I_n} \triangleq \mathbf{K_y}$$

The joint distribution of $\mathbf{y}$ and $\mathbf{f}_*$ is

$$\begin{pmatrix} \mathbf{y} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{pmatrix} \mathbf{K_y} & \mathbf{K}_* \\ \mathbf{K}_*^{\mathbf{T}} & \mathbf{K}_{**} \end{pmatrix}\right)$$

Here we assume the mean is zero for notational simplicity.

As before, using the conditional Gaussians, given observed $\mathbf{y}$, $\mathbf{x}$ and new data $\mathbf{x}_*$, $\mathbf{f}_*$ is distributed as

$$\mathbf{f}_*|y, x, x_* \sim N\left(\mu_*, \mathbf{\Sigma}_*\right)$$

where $\mu_* = \mathbf{K}_*^{\mathbf{T}}\mathbf{K}_{\mathbf{y}}^{-\mathbf{1}}\mathbf{y}$, and $\mathbf{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_*^{\mathbf{T}}\mathbf{K}_{\mathbf{y}}^{-\mathbf{1}}\mathbf{K}_*$.

For a single test input x_*, let $\mathbf{k}_* = \left[\kappa\left(\mathbf{x}_*, \mathbf{x_1}\right), ..., \kappa\left(\mathbf{x}_*, \mathbf{x_n}\right)\right]$, another way to write the posterior mean is

$$\bar{\mathbf{f}}_* = \mathbf{k}_*^{\mathbf{T}}\mathbf{K}_{\mathbf{y}}^{-\mathbf{1}}\mathbf{y} = \sum_{\mathbf{i=1}}^{\mathbf{n}} \alpha_{\mathbf{i}}\kappa\left(\mathbf{x_i}, \mathbf{x}_*\right)$$

where $\alpha_i = \mathbf{K}_{\mathbf{y}}^{-\mathbf{1}}\mathbf{y}$. Note the similarity with this equation to the equation for prediction in Support Vector Machines (SVMs). For GPs, however, there is not necessarily the sparsity implied by the choice of support vectors: every observation has some impact for a dense matrix K and dense vector y.

3 GP Regression

GPs can be used for (nonlinear) regression. Let the prior on the (possibly non-linear) regression function $f(\cdot)$ be a GP, denoted by

$$f(\mathrm{x}) \sim GP\left(m(\mathrm{x}), \kappa(\mathrm{x}, \mathrm{x}')\right)$$

where $m(\mathrm{x})$ is the mean function and $\kappa(\mathrm{x}, \mathrm{x}')$ is the kernel or covariance function, i.e.,

$$\begin{aligned} m(\mathrm{x}) &= \mathbb{E}[f(\mathrm{x})] \\ \kappa(\mathrm{x}, \mathrm{x}') &= \mathbb{E}[(f(\mathrm{x}) - m(\mathrm{x}))(f(\mathrm{x}') - m(\mathrm{x}'))^T] \end{aligned}$$

We obviously require that $\kappa(\cdot)$ be a positive definite kernel. For any finite set of points, this process defines a joint Gaussian:

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{f}|\mu, \mathbf{K})$$

where $\mu = (m(\mathrm{x}_1), \ldots, m(\mathrm{x}_N))$ and $\mathbf{K_{ij}} = \kappa(\mathrm{x_i}, \mathrm{x_j})$. Note that it is common to use a mean function of $m(\mathrm{x} = 0)$, since the GP is flexible enough to model the mean arbitrarily well.

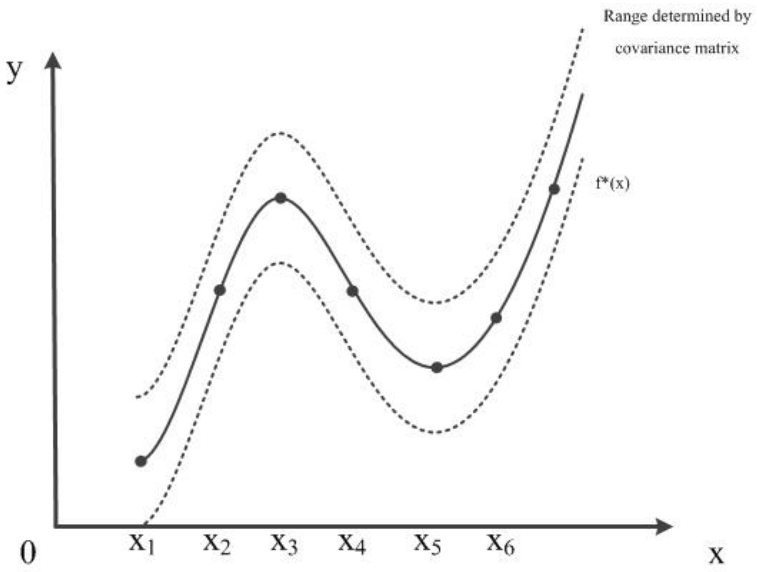

Figure 5

Then we can model GP regression, or $y_i = f(\mathbf{x_i}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ as noisy GP prediction. In particular, we have the same mean and covariance parameterization as with noisy GP prediction. Furthermore, we can compute the joint density of the response y and the underlying function f as above.

We can also do *GP classification*: as with logistic regression, 'squash' the GP response through a logistic function to get predictions between $(0, 1)$.