Methods of Data Analysis
Regression using Gaussian Process priors

Week 6

1 Motivation

In previous lectures we discussed maximum likelihood and Bayesian inference and viewed several times the example of parameter fitting in the inference context. "Parameter fitting" is a classical regression problem in statistics, or supervised learning problem in machine learning: specifically, we want to learn a mapping from a (potentially high, D-dimensional) continuous input space X into, usually, a 1D space of reals Y (alternatively, for classification problems we look for a mapping into a discrete space).

A straightforward way to do regression is to use a parametrized model for the function $f_\theta : \mathbb{R}^D \to \mathbb{R}$, where θ are the parameters. As discussed previously, this is sufficient to do a χ^2-minimization fit over θ, which is identical to doing a maximum likelihood fit under the assumption that the noise on our data is Gaussian and IID; with alternative assumptions about noise, maximum likelihood fitting is still well-defined, although maximization itself may pose a technical challenge when the number of parameters is large. Aside from the technical challenge, however, choosing a parametric model implies that we have some prior knowledge of which class of functions to even consider, which may be true for some, but definitely not all, applications.

Alternatively, we could phrase the problem as follows: given a class of functions that are sufficiently smooth (in some sense), but otherwise can be arbitrarily nonlinear, can we find one that fits data best and use it to make predictions about the function value evaluated at new points that weren't part of the training set? This can be also viewed as solving a general interpolation problem. We will see that Gaussian Processes provide a formalization of that idea. In short, a Gaussian Process prior is a prior over all functions f that are sufficiently smooth; data then "chooses" the best fitting functions from this prior, which are accessed through a new quantity, called "predictive posterior" or the "predictive distribution".

On the way toward understanding gaussian process regression, we will first revisit linear regression in a Bayesian setting. This will be the starting point for turning linear regression into nonlinear by using basis functions and projecting data into higher-dimensional "feature space" in which the nonlinear regression will turn back into the linear regression. At this point we will introduce the kernel trick (but not go into mathematical details), as the stepping stone towards gaussian processes. The crucial step will be to note that a mathematically equivalent shortcut to the "feature space" and basis functions can be made by bypassing all these constructions and, instead, simply specifying the form of the "kernel function," or a covariance function, between any pair of elements from X. Intuitively, doing regression probabilistically means drawing regression values in Y from a distribution conditioned on the values of the independent variable in X, and "smoothness" in this sense means that close by values in X need to be correlated in their Y: enforcing smoothness then means specifying the form of this correlation.

Once we are done with the technical details, simple applications of Gaussian Process regression can be implemented within a few lines of code only, which is why these tools are gaining popularity in various applications from climate change modeling to neuroscience and machine learning. Pedagogically, the formalism also brings together concepts from inference, generation of correlated random numbers, and introduces the kernel trick in a gentle way.

2 Goals

- Rephrase linear regression in a Bayesian setting.
- Extend linear regression to nonlinear using basis functions and the kernel trick.
- Introduce Gaussian Processes and their use for regression.

3 Bayesian linear regression

Consider data $\mathcal{D}$ given as a set of pairs of points in X along with the regression variable in Y: $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}$ for $t = 1, \ldots, T$. y are real valued scalars, while $\mathbf{x}$ are D-dimensional real numbers. As is customary, we can line up all training data into a matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T]$ of dimension $D \times T$.

Let's consider first the linear models of the form $y = f(\mathbf{x}) + \eta$, where $\eta \sim \mathcal{G}(0, \sigma^2)$ is IID Gaussian noise, and $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} = \mathbf{w} \cdot \mathbf{x}$ is the linear function with weight vector $\mathbf{w}$.

The likelihood of the observations is then:

$$P(\mathbf{y}|X, \mathbf{w}) = \prod_{t=1}^{T} P(y_t|\mathbf{x}_t, \mathbf{w}) = (2\pi\sigma^2)^{-D/2} \exp\left(-\frac{1}{2\sigma^2}|\mathbf{y} - \mathbf{X}^T\mathbf{w}|^2\right) = \mathcal{G}(\mathbf{X}^T\mathbf{w}, \sigma^2\mathbb{I}). \tag{1}$$

To get the posterior distribution in Bayesian inference, we combine this with a Gaussian prior over $\mathbf{w}$, $P(\mathbf{w}) = \mathcal{G}(0, \Sigma_P)$, where Σ_p is the covariance matrix. The posterior is then

$$\begin{aligned} P(\mathbf{w}|\mathbf{X}, \mathbf{y}) &\propto \mathcal{G}(\mathbf{X}^T\mathbf{w}, \sigma^2\mathbb{I}) \times \mathcal{G}(0, \Sigma_p) && (2)\\ &\propto \mathcal{G}(\sigma^{-2}\mathbf{A}^{-1}\mathbf{X}\mathbf{y}, \mathbf{A}^{-1}), && (3) \end{aligned}$$

where

$$\mathbf{A} = \frac{1}{\sigma^2}\mathbf{X}\mathbf{X}^T + \Sigma_P^{-1}. \tag{4}$$

In other words, Bayesian posterior over weights in linear regression is Gaussian, if the prior and noise distributions are Gaussian. The mean and the $D \times D$ covariance matrix $\mathbf{A}^{-1}$ of the posterior Gaussian are found easily by writing out the Gaussian product in Eq. (2) and completing the square in $\mathbf{w}$. As you could expect, the "errors" in parameters are given by $\mathbf{A}^{-1}$, which is an optimal combination of prior constraints and the data $\mathbf{X}$: the smaller the noise in the data, the more dominant is the first contribution in Eq. (4) and the less important the prior.

The best point estimate of the weights can be extracted from the posterior in the usual Bayesian fashion, by first choosing the utility function; but since the posterior in this case is a Gaussian, many choices of the utility function will yield the same estimate. For instance, the posterior mean,

$$\mathbf{w}^* = \int d\mathbf{w}\ \mathbf{w}P(\mathbf{w}|\mathbf{X}, \mathbf{y}), \tag{5}$$

which is the usual Bayesian estimator, will in this case be equal to the MAP (maximum a posteriori) estimate.

Let's now shift the perspective a bit, from making inferences about the internal parameters of the model ($\mathbf{w}$), to making predictions. Suppose we now want to ask the regression model to make a prediction of a new function value, f_*, at a new value of the independent variable, $\mathbf{x}_*$. In the Bayesian perspective, we should write down the full distribution for possible values of f_*:

$$P(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \int d\mathbf{w}\ P(\mathbf{w}|\mathbf{X}, \mathbf{y}) P(f_*|\mathbf{x}_*, \mathbf{w}). \tag{6}$$

This can be seen as first doing the Bayesian inference to get the $P(\mathbf{w}|\mathbf{X}, \mathbf{y})$, and then integrating over all possible linear models ($\mathbf{w}$) to make the prediction at the new value $\mathbf{x}_*$. Note that formally the second distribution in Eq. (6) is a delta function: $P(f_*|\mathbf{x}_*, \mathbf{w}) = \delta(f_* - \mathbf{w}^T\mathbf{x}_*)$, since the model is not a true distribution, but a function (in mathematical sense).

The new object, $P(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y})$ is called the predictive distribution, or predictive posterior. It is no longer a function of the internal parameters of the model, which have been integrated out, but solely consists of the distribution over *predicted* values given *observed* values. Evaluating this in Eq. (6) is easy, because it consists of linearly projecting a multivariate Gaussian distribution; the result then also has to be a Gaussian, whose (scalar) mean is the weighted sum of the multivariate means:

$$P(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \mathcal{G}(\sigma^{-2}\mathbf{x}_*^T\mathbf{A}^{-1}\mathbf{X}\mathbf{y},\ \mathbf{x}_*^T\mathbf{A}^{-1}\mathbf{x}_*) \tag{7}$$

4 Bayesian regression using basis functions and the kernel trick

Let's now generalize the linear regression to nonlinear regression, but try to retain the tractability of linear Gaussian models, where all integration can be done exactly. The basic idea is to "project" the independent variables, $\mathbf{x}$ into a "feature space", which could be of higher dimension than $\mathbf{x}$. One way to do this is to define a function ϕ:

$$\phi \quad : \quad \mathbf{x} \to \phi(\mathbf{x}) \tag{8}$$

$$\phi \quad : \quad \mathbb{R}^D \to \mathbb{R}^N, \tag{9}$$

where N could be larger than D, or even infinite-dimensional. Components of ϕ are then called "features", and the space into which ϕ is projecting is called the feature space. We can then make a *linear* regression in the feature space which will correspond to a nonlinear regression in the original space where $\mathbf{x}$ lives:

$$f(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}). \tag{10}$$

Note that the weight vector, $\mathbf{w}$, now has the dimension of the feature space (N, possibly infinite), rather than D as before. But so long as the function ϕ is fixed beforehand (before doing the inference), exactly the same Bayesian formalism carries through. That is, if we define $\mathbf{\Phi} = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \ldots, \phi(\mathbf{x}_T)]$ as the $N \times T$ equivalent of the matrix $\mathbf{X}$ that we had in the linear case, we can simply write down the predictive posterior distribution by everywhere replacing $\mathbf{X}$ with $\mathbf{\Phi}$ and $\mathbf{x}_*$ with $\phi(\mathbf{x}_*)$, so:

$$P(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \mathcal{G}(\sigma^{-2}\phi(\mathbf{x}_*^T)\mathbf{A}^{-1}\mathbf{\Phi}\mathbf{y},\ \phi(\mathbf{x}_*^T)\mathbf{A}^{-1}\phi(\mathbf{x}_*)), \tag{11}$$

where the new $\mathbf{A}$ is now:

$$\mathbf{A} = \sigma^{-2}\mathbf{\Phi}\mathbf{\Phi}^T + \Sigma_p^{-1} \tag{12}$$

Note that unlike in Eq. (7) where we need to be inverting matrices of dimension $D \times D$ (assumed tractable), here we need to invert matrices that have the dimension of the feature space, $N \times N$, which is possibly very large. In what follows we will rearrange Eq. (11) to show that it can be evaluated tractably. But before going there, let's consider simple examples of the basis function approach.

First, consider the case where x are scalars, and $\phi(x) = [1, x, x^2, \ldots, x^{N-1}]$. In this case, the projection expands the dimensionality of the independent variable from 1 to N, and the feature space is chosen such that the linear regression in the feature space corresponds to finding the polynomial coefficients for x. Here, the projection has turned a $(N-1)$-degree polynomial fitting problem into a higher dimensional linear problem.

Another example can be Gaussian basis functions. Suppose there is a set of Gaussian bumps G with the form $\phi(\mathbf{x}) = [G(\mathbf{x}, \mathbf{x}_0), G(\mathbf{x}, \mathbf{x}_1), G(\mathbf{x}, \mathbf{x}_2) \ldots]$, where G are indexed by a coordinate $\mathbf{x}_i$ and given by:

$$G(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\frac{1}{2\ell^2}||\mathbf{x} - \mathbf{x}_i||^2\right). \tag{13}$$

We can then consider the limit where, for example, the $\mathbf{x}_i$ form an infinite-dimensional basis that tiles the space, or even a set of functions indexed by a continuum of $\mathbf{x}_i$. In this case, the feature space is actually infinite dimensional! Why would such basis be useful? One example is that it implicitly enforces smoothness of the resulting regression, since data points are "smeared" at the scale of ℓ by the basis. But how do we make any progress in these high-dimensional spaces?

The key insight is that Eq. (11) can be rewritten in an exact form by using linear algebra. We won't fully show this result which is worked out in standard references, e.g., Ref [1], but the mean of the predictive distribution is easy to verify (as we sketch below). In short, the predictive distribution can be rewritten as:

$$\begin{aligned} P(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) &= \mathcal{G}(\phi_*^T \Sigma_p \mathbf{\Phi}(K + \sigma^2 \mathbb{I})^{-1}\mathbf{y}, & (14)\\ &\quad \phi_*^T \Sigma_p \phi_* - \phi_*^T \Sigma_p \mathbf{\Phi}(K + \sigma^2 \mathbb{I})^{-1} \mathbf{\Phi}^T \Sigma_p \phi_*), & (15) \end{aligned}$$

where $\phi_* = \phi(\mathbf{x}_*)$ and $K = \mathbf{\Phi}^T \Sigma_p \mathbf{\Phi}$.

(To show the above equivalence of the means, we have to show that $\Sigma_p \mathbf{\Phi}(K + \sigma^2 \mathbb{I})^{-1} = \sigma^{-2}\mathbf{A}^{-1}\mathbf{\Phi}$. Let's consider $\sigma^{-2}\mathbf{\Phi}(K + \sigma^2\mathbb{I}) = \sigma^{-2}\mathbf{\Phi}(\mathbf{\Phi}^T\Sigma_p\mathbf{\Phi} + \sigma^2\mathbb{I}) = (\mathbf{\Phi}\mathbf{\Phi}^T\Sigma_p\sigma^{-2} + \mathbb{I})\mathbf{\Phi} = (\sigma^{-2}\mathbf{\Phi}\mathbf{\Phi}^T + \Sigma_p^{-1})\Sigma_p\mathbf{\Phi} = \mathbf{A}\Sigma_p\mathbf{\Phi}$. Multiplying from left by $\mathbf{A}^{-1}$ and right by $(K + \sigma^2\mathbb{I})^{-1}$ completes the proof.)

The main observations about the predictive distribution in Eq. (15) are as follows:

- The dimension of the matrix that needs to be inverted, $(K + \sigma^2\mathbb{I})$, is $T \times T$ and not $N \times N$ as before. N could be in principle infinite dimensional, whereas T is the number of data points, which is finite. It can be large, and in that case approximations to the above solution might have to be devised, but in many cases the inversion is tractable and can be done numerically by, e.g., Cholesky decomposition.

- The features always enter as scalar products, e.g., as $\mathbf{\Phi}^T\Sigma_p\mathbf{\Phi}$, $\phi_*^T\Sigma_p\mathbf{\Phi}$, or $\phi_*^T\Sigma_p\phi_*$. This turns out to be crucial. Let's define a kernel (or covariance function), $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T\Sigma_p\phi(\mathbf{x}')$. Given Σ_p, we can redefine the features slightly so that the kernel can be rewritten in terms of the redefined functions and interpreted as a scalar product in the feature space, $k(\mathbf{x}, \mathbf{x}') = \tilde{\phi}(\mathbf{x})^T\tilde{\phi}(\mathbf{x}')$. Now comes the paradigm shift that enabled the success of kernel-based methods. Instead of thinking of feature space $\phi(\mathbf{x})$ as primary which induces some kernel, we *first* pick the kernel; *Mercer theorem* says that that induces a feature space (possibly of infinite dimension) where the kernel is a scalar product, but we never have to compute anything in that space, we only need to be able to tractably evaluate kernels, which are functions operating in space X of dimension D. Any algorithm that can be written exclusively in terms of inner products can be made nonlinear, by replacing scalar products with some choice of kernel functions; this is known as the *kernel trick*. The choice of kernel determines, in a non-obvious way, what class of nonlinear relationships can be captured by the regression.

We can rewrite Eq. (15) in a simpler form by making the kernel explicit. Let's define $\mathbf{k}_*$ to be the vector of values between the test point, $\mathbf{x}_*$ and the regression samples $\mathbf{x}$, that is, the vector corresponding to products $\phi_*^T \Sigma_p \mathbf{\Phi}$. Then, the predictive mean and variance of Eq. (15) can be written as

$$\begin{aligned} \mathbb{E}[f_*] &= \mathbf{k}_*^T (K + \sigma^2 \mathbb{I})^{-1} \mathbf{y} && (16) \\ \mathbb{V}[f_*] &= k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T (K + \sigma^2 \mathbb{I})^{-1} \mathbf{k}_*. && (17) \end{aligned}$$

This makes explicit the operations we need to do for kernelized regression: (i) construct the kernel matrix K by executing the kernel function over all pairs of training points, and the kernel vector $\mathbf{k}_*$; (ii) invert the (regularized) kernel matrix; this needs to be done once only for all predictions; (iii) solve for the predicted mean and covariance at any desired set of points, $\mathbf{x}_*$.

- Finally, note that the predictive mean, $\mathbb{E}[f_*]$, the average over the predictive distribution, is a *linear* function of the observations, $\mathbf{y}$. It can also be seen as a weighted average of T kernel functions, if we rewrite Eq. (16) as

$$\mathbb{E}[f_*] = \sum_{t=1}^{T} \alpha_t k(\mathbf{x}_t, \mathbf{x}_*), \tag{18}$$

  and the coefficients α_t can be read out of Eq. (16). This is the case even if the feature space is infinite dimensional, and is a consequence of a mathmatical theorem called the *representer theorem*.

In summary, we have shown how Bayesian inference in linear regression can be "lifted up" to a nonlinear feature space, thus becoming nonlinear regression in $\mathbf{x}$, while retaining some tractability—limited primarily by the construction and inversion of $T \times T$ kernel matrices. What we need to show next is that the same machinery can be viewed as performing an inference over a distribution of smooth nonlinear functions.

5 Gaussian processes for regression

Let's introduce now the Gaussian Processes and connect them to the regression discussions from the previous section. A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution. The process is fully specified by the *mean* and *covariance* functions, and we write

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \tag{19}$$

where

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], && (20) \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]. && (21) \end{aligned}$$

We can think of GP as a prior on the space of functions after we specify the covariance function, k; in this case we often take the mean function to be zero, $m = 0$. Why can GP be thought of as a prior over functions? This is because operationally we often represent functions as a set of $(\mathbf{x}_i, y_i)$ pairs of independent and dependent variables. Clearly, for the specification of an arbitrary function, and infinite number of pairs is needed. But when functions are smooth, and this is what the kernel takes care of, we can represent functions well with a finite number of pairs of points. This finite list—even if very big, so long as it is finite—can be drawn from the GP.

What is the role of the covariance function? If you consider the SE (squared exponential) also known as Gaussian bump functions (see Eq. (13)) or RBF (radial basis functions):

$$k(\mathbf{x}, \mathbf{x}') = \mathrm{Cov}(f(\mathbf{x}), f(\mathbf{x}')) = \alpha \exp\left(-\frac{1}{2\ell^2}||\mathbf{x} - \mathbf{x}'||^2\right) \tag{22}$$

then we see that any two points drawn from the GP that are close in their values of $\mathbf{x}$ are going to be strongly correlated, and points much further away than ℓ will be essentially independent: this prior will thus impose a particular lengthscale on the fluctuations on the function value (it is important to see that the kernel is written in terms of $\mathbf{x}$, but prescribes the correlation structure of function values, f). α modulates the overall width of the distribution, i.e., how tightly the function values will be clustered about the mean function.

What covariance functions are possible? The main constraint is that the covariance kernel function has to be symmetric and positive semidefinite, i.e.,

$$\int d\mathbf{x}d\mathbf{x}'\; k(\mathbf{x}, \mathbf{x}')f(\mathbf{x})f(\mathbf{x}') \geq 0, \tag{23}$$

for all integrable functions f; in particular, for any set of vectors $\{\mathbf{x}_i\}$, the kernel or "Gram" matrix, $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ will be a *covariance matrix*, which is positive semidefinite (has non-negative eigenvalues). There are many standard covariance functions that can be used in applications, see Ref. [**?**] Chapter 4 for an extensive discussion. A few examples additional to SE function are given below: The Rational Quadratic function:

$$k_{RQ}(r) = \left(1 + \frac{r^2}{2\alpha\ell^2}\right)^{-\alpha}, \tag{24}$$

where r is the Euclidean distance between $\mathbf{x}$ and $\mathbf{x}'$ and ℓ and α are the parameters of the kernel function. This has algebraic (long-tailed) decay with distance, which can be obtained as a mixture of SE kernel functions with different length scales.

Exponential covariance function,

$$k(r) = \exp\left(-\left(\frac{r}{\ell}\right)^{\gamma}\right), \tag{25}$$

is a generalization of SE kernel. For $\gamma = 1$ this is interesting since it defines in 1D an Ornstein-Uhlenbeck process (or a mean-reverting random walk), which is not smooth (not differentiable); GP defines smooth functions with $\gamma = 2$ (SE kernel).

Polynomial covariance function:

$$k(\mathbf{x}, \mathbf{x}') = \left(\sigma_0^2 + \mathbf{x}^T \Sigma_p \mathbf{x}'\right)^p, \tag{26}$$

which for $p = 1$ and $\sigma_0 = 0$ reduces to the normal dot product. In the simpler case with $\sigma_0^2 = 0$ it is actually easy to see what is the feature space that this kernel corresponds to; taking

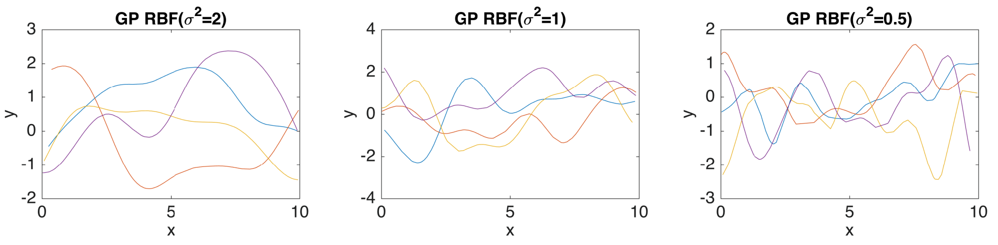

Figure 1: Each panel shows three example draws of functions from the GP prior (more specifically, 100 (x, y) pairs with x uniformly sampling its range). Panels differ by degree of smoothness, ℓ^2, of the SE covariance function.

$p = 2$ in $D = 2$ dimensions, $\phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)^T$, so that $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}')$. Kernel functions can, in general, also be combined: sums and products of kernels are new valid kernels, for instance.

What is the link between the regression discussed previously and Gaussian processes? If we think of a probabilistic regression model, $f(\mathbf{x}) = \phi(\mathbf{x})^T\mathbf{w}$, where $\mathbf{w}$ are the weights that are distributed according to the prior $P(\mathbf{w}) = \mathcal{N}(0, \Sigma_p)$, then this model itself—without using it for inference, i.e., without specifying data $\mathcal{D}$—corresponds to a GP. This can be seen by computing the mean and variance of f:

$$\mathbb{E}[f(\mathbf{x})] = \phi(\mathbf{x})^T\mathbb{E}[\mathbf{w}] = 0 \tag{27}$$

$$\mathbb{E}[f(\mathbf{x})f(\mathbf{x}')] = \phi(\mathbf{x})^T\mathbb{E}[\mathbf{w}\mathbf{w}^T]\phi(\mathbf{x}') = \phi(\mathbf{x})\Sigma_p\phi(\mathbf{x}'), \tag{28}$$

and realizing that since f in the regression is seen as a linear combination of Gaussian-distributed weights, the draws from f are also (jointly) Gaussian.

How can we draw the functions from the GP prior in practice? This is quite easy: first, choose a set of N_* points, $X_* = [\mathbf{x}_{*,1}, \ldots, \mathbf{x}_{*,N_*}]$ at which this function will be represented. By properties of the GP, the function values $\mathbf{f}_*$ need to be jointly distributed with zero mean (for example), and a covariance matrix, which is given by the covariance kernel $K(X_*, X_*)$:

$$\mathbf{f}_* \sim \mathcal{G}(0, K(X_*, X_*)), \tag{29}$$

where we used the notation $K(X_*, X_*)_{ij} = k(\mathbf{x}_{*,i}, \mathbf{x}_{*,j})$. In practice, we need to draw N_* correlated Gaussian RNs with a prescribed covariance structure. One of the easiest methods to do this is by Cholesky decomposition. Let's write $K(X_*, X_*) = L^TL$, where L is a lower-triangular matrix. Since K is a covariance matrix, this decomposition always exists. It is easy to check that if y are IID Gaussian random numbers, then $f_* = Ly$ will have the desired covariance structure, giving us a prescription for generating f_*. Examples of such functions (evaluated at many test points) for different degrees of smoothness using the Gaussian bump (RBF, SE) kernels are shown in Fig. 1.

How can the GP framework be used for inference? Now we think of points $\mathbf{x}$ split into two sets: the set X where we are given the corresponding regression values in Y, and the set X_* where we want to make predictions. Jointly, the distribution over X and Y will be a GP prior, with zero mean and given covariance. But what we are interested in is the distribution over the set X_* *given* data $\mathcal{D} = (X, Y)$. This is exactly the predictive distribution, but we skipped the whole formalism of its derivation based on regression in parametric models. Here, we know that if we use a GP prior, the conditional (predictive) distribution will also be a Gaussian, given by

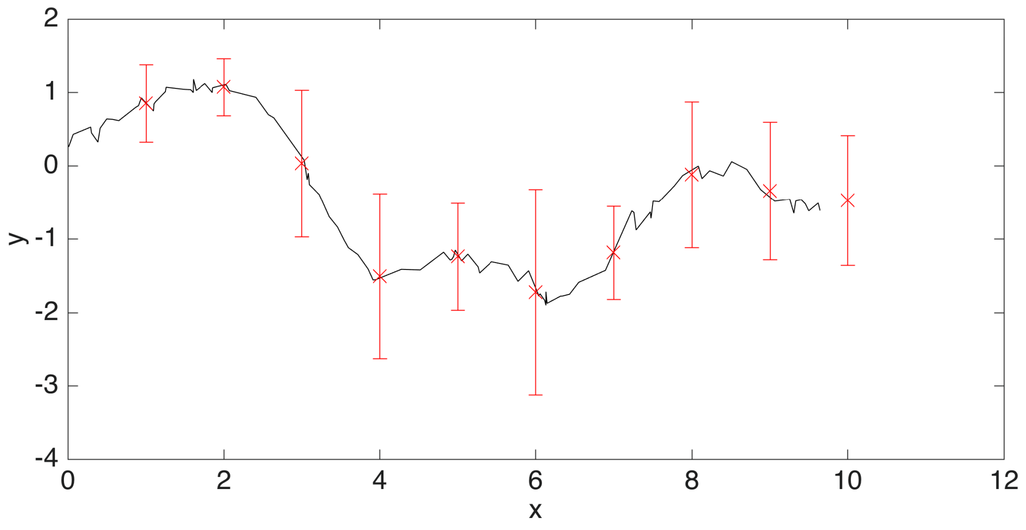

Figure 2: 100 data points (training data) were first drawn from GP with SE kernel with smoothness $\ell = 1$ and noise variance $\sigma^2 = 0.1$. 10 independent predictions are shown in red, with associated predictive variance (error bars), using a matching kernel.

a simple marginalization. It is easy to show that

$$\begin{aligned} P(\mathbf{f}_*|X_*, X, \mathbf{y}) \quad &= \quad \mathcal{G}(K(X_*, X)K(X, X)^{-1}\mathbf{y}, & (30) \\ & \quad\quad K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*). & (31) \end{aligned}$$

Intuitively, it is easy to understand that conditioning a zero-mean prior on observed data generates a new Gaussian process, but this time with a non-zero mean: the predictive mean has been "pulled" into the direction of the observed data points.

Equation 30 is almost exactly of the form of predictive mean and variance that we used in kernelized regression, in Eqs.(16-17). The only difference is the presence, in kernelized regression equations, of the $\sigma\mathbb{I}$ term in the inversion of the kernel matrix: indeed, such a term should be added to Eq. (30) setup when we think of application from real data. Eq. (30), as written, conditions on perfect, error-free observations of function values, $y_i = f(\mathbf{x}_i)$. In case of Gaussian IID noise, $y_i = f(\mathbf{x}_i) + \eta$, we should instead use the covariance function:

$$k(y_i, y_j) = k(\mathbf{x}_i, \mathbf{x}_j) + \delta_{ij}\sigma^2, \tag{32}$$

giving us the change of the kernel matrix, $K(X, X) \to K(X, X) + \sigma^2\mathbb{I}$. If we will actually also observe the values corresponding to f_* by a noisy process with the same variance, σ^2, that term should also be added to the predictive variance in Eq. (17). Note that Eq. (30) generates a full joint distribution over predictions, potentially at multiple prediction points. This joint Gaussian doesn't factorize: if predictions are to be made at two close-by points, they need to be correlated as required by the covariance kernel!

Prediction example is shown in Fig. 2. The algorithm is a direct implementation of Eqs. (16-17), where for efficiency Cholesky decomposition of $(K + \sigma^2\mathbb{I})$ is used for matrix inversion; in this basic setting, GP regression can be implemented literally in only a few lines of code.

How can we assess model fits (for different kernels) and fix the parameters of the kernel functions?

First, we can use a fully Bayesian framework. In this framework, we think of inferring the weights given data and (hyper)parameters of the kernel function θ, e.g.

$$P(\mathbf{w}|\mathbf{y}, \mathbf{X}, \theta) = \frac{P(\mathbf{y}|\mathbf{X}, \mathbf{w})P(\mathbf{w}|\theta)}{P(\mathbf{y}|\mathbf{X}, \theta)}, \tag{33}$$

where the normalization, called *marginal likelihood* or *evidence*, is

$$P(\mathbf{y}|\mathbf{X}, \theta) = \int d\mathbf{w}\ P(\mathbf{y}|\mathbf{X}, \mathbf{w})P(\mathbf{w}|\theta). \tag{34}$$

This quantity is independent of the model parameters. One can repeat this step by integrating over the hyper-parameters, if there are any, to obtain:

$$P(\mathbf{y}|\mathbf{X}) = \int d\theta\ P(\mathbf{y}|\mathbf{X}, \theta)P(\theta); \tag{35}$$

this can be seen as a normalization to the posterior over hyperparameters, $P(\theta|\mathbf{y}, \mathbf{X})$.

One could consider also discrete model classes, which specify which hyperparameters to use (if we want to do comparison between different models); in this case all the above expressions would be also explicitly conditioned on a discrete model class, $\mathcal{H}$; the last marginal likelihood, Eq. (35), would then be conditional on $\mathcal{H}$, and one could compute the posterior over $\mathcal{H}$ to find which model class the data supports best.

In the fully Bayesian framework, marginal likelihood implements all the model fit / complexity tradeoffs: models with higher marginal likelihood, even if they have more parameters, should be preferred over models with lower marginal likelihood, and the model class with the highest posterior should be the one chosen. In fact, we have seen this argument once before: the BIC criterion that we introduced when we discussed maximum likelihood fitting is precisely this construction (where we integrated out all the parameters for the polynomial fitting and only looked at the marginal likelihood of the polynomial degree). There, model complexity (degree) was automatically penalized.

If all the above integrals were tractable, one could thus do model comparison by comparing marginal likelihoods. Similarly, in predictive distributions, one could integrate over the posterior distribution of hyperparameters, $P(\theta|\mathbf{y}, \mathbf{X})$, etc. But since these integrals are often intractable, one can make various types of approximations, for instance, (i) pick the value of hyperparameters that maximizes the posterior (if there are many hyperparameters, this may again overfit); (ii) approximate the distribution over hyperparameters using Laplace (steepest-descent) approximation, i.e., a Gaussian around the most likely value, to carry out the integral analytically; (iii) use cross-validation to set the values for hyperparameters, by maximizing the predictive probability over test data (in this case, fast methods exist for not only k-fold crossvalidation but even leave-one-out (LOO) cross-validation).

An attractive feature of the Gaussian Processes is that the marginal likelihood is analytically tractable (since it is an integration over a high-dimensional Gaussian for weights $\mathbf{w}$):

$$\log P(\mathbf{y}|\mathbf{X}, \theta) = -\frac{1}{2}\mathbf{y}^T \tilde{K}^{-1}\mathbf{y} - \frac{1}{2}\log\det\tilde{K} - \frac{T}{2}\log 2\pi. \tag{36}$$

where $\tilde{K} = K + \sigma^2\mathbb{I}$ and K is the kernel matrix of the data; note that this marginal likelihood still depends on the hyperparameters, θ. As in BIC, the first term corresponds to the goodness of the fit, while the other terms penalize the model complexity. Concretely, one can set the hyperparameters (e.g, the smoothing scale ℓ of the SE kernel and its magnitude) by, e.g., maximizing the marginal likelihood of the data.

6 Study literature

- These notes mainly follow Ref [1], but the treatment in Ref [2] is very similar.

7 Homework

References

[1] Rasmussen CE, Williams CKI (2006) Gaussian Processes for Machine learning. MIT Press.

[2] MacKay DJC (1998) Introductiopn to Gaussian processes. CM Bishop, ed. Neural Networks and Machine Learning, NATO ASI Series 168. Springer, Berlin.