2018-06-04

1 Kernels

Kernels are key to many data analysis methods. A kernel is a function of two arguments, and often we think of $k(x, y)$ as a measure of how similar or related the objects x and y are to each other. But there are several stories we tell that describe different ways to think about kernels. We discuss four such approaches in this lecture:

1. Kernels come from composing linear methods (e.g. linear regression) with *feature maps*, nonlinear functions that map from the original function domain (say $\mathbb{R}^n$) into space of much higher dimension ($\mathbb{R}^N$). In this story, the kernel represents an inner product between feature vectors.

2. Kernels define *basis functions for approximation spaces*. Unlike some schemes to approximate from a fixed function space (e.g. low-degree polynomials), kernel methods are *adaptive*, allowing more variation in regions of the function domain where we have more data.

3. Kernels are associated with a *quadratic form on a space of functions*, which we think of as "energy." Sometimes, as in the case of cubic splines and thin plate splines, the energy and kernel methods for regression minimize this quadratic form subject to data constraints. Thinking of kernel methods in this way gives us one framework for understanding the error in kernel methods.

4. Kernels are also used to represent the *covariance of random processes* in general, and Gaussian processes in particular. In this view of kernel-based regression, we start with a prior distribution over functions, and then use Bayes rule to get a posterior distribution based on measurements of the function.

In each case, using a kernel leads us in the end to a linear algebra problem: all the nonlinearity in a kernel method is summarized in the kernel construction.

These notes are strongly influenced by the Acta Numerica article "Kernel techniques: From machine learning to meshless methods" by Robert Schaback and Holger Wendland. This survey article runs to 98 pages, but if you have the time and inclination to read more deeply, I strongly recommend it s a starting point!

1.1 Some common examples

The choice of an appropriate kernel is the key to the success or failure of a kernel method. But we often lack the insight to make an inspired choice of kernels, and so fall back on a handful of standard families of kernels. Let us mention a few that are commonly used in function approximation on $\mathbb{R}^n$.

Squared exponential The *squared exponential* kernel has the general form

$$k_{\text{SE}}(x,y) = s^2 \exp\left(-\frac{1}{2}(x-y)^T P^{-1}(x-y)\right),$$

for positive s and positive definite P. Most often we choose $P = \ell^2 I$, i.e.

$$k_{\text{SE}}(x,y) = s^2 \exp\left(-\frac{1}{2}\left(\frac{\|x-y\|}{\ell}\right)^2\right).$$

The *scale factor* s and the *length scale* ℓ are examples of kernel *hyper-parameters*. In the case where we use a single length scale parameter (rather than a more general P), the squared exponential kernel is an example of a *radial basis function*, i.e. a kernel that depends only on the distance between the two arguments. In some corners of machine learning, this kernel is referred to as "the" radial basis function, but we will avoid this usage in deference to the many other useful radial basis functions in the world.

Exponential The (absolute) *exponential kernel* has the form

$$k_{\text{exp}}(x,y) = s^2 \exp\left(-\frac{\|x-y\|}{\ell}\right)$$

where s and ℓ are again the scale factor and length scale hyper-parameters. As with the squared exponential kernel, there is a more general form in which $\|x-y\|/\ell$ is replaced by $\|x-y\|_{P^{-1}}$ for some positive definite P matrix. Where the squared exponential function is smooth, the exponential kernel is only continuous — it is not differentiable. This has important implications in modeling, as the function approximations produced by kernel methods inherit the smoothness of the kernel. Hence, a smooth kernel (like the squared exponential) is good for fitting smooth functions, while a non-differentiable kernel (like the absolute exponential) may be a better choice for fitting non-differentiable functions.

Cubic splines and thin plate splines At the end of the last lecture, we saw that we can write *cubic splines* for 1D function approximation in terms of the kernel $k(x,y) = |x-y|^3$. For functions on $\mathbb{R}^2$, the analogous choice is the *thin plate spline* with kernel $r \log r$ for $r = \|x-y\|$. Unlike the squared exponential and exponential kernels, we express these kernels in terms of the distance r and not a scaled distance r/ℓ, as approximation methods using cubic and thin plate splines are *scale invariant*: scaling the coordinate system does not change the predictions. Also unlike the squared exponential and exponential kernels, these kernels *grow* with increasing values of r, and so our intuition that kernels measure "how similar things are" founders a bit on these examples. Other ways of explaining kernels that are more appropriate to describing why these choices are useful.

1.2 Definitions and notation

A *kernel* on a set $\mathcal{X}$ is a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. We will restrict our attention to *symmetric* kernels, for which $k(x,y) = k(y,x)$. Some kernels depend on additional *hyper-parameters*, e.g. the length scale and smoothness parameters described for squared exponential and Matérn kernels in the previous section.

Often the set $\mathcal{X}$ has special structure, and we want kernels to have invariants that are natural to that structure. For $\mathcal{X} = \mathbb{R}^d$, we like to think about invariance under translation and rotation. We say k is *stationary* if it is invariant under translation, i.e. $k(x+u, y+u)$ for any u. If $k(x,y)$ depends only on x and the distance between x and y, we say it is *isotropic*. When k is both stationary and isotropic, we often identify it with a *radial basis function* ϕ, i.e. $k(x,y) = \phi(\|x-y\|)$.

For ordered $X, Y \subset \mathcal{X}$, we write the matrix of pairwise evaluations as

$$(K_{XY})_{ij} = k(x_i, y_j).$$

We say K_{XX} is the *kernel matrix* for the set X and kernel k. The kernel function k is *positive definite* if K_{XX} is positive definite whenever X consists of distinct points. The squared exponential and absolute exponential kernels are positive definite; the cubic spline and thin plate spline kernels are not.

A kernel is *conditionally positive definite* relative to a space of functions $\mathcal{U}$ from $\mathcal{X} \to \mathcal{R}$ if

$$v^T K_{XX} v \geq 0 \text{ whenever } v \neq 0 \text{ and } \forall u \in \mathcal{U},\ u_X^T v = 0.$$

A kernel on $\mathcal{X} = \mathbb{R}^n$ is *conditionally positive definite of order* d if it is conditionally positive definite relative to the space $\mathcal{P}_{d-1}$ of polynomials of total degree at most $d-1$. The cubic spline and thin plate spline kernels are both conditionally positive definite of order 2.

2 Feature maps

Suppose we want to approximate $f : \mathbb{R}^n \to \mathbb{R}$. A very simple scheme is to approximate f by a *linear* function,

$$f(x) \approx \hat{f}(x) = c^T x$$

where the coefficients c are determined from $m \geq n$ samples by a least squares fitting method. That is, we solve

$$\text{minimize } \|X^T c - f_X\|^2$$

where X is a matrix whose columns are data points $x_1, \ldots, x_m$ and f_X is a vector of function values $f(x_1), \ldots, f(x_m)$.

Unfortunately, the space of linear functions is rather limited, so we may want a richer class of models. The next step up in complexity would be to look at a vector space $\mathcal{H}$ of functions from $\mathbb{R}^n \to \mathbb{R}$, with basis functions $\psi_1, \ldots, \psi_N$ that we collect into a single vector-valued function ψ. For example, for one-dimensional function approximation on $[-1, 1]$, we might choose a polynomial space of approximating functions $\mathcal{H} = \mathcal{P}_{N-1}$ and use the Chebyshev basis functions $T_0(x), \ldots, T_{N-1}(x)$. I think of these as basis vectors for a space of candidate approximating functions, but in the language of machine learning, we say that the functions ψ_i are *features* and the vector-valued function ψ is a *feature map*. We write our approximation as

$$\hat{f}(x) = c^T \psi(x) = \sum_{j=1}^{N} c_j \psi_j(x),$$

and if we have function values at $m \geq N$ points, we can again fit the coefficients c by the least squares problem

$$\text{minimize} \|\Psi^T c - f_X\|^2$$

where $[\Psi]_{ij} = \psi_i(x_j)$. All nonlinearity in the scheme comes from the nonlinear functions in ψ; afterward, we have a standard linear problem.

What if the dimension of our approximating space $\mathcal{H}$ is much larger than the amount of data we have — or even if it is infinite dimensional? Or, equivalently: what if we have more features than training points? In this case, many different approximations in the space all fit the data equally well, and we need a rule to choose from among them. From a linear numerical algebra perspective, a natural choice is the *minimal norm* solution; that is, we approximate $\hat{f}(x) = c^T \psi(x)$ as before, but choose the coefficients to minimize $\|c\|$ subject to $\Psi^T c = f_X$. The solution to this linear system is

$$c = \Psi(\Psi^T \Psi)^{-1} f_X$$

and therefore

$$\hat{f}(x) = \psi(x)^T \Psi(\Psi^T \Psi)^{-1} f_X.$$

We now observe that all the entries of the vector $\psi(x)^T \Psi$ and the Gram matrix $\Psi^T \Psi$ can be written in terms of inner products between feature vectors. Let $k(x, y) = \psi(x)^T \psi(y)$; then

$$\hat{f}(x) = k_{xX} K_{XX}^{-1} f_X$$

where k_{xX} denotes the row vector with entries $k(x, x_i)$ and $[K_{XX}]_{ij} = k(x_i, x_j)$. The function $k(x, y) = \psi(x)^T \psi(y)$ is the kernel function, and this way of writing the approximation only in terms of these inner products, without writing down a feature map, is sometimes called the "kernel trick."

So far we have shown how to get from feature maps to kernels; what if we want to go in the other direction? As it happens, if we are given a positive (semi)definite kernel function, we can always construct a (possibly infinite) feature map associated with the kernel. To do this, we define an integral operator on an appropriate space of distributions g

$$[\mathcal{K}g](x) = \int k(x, y) g(y) \, dy.$$

This encodes all the information about the kernel — formally, for example, if we let g be a Dirac delta at x_i, then $[\mathcal{K}g](x) = k(x, x_i)$. *Mercer's theorem* tell us there is an eigenvalue decomposition with eigenpairs $(\lambda_j, v_j(x))$ so that

$$[\mathcal{K}g](x) = \sum_{j=1}^{\infty} \lambda_j v_j(x) \left[\int v_j(y) g(y) \, dy \right],$$

and the features $\psi_j(x) = \sqrt{\lambda_j} v_j(x)$ give us the kernel.

3 Kernels as basis functions

As we have seen in the previous section, a standard idea for function approximation is to choose a function from some fixed approximation space $\mathcal{H}$. For example, if we define the function

$$\phi(r) = \exp\left(-\frac{1}{2}\left(\frac{r}{l}\right)^2\right),$$

then a reasonable space for functions on $[0,1]$ might consist of approximants that are combinations of these radial basis function "bumps" centered at each of the nodes j/N for $j = 0, \ldots, N$:

$$\hat{f}(x) = \sum_{j=0}^{N} c_j \phi(x - j/N).$$

If we have data on a set of $N+1$ points X, and X' denotes the points $0, 1/N, \ldots, N$, then the interpolation equations $\hat{f}_X = f_X$ take the form

$$\hat{f}(x_i) = \sum_{j=0}^{n} \phi(x_i - x'_j) c_j = f(x_i)$$

which we write in matrix form as

$$K_{XX'} c = f_X$$

where $k(x,y) = \phi(|x-y|)$. With more data points, we use least squares.

What if the data points are not uniformly distributed on $[0,1]$? For example, what if we have more data points close to 1 than we have close to 0? The choice of a fixed approximation space limits us: we have no way of saying that with more data close to 1, we should allow the approximation more wiggle room to fit the function there. We can do this by putting more basis functions that are "centered" in the area where the data points are dense, e.g. by making the data points and the centers coincide:

$$\hat{f}(x) = \sum_{j=1}^{N} c_j \phi(|x - x_j|).$$

Then the interpolation conditions are

$$\sum_{j=1}^{N} \phi(|x_i - x_j|)c_j = f(x_i).$$

or $K_{XX}c = f_X$. By adapting the space to allow more flexibility close to where the data is, we hope to get better approximations, but there is another advantage as well: if the kernel is positive definite then the matrix K_{XX} is symmetric and positive definite, and we need not worry about singularity of the linear system.

Of course, nothing says we are not allowed to use an approximation space that is adapted to the sample points *as well as* a fixed approximation space. We saw this already in our discussion of cubic splines, where we had a piece associated with the cubic radial basis function together with a linear "tail":

$$\hat{f}(x) = \sum_{j=1}^{m} c_j|x - x_j|^3 + d_1 + d_2 x.$$

In order to uniquely solve the linear system, we need some additional constraints; for cubic splines, we use the *discrete orthogonality condition*

$$\sum_j c_j p(x_j) = 0, \text{ any linear } p(x).$$

More generally, if a kernel is *conditionally positive definite* relative to a space of functions $\mathcal{U}$ spanned by the basis $p_1(x), \ldots, p_{m'}(x)$, then the coefficients of the approximation

$$\hat{f}(x) = \sum_j c_j k(x, x_j) + \sum_j d_j p_j(x)$$

are determined by the interpolation conditions and the discrete orthogonality condition

$$\sum_j c_j u(x_j) = 0, \forall u \in \mathcal{U}.$$

We can write these conditions in matrix form as

$$\begin{bmatrix} K_{XX} & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} f_X \\ 0 \end{bmatrix}$$

where $[P]_{ij} = p_j(x_i)$. So long as P is full rank (we call this well-posedness of the points for interpolation in $\mathcal{U}$), this linear system is invertible. Most often, we include polynomial tails to guarantee solvability of the interpolation problem with conditionally positive definite kernels, but nothing prevents us from incorporating such terms in other approximations as well — and, indeed, it may do our approximations a great deal of good.

4 Kernels and quadratic forms

4.1 From feature maps to RKHS

Let us return again to the feature map picture of kernels: we have an approximation space $\mathcal{H}$, which we will assume for the moment is finite dimensional, with a basis $\psi_1(x), \psi_2(x), \ldots, \psi_N(x)$. From this space, we seek an approximation of the form

$$\hat{f}(x) = \sum_i c_i \psi_i(x)$$

so that $\sum_i c_i^2$ is minimal subject to the data constraints. Hidden in this construction is that we have implicitly defined an *inner product* for the space of functions $\mathcal{H}$ for which the $\{\psi_i\}_{i=1}^N$ basis is orthonormal; that is,

$$\langle \psi_i, \psi_j \rangle_{\mathcal{H}} = \delta_{ij},$$

and we can rephrase the approximation problem without direct reference to the expansion coefficients as

$$\text{minimize } \|\hat{f}\|_{\mathcal{H}}^2 \text{ s.t. } \hat{f}_X = f_X.$$

We can also express evaluation of $\hat{f}$ at a point in terms of the inner product; if we define $k_y \in \mathcal{H}$ by

$$k_y(x) = \sum_{i=1}^N \psi_i(y)\psi_i(x),$$

then

$$\begin{aligned}
\left\langle \hat{f}, k_y \right\rangle_{\mathcal{H}} &= \left\langle \sum_{i=1}^N c_i \psi_i, \sum_{j=1}^N \psi_j(y)\psi_j \right\rangle_{\mathcal{H}} \\
&= \sum_{i,j} c_i \psi_j(y) \langle \psi_i, \psi_j \rangle_{\mathcal{H}} = \sum_{i=1}^N c_i \psi_i(y) = \hat{f}(y).
\end{aligned}$$

This idea applies more generally: we can write point evaluation at y for any function $g \in \mathcal{F}$ as $\langle g, k_y \rangle_{\mathcal{H}}$, where the feature vector $\psi(y)$ gives the coefficients for the evaluation function k_y. Note that $k(x,y) = \langle k_x, k_y \rangle_{\mathcal{H}} = \psi(x)^T \psi(y)$.

So far, we have only described what happens with finite-dimensional vector spaces of approximating functions. But though there are some technicalities that we must deal with in the infinite-dimensional case, the finite-dimensional picture sets the stage for the more general definition. A space of functions $\mathcal{H}$ is a *reproducing kernel Hilbert space* (RKHS) if function evaluation can be written in terms of the inner product:

$$g(x) = \langle g, k_x \rangle_{\mathcal{H}} \text{ for any } g \in \mathcal{H},$$

where $k_x(y) = k(x,y) = k(y,x) = k_y(x)$ is the associated (reproducing) kernel, so called because it reproduces function evaluations. Given an orthonormal basis (a feature map) for $\mathcal{H}$, we can write the inner product on $\mathcal{H}$ and the inner product. Of course, we often would prefer not to use feature maps for concrete computation; is there another way to get our hands on the inner product? The answer, as it turns out, is yes.

4.2 From kernels to inner products

Every RKHS has an associated kernel function. We now sketch the path going in other direction: given a positive definite kernel function, reconstruct the RKHS. The key observation is that for any finite set of points X,

$$\left\langle \sum_i c_i k_{x_i}, \sum_j d_j k_{x_j} \right\rangle_{\mathcal{H}} = \sum_{i,j} c_i d_j k(x_i, x_j) = c^T K_{XX} d.$$

This allows us to write the inner products between any functions that can be expressed as a linear combination of kernel shapes centered at any *finite* number of points. This is a rich set of functions, almost rich enough to get a full RKHS. It is not quite enough: technically, the set of functions that can be written in terms of a finite number of centers is a *pre-Hilbert space*; to get the rest of the way to a RKHS, we need to "complete" the space by including limits of Cauchy sequences. The RKHS constructed in this way is sometimes called the *native space* for the associated kernel or radial basis function. Unfortunately, it is not always easy to characterize the functions in the native space for a kernel! The native space for the squared exponential kernel is small, consisting of only very smooth functions, while the native spaces of some other kernels are larger.

4.3 Error analysis of kernel interpolation

Let us approximate $f \in \mathcal{H}$ by kernel interpolation at $X = (x_1, \ldots, x_n)$, i.e.

$$\text{minimize } \|\hat{f}\|_{\mathcal{H}}^2 \text{ s.t. } \hat{f}_X = f_X.$$

What is the error $|f(y) - \hat{f}(y)|$ at a new point y? Our approach looks like the approach to error analysis for polynomial interpolation:

1. Define a function $\tilde{f}$ by interpolating at one more point.
2. Use regularity to control the difference between $\tilde{f}$ and $\hat{f}$.

Specifically, let $\tilde{f}$ interpolate at $X' = (x_1, \ldots, x_n, y)$:

$$\text{minimize } \|\tilde{f}\|_{\mathcal{H}}^2 \text{ s.t. } \tilde{f}_X = f_X \text{ and } \tilde{f}(y) = f(y).$$

Because $\hat{f}$ and $\tilde{f}$ involve the same optimization objective, but with more constraints for $\tilde{f}$ (and f involves a limiting case of even more constraints), we have

$$\|\hat{f}\|_{\mathcal{H}}^2 \leq \|\tilde{f}\|_{\mathcal{H}}^2 \leq \|f\|_{\mathcal{H}}^2.$$

The advantage of this is that we can write the norms in this inequality in a nice way. Observe that $e = \tilde{f} - \hat{f}$ is a kernel interpolant with centers at X' and coefficients $d = K_{X'X'}^{-1} e_{X'}$;

$$\|e\|_{\mathcal{H}}^2 = d^T K_{X'X'} d = e_{X'}^T K_{X'X'}^{-1} e_{X'}$$

Now we use the fact that $e_{X'}$ is zero except in the last component $e(y)$, and we know how to write the last diagonal element of the inverse of a matrix:

$$\|e\|_{\mathcal{H}}^2 = [K_{X'X'}^{-1}]_{yy} e(y)^2$$

Therefore, we have

$$|e(y)| = P_X(y)\|e\|_{\mathcal{H}}, \quad \text{where } P_X(y)^2 = k_{yy} - k_{yX} K_{XX}^{-1} k_{Xy}.$$

The function P is known as the *power function* in some communities. Now, we observe that $\langle e, \hat{f} \rangle_{\mathcal{H}} = 0$ by construction, so the Pythagorean theorem gives us

$$\|\hat{f}\|_{\mathcal{H}}^2 + \|e\|_{\mathcal{H}}^2 = \|\tilde{f}\|_{\mathcal{H}}^2.$$

Combining with the bound $\|\tilde{f}\|_{\mathcal{H}} \leq \|f\|_{\mathcal{H}}$, we have

$$\|e\|_{\mathcal{H}}^2 \leq \|f\|_{\mathcal{H}}^2 - \|\hat{f}\|_{\mathcal{H}}^2.$$

Putting all the pieces together, we have the error bound

$$|f(y) - \hat{f}(y)| = |e(y)| \leq P_X(y) \sqrt{\|f\|_{\mathcal{H}}^2 - \|\hat{f}\|_{\mathcal{H}}^2}.$$

4.4 Beyond the positive definite

So far in this section, we have only considered positive definite kernels. What about *conditionally* positive definite kernels, like the thin plate spline or cubic spline kernels? For a kernel that is conditionally positive definite with respect to a space $\mathcal{U}$, we define a pre-Hilbert space of interpolants with a tail in $\mathcal{U}$, i.e.

$$\check{\mathcal{H}} = \left\{ g(x) = \sum_{j=1}^{|X|} c_j k(x, x_j) + u(x) : c \in \mathbb{R}^{|X|}, u \in \mathcal{U}, \text{ and } \forall v \in \mathcal{U}, c_X^T v_X = 0 \right\}.$$

We define the quadratic form as in the positive definite case (e.g. $c^T K_{XX} c$), but now the quadratic form is only *semi-definite*, as it will be zero for functions that lie in $\mathcal{U}$. We can complete $\check{\mathcal{H}}$ under this *semi-norm*, and the remainder of the analysis goes forward as with the positive definite case, except with some more technicalities.

There is a reason to think about the conditionally positive definite case, though: the cubic spline is conditionally positive definite, and for the cubic spline we can give a mechanical intuition behind the rather formal-looking error analysis in the previous subsection. A cubic spline corresponds to the interpolant we would get if we bent a thin beam of wood to the shape of our data. This shape minimizes the *bending energy*

$$\mathcal{E}[u] = |u|_{\mathcal{H}}^2 = \frac{1}{2} \int |u''(x)|^2 \, dx.$$

When we interpolate f at points X, we use a certain amount of energy to bend the beam to the right shape; this energy is less than the total bending energy of f. If we want to push the beam at y from its minimal-energy position $\hat{f}(y)$ to a new position $\hat{f}(y) + e(y)$, we use an amount of energy associated with the stiffness times $|e(y)|^2$. This additional energy or work to bend $\hat{f}$ to become $\tilde{f}$ can be no more than the difference in the bending energy of $\hat{f}$ and the bending energy of f.

5 Gaussian processes

Our final story comes from *Gaussian processes* (GP). Informally, just as the ordinary Gaussian distribution is over numbers and multivariate Gaussian

distributions are over vectors, a Gaussian process is a distribution over functions. More formally, a Gaussian process on a set $\mathcal{X}$ with mean field μ and covariance kernel k is a collection of Gaussian random variables indexed by $\mathcal{X}$, any finite subset of which obeys a multi-variate Gaussian distribution; that is, if f is a draw from a Gaussian process, then for any $X \subset \mathcal{X}$,

$$f_X \sim N(\mu_X, K_{XX}).$$

In Bayesian inference using GPs, we start with a *prior* GP from which we assume f is drawn, then compute a *posterior* GP conditioned on data.

Because GPs are defined in terms of the behavior at finite subsets of points, we can really focus on the multivariate normal case. Suppose a multivariate Gaussian random variable Y is partitioned into Y_1 and Y_2. For simplicity, we assume Y has mean zero. Then the distribution is

$$Y \sim N(0, K),$$

i.e. we have the probability density

$$p(y) = \frac{1}{\det(2\pi K)} \exp\left(-\frac{1}{2} y^T K^{-1} y\right).$$

Now, we rewrite the quadratic form $y^T \Omega y$ (where $\Omega = K^{-1}$ is the *precision matrix*) in terms of the components y_1 and y_2:

$$y^T K^{-1} y = y^T \Omega y = (y_2 - z)\Omega_{22}(y_2 - z), \qquad \text{where } z = -\Omega_{22}^{-1}\Omega_{21} y_2$$

Now we rewrite the $(2,1)$ block of the equation $\Omega K = I$ as $\Omega_{21} K_{11} + \Omega_{22} K_{21} = 0$, then rearrange to $-\Omega_{22}^{-1}\Omega_{21} = K_{21} K_{11}^{-1}$, to get the more convenient formula

$$z = K_{21} K_{11}^{-1} y_1.$$

The same approach gives us the Schur complement relation $\Omega_{22}^{-1} = K_{22} - K_{21} K_{11}^{-1} K_{12} = S$. Plugging this formulation of the quadratic into the joint density and dividing out by the marginal for y_1 gives the conditional density

$$p(y_2|y_1) = \frac{p(y_1, y_2)}{\int p(y_1, w)\, dw} = \frac{1}{\sqrt{\det(2\pi S)}} \exp\left(-\frac{1}{2}(y_2 - z)^T S^{-1} (y_2 - z)\right).$$

Thus, the conditional distribution for Y_2 given $Y_1 = y_1$ is again Gaussian:

$$Y_2|Y_1 = y_1 \quad \sim \quad N(z, S).$$

Applying the same logic to Gaussian processes, we find that if f is drawn from a GP with mean field 0 and covariance kernel k, then conditioning on observations at points X gives a new GP with mean and covariance kernel

$$\hat{\mu}(x) = k_{xX} K_{XX}^{-1} f_X \quad \text{and} \quad \hat{k}(x,x') = k(x,x') - k_{xX} K_{XX}^{-1} k_{Xx'}.$$

The conditional mean field $\hat{\mu}(x)$ is exactly the same as the kernel prediction that we have seen derived in other ways, and we might recognize the *preditive variance at* x conditioned on data at X as

$$\hat{k}(x,x) = k(x,x) - k_{xX} K_{XX}^{-1} k_{Xx} = P_X(x)^2,$$

where $P_X(x)$ is the power function for x that we saw in the last section.

6 Deterministic or stochastic?

In the previous two sections, we have developed two ways to think about the error analysis of kernel methods to interpolate f:

1. *Optimal approximation*: suppose $\|f\|_{\mathcal{H}} \leq C$ and let

$$\begin{aligned}\mathcal{F} &= \{g \in \mathcal{H} : \|g\|_{\mathcal{H}} \leq C \text{ and } g_X = f_X\} \\ &= \left\{ \hat{f} + u \in \mathcal{H} : \langle \hat{f}, u \rangle_{\mathcal{H}} = 0 \text{ and } \|u\|_{\mathcal{H}}^2 \leq C^2 - \|\hat{f}\|_{\mathcal{H}}^2 \right\}.\end{aligned}$$

   That is, $\hat{f}$ is the center point of a region that is both consistent with the data constraints and the norm constraints. Because it is at the center of this set, $\hat{f}$ minimizes the *worst possible error* (in the native space norm) over all possible f that are consistent with what we know. To get pointwise error estimates, we look at bounds on $|u(x)|^2$ for all possible u that satisfy $u_X = 0$ and $\|u\|_{\mathcal{H}}^2 \leq C^2 - \|\hat{f}\|_{\mathcal{H}}^2$.

2. *Bayesian inference*: suppose f is drawn from a Gaussian process with some known covariance kernel. Conditioned on the data f_X, we have a new Gaussian process with mean field $\hat{f}$ and a conditional kernel. To get pointwise error estimates, we look at the predictive distribution at a new test point x (itself a Gaussian distribution), including the predictive variance.

The deterministic approach gives us the *best worst-case error* given what we know about the function; the Bayesian approach gives us an *expected value*. Both give the same predictions, and both use the same quantities in computing an error result, though with different interpretations. Both approaches also use information that might not be easy to access (a bound on a native space norm of f, or an appropriate prior distribution).

Why should we not just pick one approach or the other? Apart from the question of modeling assumption, the two approaches yield different predictions if the measurements of f are more complex, e.g. if we have information such as an inequality bound or a nonlinear relationship between point values of f. However, this is beyond the scope of the current discussion.