The Kernel Trick

Consider the assumption of linear regression with an explicit feature transformation $\phi\colon \mathbf{x} \mapsto \phi(\mathbf{x})$:

$$y(\mathbf{x}) = \phi(\mathbf{x})^\top \mathbf{w} + \varepsilon(\mathbf{x}).$$

Given training data $\mathcal{D} = \big\{(\mathbf{x}, y)\big\}_{i=1}^{n} = (\mathbf{X}, \mathbf{y})$, we first apply ϕ to each data point:

$$\boldsymbol{\Phi} = \phi(\mathbf{X}) = \begin{bmatrix} \phi(\mathbf{x}_1)^T \\ \vdots \\ \phi(\mathbf{x}_n)^T \end{bmatrix}.$$

Next, we proceed as we did before by modeling the residuals $\varepsilon(\mathbf{x})$ as zero-mean independent, identically distributed Gaussians with variance σ^2:

$$p(\boldsymbol{\varepsilon}) = \mathcal{N}(\boldsymbol{\varepsilon}; \mathbf{0}, \sigma^2\mathbf{I}),$$

giving rise to the following likelihood:

$$p(\mathbf{y} \mid \boldsymbol{\Phi}, \mathbf{w}, \sigma^2) = \mathcal{N}(\mathbf{y}; \boldsymbol{\Phi}\mathbf{w}, \sigma^2\mathbf{I}).$$

In Bayesian linear regression, we further chose a multivariate Gaussian prior for $\mathbf{w}$:

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}; \boldsymbol{\mu}, \boldsymbol{\Sigma}).$$

For simplicity, in the derivations below, we will assume the prior mean for $\mathbf{w}$ is $\boldsymbol{\mu} = \mathbf{0}$.

Given these assumptions, we can derive the posterior distribution of $\mathbf{w}$ given the (transformed) data $\mathcal{D}$:

$$p(\mathbf{w} \mid \mathcal{D}, \sigma^2) = \mathcal{N}(\mathbf{w}; \boldsymbol{\mu}_{\mathbf{w}\mid\mathcal{D}}, \boldsymbol{\Sigma}_{\mathbf{w}\mid\mathcal{D}}),$$

where

$$\begin{aligned} \boldsymbol{\mu}_{\mathbf{w}\mid\mathcal{D}} &= \boldsymbol{\Sigma}\boldsymbol{\Phi}^\top(\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^\top + \sigma^2\mathbf{I})^{-1}\mathbf{y}; \\ \boldsymbol{\Sigma}_{\mathbf{w}\mid\mathcal{D}} &= \boldsymbol{\Sigma} - \boldsymbol{\Sigma}\boldsymbol{\Phi}^\top(\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^\top + \sigma^2\mathbf{I})^{-1}\boldsymbol{\Phi}\boldsymbol{\Sigma}. \end{aligned}$$

If we wish to use our model to predict the outputs $\mathbf{y}^*$ associated with a set of inputs $\mathbf{X}^*$, we previously derived:

$$p(\mathbf{y}^* \mid \boldsymbol{\Phi}^* = \phi(\mathbf{X}^*), \mathcal{D}, \sigma^2) = \mathcal{N}(\mathbf{y}^*; \boldsymbol{\Phi}^*\boldsymbol{\mu}_{\mathbf{w}\mid\mathcal{D}}, \boldsymbol{\Phi}^*\boldsymbol{\Sigma}_{\mathbf{w}\mid\mathcal{D}}\boldsymbol{\Phi}^{*T} + \sigma^2\mathbf{I}).$$

Examining the forms of these expressions (after plugging in the posteriors $\boldsymbol{\mu}_{\mathbf{w}\mid\mathcal{D}}$ and $\boldsymbol{\Sigma}_{\mathbf{w}\mid\mathcal{D}}$, we see that the feature expansion ϕ always appears in one of the following expressions:

$$\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^\top \qquad \boldsymbol{\Phi}^*\boldsymbol{\Sigma}\boldsymbol{\Phi}^\top \qquad \boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^{*T} \qquad \boldsymbol{\Phi}^*\boldsymbol{\Sigma}\boldsymbol{\Phi}^{*T}.$$

The entries of these matrices are always of the form $\phi(\mathbf{x})^\top\boldsymbol{\Sigma}\phi(\mathbf{x}')$, where $\mathbf{x}$ and $\mathbf{x}'$ are two arbitrary inputs. To simplify our expressions, we can define a function

$$K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top\boldsymbol{\Sigma}\phi(\mathbf{x}').$$

Because $\mathbf{\Sigma}$ is positive definite, it has a "matrix square root", $\mathbf{\Sigma}^{1/2}$ with the property $(\mathbf{\Sigma}^{1/2})^2 = \mathbf{\Sigma}$.[1] If we define the function $\psi(\mathbf{x}) = \mathbf{\Sigma}^{1/2}\phi(\mathbf{x})$, we can see that K is simply an inner product:

$$K(\mathbf{x}, \mathbf{x}') = \psi(\mathbf{x})^\top \psi(\mathbf{x}').$$

Such a function K is called a **kernel** or **covariance function**. Valid kernel functions are guaranteed to always produce positive semi-definite Gram matrices: a **Gram matrix** is a square matrix of inner products between pairs of elements. Formally, given a set of vectors

$$X = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}$$

the Gram matrix

$$K(X, X) = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & \cdots & K(\mathbf{x}_1, \mathbf{x}_n) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) & \cdots & K(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(\mathbf{x}_n, \mathbf{x}_1) & K(\mathbf{x}_n, \mathbf{x}_2) & \cdots & K(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

$$= \begin{bmatrix} \psi(\mathbf{x}_1)^\top \psi(\mathbf{x}_1) & \psi(\mathbf{x}_1)^\top \psi(\mathbf{x}_2) & \cdots & \psi(\mathbf{x}_1)^\top \psi(\mathbf{x}_n) \\ \psi(\mathbf{x}_2)^\top \psi(\mathbf{x}_1) & \psi(\mathbf{x}_2)^\top \psi(\mathbf{x}_2) & \cdots & \psi(\mathbf{x}_2)^\top \psi(\mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ \psi(\mathbf{x}_n)^\top \psi(\mathbf{x}_1) & \psi(\mathbf{x}_n)^\top \psi(\mathbf{x}_2) & \cdots & \psi(\mathbf{x}_n)^\top \psi(\mathbf{x}_n) \end{bmatrix}$$

is positive semi-definite $\forall$ sets X.

Sometimes it is possible to specify a covariance function K directly without ever computing the feature map explicitly. With such a function, we could perform efficient Bayesian linear regression even with a high-dimensional (or infinite dimensional) feature expansion ϕ *implicitly.* This idea of computing inner products in a feature space directly is called the **kernel trick** and has been the basis of a large amount of work in the machine-learning community. Effectively, any algorithm that operates purely in terms of inner products between input vectors can be made nonlinear by replacing normal inner products with the evaluation of a kernel.

With this definition, we may rewrite the predictive distribution for $\mathbf{y}^*$:

$$p(\mathbf{y}_* \mid \mathbf{X}^*, \mathcal{D}, \sigma^2) = \mathcal{N}(\mathbf{y}^*; \mu_{\mathbf{y}^*|\mathcal{D}}, K_{\mathbf{y}^*|\mathcal{D}}),$$

where

$$\begin{aligned} \boldsymbol{\mu}_{\mathbf{y}^*|\mathcal{D}} &= K(\mathbf{X}^*, \mathbf{X})\big(K(\mathbf{X}, \mathbf{X}) + \sigma^2\mathbf{I}\big)^{-1}\mathbf{y}; \\ K_{\mathbf{y}^*|\mathcal{D}} &= K(\mathbf{X}^*, \mathbf{X}^*) - K(\mathbf{X}^*, \mathbf{X})\big(K(\mathbf{X}, \mathbf{X}) + \sigma^2\mathbf{I}\big)^{-1}K(\mathbf{X}, \mathbf{X}^*) + \sigma^2\mathbf{I}^*. \end{aligned}$$

[1] You can prove this via the singular value decomposition (SVD): $\mathbf{\Sigma} = \mathbf{U}\mathbf{D}\mathbf{U}^\top$, where $\mathbf{U}$ is unitary and $\mathbf{D}$ is diagonal with positive entries (because $\mathbf{\Sigma}$ is positive definite), then $\mathbf{\Sigma}^{1/2} = \mathbf{U}\mathbf{D}^{1/2}\mathbf{U}^\top$ as $\mathbf{U}^T\mathbf{U} = \mathbf{I}$.

Examples

Perhaps the most-commonly used kernel is the **squared exponential** covariance function:

$$K(\mathbf{x}, \mathbf{x}'; \lambda, \ell) = \lambda^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\ell^2}\right),$$

where λ and ℓ are hyperparameters that control the covariance function's behavior. The former is simply a multiplicative scaling constant (you can think of this as an implicit scalar multiplication in the implicit feature map ϕ). The latter takes the role of a **length scale;** vectors separated by more than a couple length scales will have a kernel value near zero.

An example of Bayesian linear regression using this kernel function is shown in Figure 1. We see that the use of this kernel function allowed us to achieve nice nonlinear regression without computing explicit basis expansions. In fact, you can show that the squared exponential kernel corresponds to an *infinite-dimensional* basis expansion, where we use a Gaussian basis function centered on every point. Such a feature expansion would be impossible to use if we attempted to use explicit feature computation.

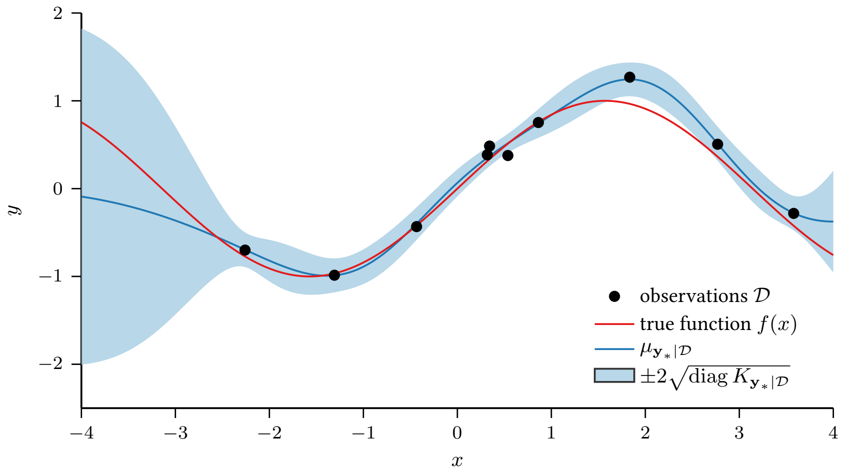

Figure 1: Example of Bayesian linear regression using the squared exponential covariance function. The true function is $f = \sin(x)$. The kernel parameters are $\lambda = \ell = 1$, and the noise variance was set to $\sigma^2 = 0.1^2$.

Sometimes thinking in terms of the kernel can help even when you have an explicit feature expansion on hand. As an example, imagine our inputs are binary vectors of length n (so each input $\mathbf{x}$ is a subset, a member of the power set $\mathcal{P}(n)$). One rather expensive feature expansion we could try would be to enumerate every member of $\mathcal{P}(n)$ and define $\phi(\mathbf{x})_i = \mathbf{s}_i \subset \mathbf{x}$, where $\mathbf{s}_i$ is the i^{th} element of the power set. So we represent our set $\mathbf{x}$ by a feature vector of length 2^n indicating every subset of $\mathbf{x}$. This is a very expensive feature expansion, requiring exponential space to store for each input. However, if we take $\mathbf{\Sigma} = \mathbf{I}$, we can compute the dot product as:

$$K(\mathbf{x}, \mathbf{x}') = 2^{|x \cap x'|},$$

which only requires time and space linear in n!