Gaussian Processes

Robin Aldridge-Sutton

Definition

A Gaussian process (GP) is a stochastic process (a distribution over functions) such that for any finite set of input values the function values have a multivariate Gaussian distribution, e.g. for $\mathbf{x} \in \mathbb{R}^k$,

$$f(\mathbf{x}) \sim N(\mu(\mathbf{x}) = \mathbf{0}, \Sigma = K(\mathbf{x}, \mathbf{x})),$$
$$K(\mathbf{x}, \mathbf{x}')_{i,j} = k(x_i, x'_j),$$

where $k(x, x')$ is called the covariance or kernel function, and can be any positive definite function, e.g.

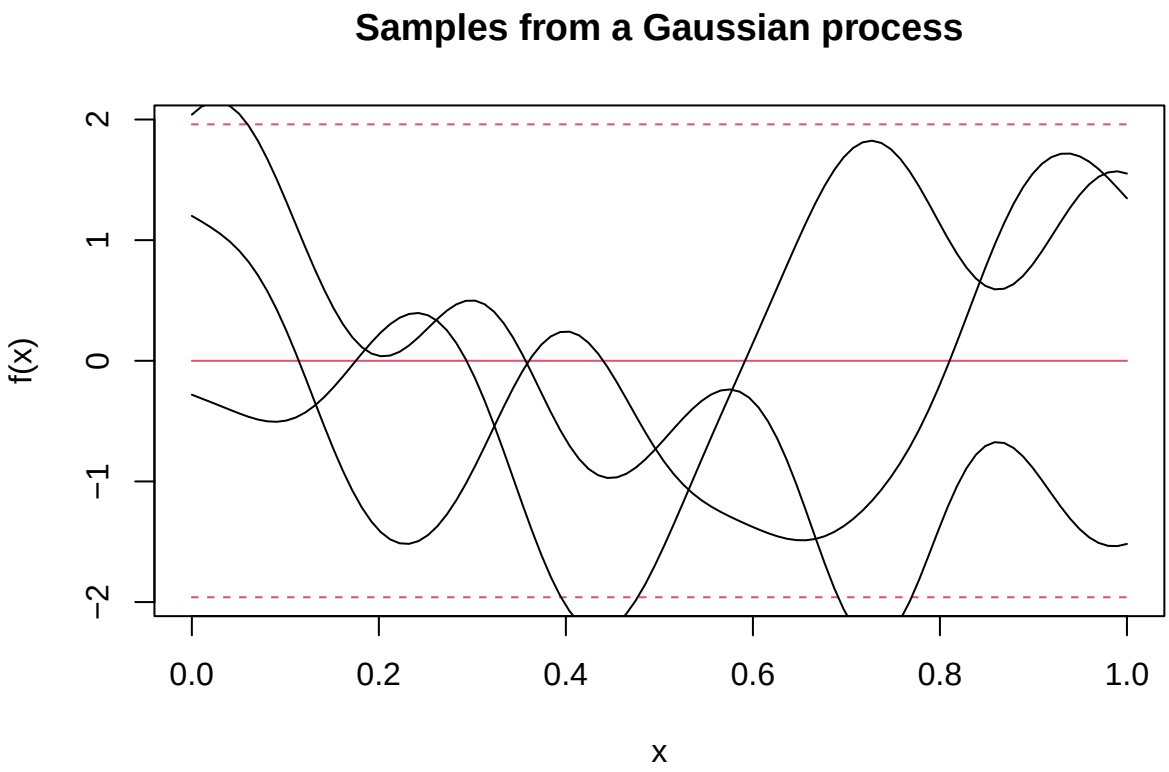
$$\sigma_f^2 \exp\left(\frac{(x_i - x'_j)^2}{2l^2}\right).$$

A positive definite function is a function $f : \mathbb{R} \to \mathbb{C}$ such that for any $x_1, ..., x_n \in \mathbb{R}$, the matrix $\Sigma = (f(x_i - x_j))_{i,j=1}^n$ is positive semi-definite (there is an alternative definition in dynamical systems).

You can sample from a GP by taking the square-root of the covariance matrix and multiplying a standard normal vector by it. (My code to do this when fitting the GP by MCMC had the transpose of the square-root matrix because R does it differently than I expected, which must be one reason it never worked. Also it helps to add a small value to the main diagonal before taking the square-root, because otherwise it can be very numerically unstable, and can fail for larger numbers of input values, and larger length scales. I'm not sure why that is, but it's mentioned in the appendix of the GP book. I think that was a problem in my python code which meant that I couldn't predict a large number of values at once.)

```
# Functions to sample from and predict values of a Gaussian process.
source("GP funcs.R")

# Plot samples from a GP
plot_GP_samps(
  l = 0.1, # Length scale
  sigma_f = 1, # Function standard deviation
  n_samps = 3 # Number of samples
)
```



Connection to inner product and basis function expansion

Projecting the input values into a feature space with an infinite set of basis functions $\phi(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$, and taking the inner product in that space, defines a covariance function for a GP, and any positive definite covariance function can be expressed as such an inner product.

For a finite set of inputs/basis functions the covariance matrix would not have full rank for a larger set of points, so the inverse would not exist to define a Gaussian distribution for them.

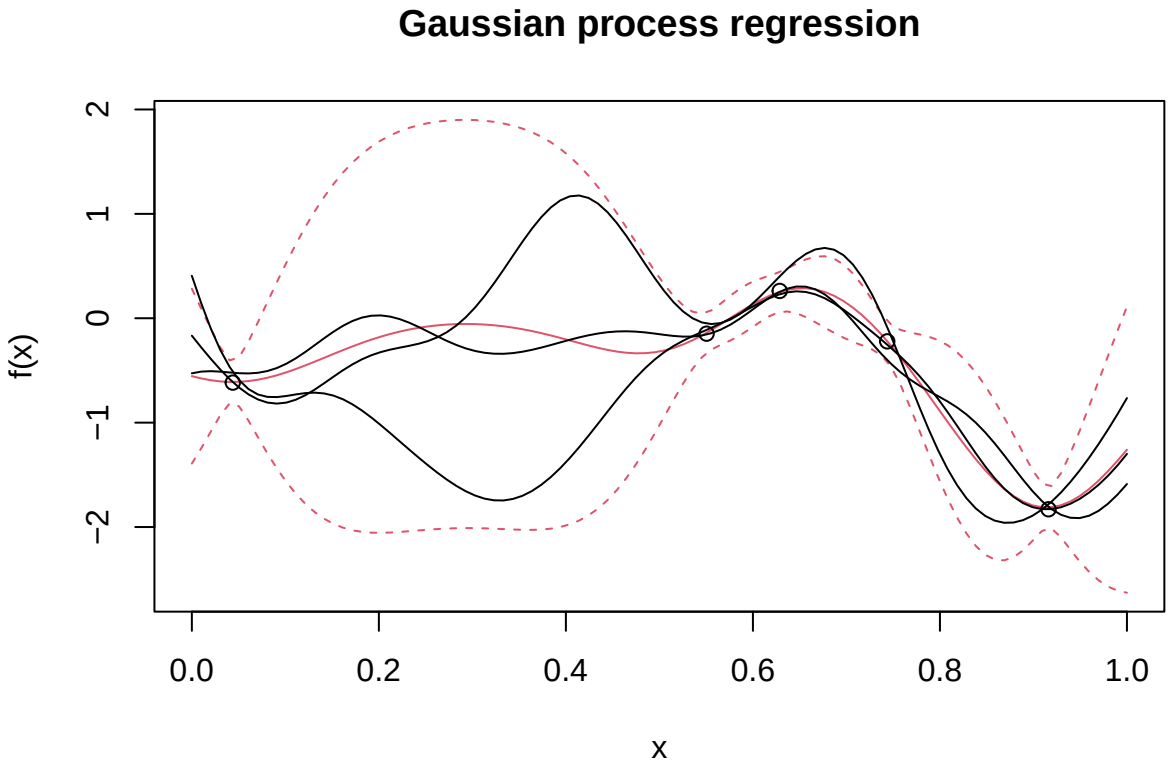
Gaussian process regression

A GP can be used as a functional prior. The posterior is then the conditional distribution of functions given observations at a set of input values. If the observations are assumed to include some Gaussian noise this is another GP, e.g.

$$
\begin{aligned}
\mathbf{y} &= f(\mathbf{x}) + \epsilon, \\
f(\mathbf{x}) &\sim N(\mu(\mathbf{x}) = \mathbf{0}, \Sigma = K(\mathbf{x}, \mathbf{x})), \\
\epsilon &\sim N(0, \sigma_n^2 I_d), \\
\implies f(\mathbf{x}')|\mathbf{y}(\mathbf{x}) &\sim N(\mu(\mathbf{x}'), \Sigma), \\
\mu(\mathbf{x}') &= K(\mathbf{x}', \mathbf{x})[K(\mathbf{x}, \mathbf{x}) + \sigma_n^2 I_d]^{-1}\mathbf{y}(\mathbf{x}), \\
\Sigma &= K(\mathbf{x}', \mathbf{x}') - K(\mathbf{x}', \mathbf{x})[K(\mathbf{x}, \mathbf{x}) + \sigma_n^2 I_d]^{-1}K(\mathbf{x}, \mathbf{x}').
\end{aligned}
$$

The posterior mean function can be seen to be a linear function, either of the observed values, or of the covariances of the input values of the predictions with those of each of the observations.

```
plot_GP_regression(
  n_obs = 5, # Number of points to observe
  l = 0.1, # Length scale
  sigma_f = 1, # Function standard deviation
  sigma_n = 0.1 # Noise standard deviation
)
```



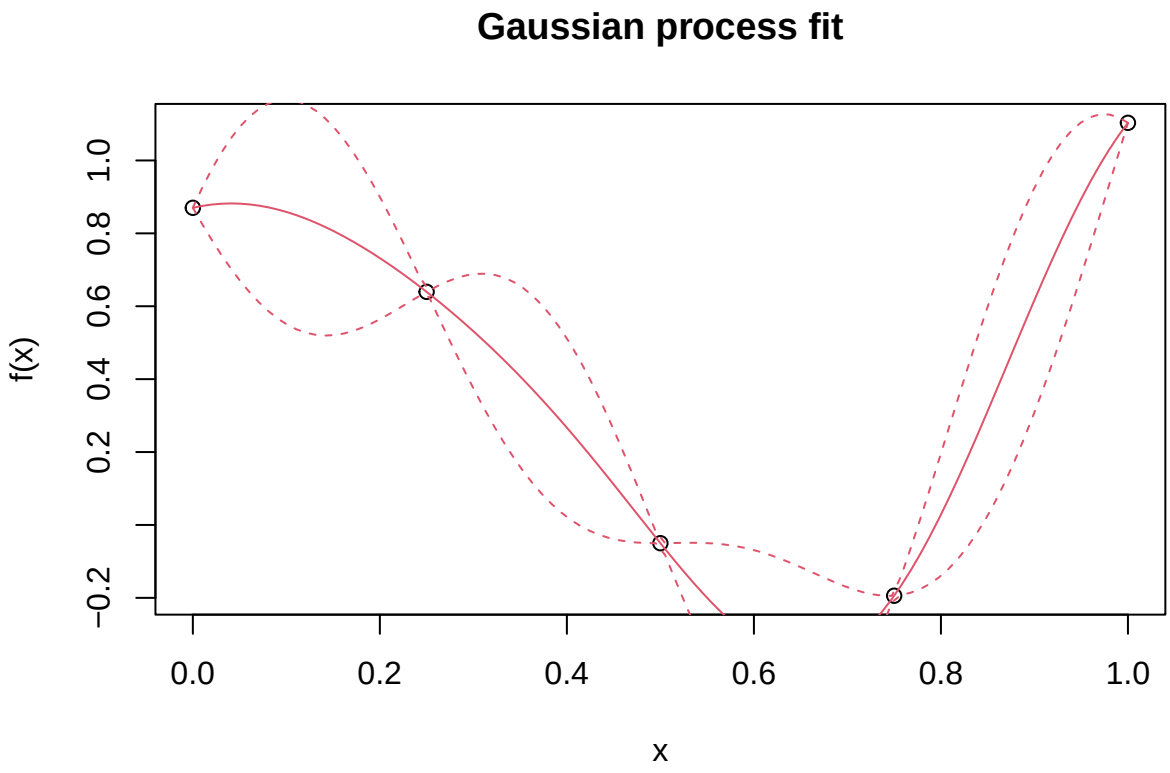
Connection to Bayesian linear regression

When a Gaussian prior is placed on the weights of a linear regression model with Gaussian noise, and projection of the inputs into a feature space with infinitely many basis functions, the posterior is a Gaussian process of the same form as in GP regression, with the covariance function given by the inner product of the inputs in the feature space multiplied by the square root of the prior covariance matrix.

Hyper-parameter tuning

There are various methods of model selection and hyper-parameter tuning. One is maximizing the posterior probability of the observations. The partial derivatives are known for some covariance functions making the numerical process tractable.

```
plot_GP_fit(
  n_obs = 5, # Number of points to observe
  l = 0.1, # Length scale
  sigma_f = 1, # Function standard deviation
  sigma_n = 0.1 # Noise standard deviation
)
```



Notes taken mainly from Rasmussen & Williams, Gaussian Processes for Machine Learning, 2006.