Gaussian process regression

Kadin Zhang

2/25

1 Gaussian processes

Typically, regression problems seek to model a function $f : \mathbb{R}^n \to \mathbb{R}$ by assuming some parametric form for f, and then reasoning about these parameters. In the Bayesian framework, we might assume $f(x) = x^\top w$, place a prior $w \sim N(\mu, \Sigma)$, then derive a distribution on our posterior w_D.

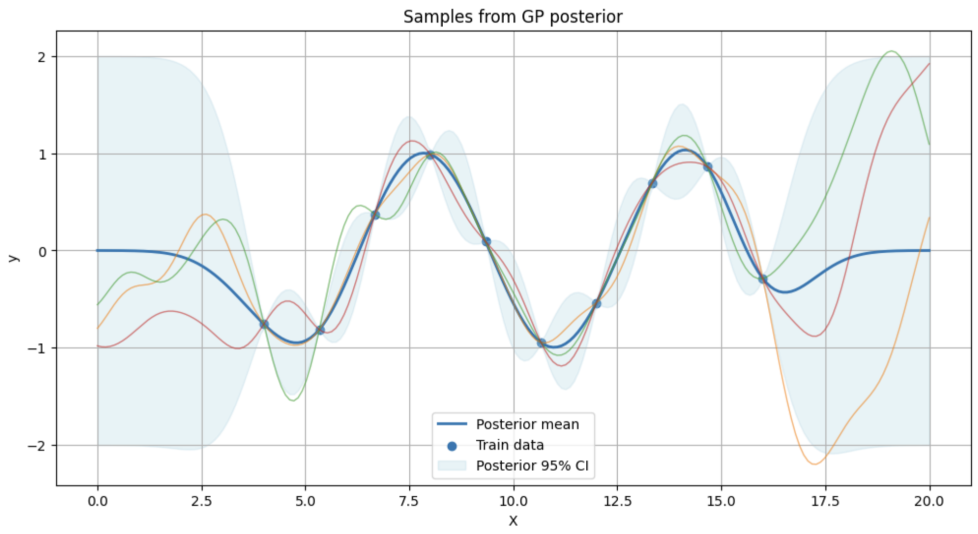
Rather than enforcing any strict functional form, Gaussian process regression places our uncertainty on the entire function itself, only assuming that our belief about our function is well modeled by a Gaussian process:

Definition (Gaussian process): A *Gaussian process* is a collection of random variables such that any finite collection has a joint Gaussian distribution.

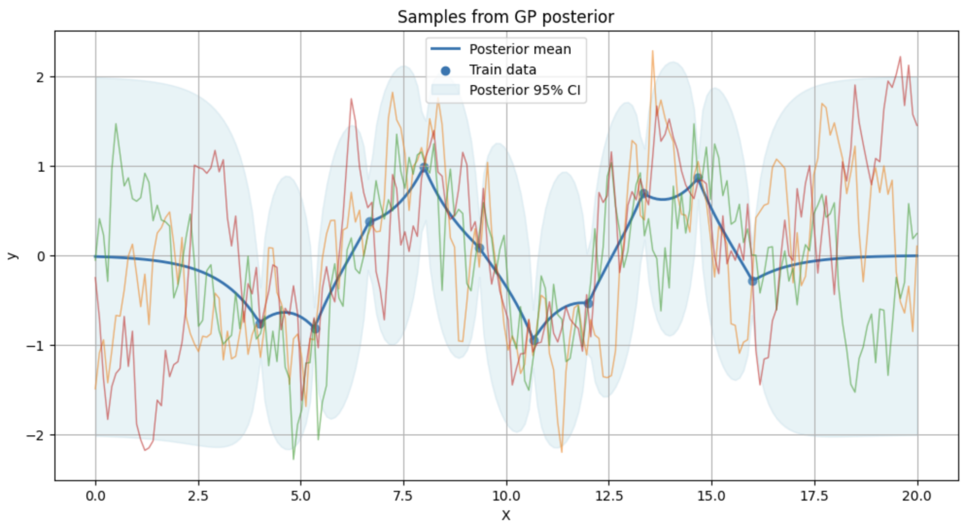
We can think of these random variables as the evaluation of a function f at all points in the domain. We can parameterize a Gaussian process with mean function $\mu : \mathcal{X} \to \mathbb{R}$ and positive semi-definite kernel (i.e. the Gram matrix is PSD for all subsets of $\mathcal{X}$) $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. We say $f \sim GP(\mu, K)$ to mean for all $X \subset \mathcal{X}$,

$$f(X) \sim N(\mu(X), K(X, X)),$$

As an example, say $\mathcal{X} = \mathbb{R}$, $\mu = 0$, and K is the squared exponential kernel. Then, samples f might look like



Because of our squared exponential kernel, we see that samples are smooth. Instead, if we used $k(x_1, x_2) = \exp(-\|x_1 - x_2\|_1)$, our samples would look like:



2 Posterior of Gaussian process

A key property of Gaussian processes is that the posterior is also a Gaussian process. The intuition is that we can condition our "infinite-dimensional" Gaussian on the outcome $f := f(X)$ of some observed X. In particular, by our Gaussian process property, for any test query X^*, the joint distribution of f and $f^* := f(X^*)$ is Gaussian:

$$N\left(\begin{bmatrix}\mu(X)\\ \mu(X^*)\end{bmatrix}, \begin{bmatrix}K(X,X) & K(X,X^*)\\ K(X^*,X) & K(X^*,X^*)\end{bmatrix}\right),$$

Now we can condition on f to get posterior $N(\mu_{f^*|D}, \Sigma_{f^*|D})$, where

$$\begin{aligned}\mu_{f^*|D} &= \mu(X^*) + K(X^*,X)K(X,X)^{-1}(f-\mu(X))\\ \Sigma_{f^*|D} &= K(X^*,X^*) - K(X^*,X)K(X,X)^{-1}K(X,X^*).\end{aligned}$$

In particular, note that these represent valid mean and covariance functions as a function of X^*. So the posterior distribution is a Gaussian process. Note that the posterior mean and kernel reflects the visual intuition of our posterior: test points close to data points should have mean close to the observed data values, and near 0 covariance matrix.

3 Noisy observations

Much like with linear regression, we can assume zero mean noise around our observed datapoints, i.e.

$$y = f + \varepsilon,$$

where $\varepsilon \sim N(0, \sigma^2)$. Then, the joint distribution of y, f^* becomes

$$N\left(\begin{bmatrix}\mu(X)\\ \mu(X^*)\end{bmatrix}, \begin{bmatrix}K(X,X)+\sigma^2 I & K(X,X^*)\\ K(X^*,X) & K(X^*,X^*)\end{bmatrix}\right),$$

As before we condition on y to get posterior on f^* of $N(\mu_{f^*|D}, \Sigma_{f^*|D})$, where

$$\begin{aligned}\mu_{f^*|D} &= \mu(X^*) + K(X^*,X)(K(X,X)+\sigma^2 I)^{-1}(f-\mu(X))\\ \Sigma_{f^*|D} &= K(X^*,X^*) - K(X^*,X)(K(X,X)+\sigma^2 I)^{-1}K(X,X^*).\end{aligned}$$

4 Marginal likelihood

The marginal likelihood of observed data is an important metric for Bayesian model selection. Assuming again noisy datapoints as well as mean and covariance of our prior parameterized by θ, we have

$$\begin{aligned} p(y \mid x, \theta) &= \int p(y \mid f) p(f \mid x, \theta) df \\ &= \int N(y; f, \sigma^2 I) N(f; \mu(x; \theta), K(x, x; \theta)) df \\ &= N(y; \mu(x; \theta), \sigma^2 I + K(x, x; \theta)), \end{aligned}$$

by closure of Gaussians under convolution. So the log likelihood is

$$\begin{aligned} &-\frac{1}{2}(y - \mu(x;\theta))^\top (K(x, x; \theta) + \sigma^2 I)^{-1} (y - \mu(x; \theta)) \\ &-\frac{1}{2} \log\det(K(x, x; \theta) + \sigma^2 I) - \frac{N}{2} \log(2\pi). \end{aligned}$$

Note that this incorporates a penalty on the prior covariance (i.e. the complexity of the model).

5 Implementation

```python
import numpy as np
import matplotlib.pyplot as plt
import math

def f(x):
  return math.sin(x)

def rbf_kernel(x1, x2, length_scale=1.0):
  x1 = x1.reshape(-1, 1)
  x2 = x2.reshape(1, -1)
  return np.exp(-np.square(x1 - x2) / 2 * (length_scale ** 2))

def abs_kernel(x1, x2):
  x1 = x1.reshape(-1, 1)
  x2 = x2.reshape(1, -1)
  return np.exp(-np.abs(x1 - x2))

def gp_posterior(X_train, y_train, X_test, kernel_fn):
  K = kernel_fn(X_train, X_train)  # K(X, X)
  K_s = kernel_fn(X_test, X_train) # K(X^*, X) = K(X, X^*).T
  K_ss = kernel_fn(X_test, X_test) # K(X^*, X^*)

  mu_s = 0 + K_s @ np.linalg.inv(K) @ (y_train - 0)
  cov_s = K_ss - K_s @ np.linalg.inv(K) @ K_s.T
  return mu_s, cov_s
```

```python
X_train = np.linspace(4, 16, 10)
y_train = np.sin(X_train)
X_test = np.linspace(0, 20, 200)
mu_s, cov_s = gp_posterior(X_train, y_train, X_test, rbf_kernel)
samples = np.random.multivariate_normal(mu_s, cov_s, 3)

plt.figure(figsize=(12, 6))
plt.title('Samples from GP posterior')
plt.xlabel('X')
plt.ylabel('y')
plt.plot(X_test, mu_s, label='Posterior mean', lw=2)
plt.scatter(X_train, y_train, label='Train data')
plt.fill_between(X_test,
                 mu_s - 2 * np.sqrt(np.diag(cov_s)),
                 mu_s + 2 * np.sqrt(np.diag(cov_s)),
                 label='Posterior 95% CI',
                 color='lightblue', alpha=0.3)
for sample in samples:
  plt.plot(X_test, sample, alpha=0.6, lw=1)
plt.grid()
plt.legend()
plt.show()
```