Gaussian Processes

Kaiwen Cai

1. Mutivariate Gaussian Distribution

Let us look at the multivariate case of Gaussian Distribution (aka., Normal Distribution). Each random variable is distributed randomly and their joint distribution is also a Gaussian distribution. A multivariate Gaussian distribution is defined by its mean $\boldsymbol{\mu}$ and $\mathbf{\Sigma}$.

$$\boldsymbol{X} = \left[\begin{array}{c} X_1 \\ X_2 \\ \vdots \\ X_n \end{array} \right] \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$$

where

$$\mathbf{\Sigma} = \mathbf{Cov}\left(X_i, X_j\right) = \mathbf{E}\left[\left(X_i - \mu_i\right)\left(X_j - \mu_j\right)^T\right]$$

2. Gaussian Process Regression

Suppose we observe a training set $\mathcal{D} = \{(\boldsymbol{x}_i, f_i), i = 1, 2, .., N\}$, where $f_i = f(\boldsymbol{x}_i)$. For a test set $\mathcal{D}_* = \{(\boldsymbol{x}_{i,*}), i = 1, 2, ..., N_*\}$, we want to predict the corresponding output $f_{i,*}$. We regard the combination of the training set and the test set as a multivariate Gaussian distribution:

$$\left[\begin{array}{c} \mathbf{f} \\ \mathbf{f}_* \end{array} \right] \sim \mathcal{N}\left(\left[\begin{array}{c} \boldsymbol{\mu} \\ \boldsymbol{\mu}_* \end{array} \right], \left[\begin{array}{cc} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^T & \mathbf{K}_{**} \end{array} \right] \right)$$

where $\mathbf{K} = \kappa(\mathbf{X}, \mathbf{X}) \in \mathbb{R}^{\mathbf{N} \times \mathbf{N}}$, $\mathbf{K}_* = \kappa(\mathbf{X}, \mathbf{X}_*) \in \mathbb{R}^{\mathbf{N} \times \mathbf{N}_*}$, $\mathbf{K}_{**} = \kappa(\mathbf{X}_*, \mathbf{X}_*) \in \mathbb{R}^{\mathbf{N}_* \times \mathbf{N}_*}$, κ is a prefined kernel function(here we adopt a RBF kernel):

$$\kappa\left(x, x'\right) = \sigma_f^2 \exp\left(-\frac{1}{2\ell^2}\left(x - x'\right)^2\right)$$

With the observed training set, we update the test set

$$p(\mathbf{f}_* | \mathbf{X}*, \mathbf{X}, \mathbf{f}) \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{*}}, \mathbf{\Sigma}_*)$$

where

$$\begin{aligned} \boldsymbol{\mu}_* &= \boldsymbol{\mu}(\mathbf{X}_*) + \mathbf{K}_*^{\mathbf{T}} \mathbf{K}^{\mathbf{-1}} (\mathbf{f} - \boldsymbol{\mu}(\mathbf{X})) \\ \mathbf{\Sigma}_* &= \mathbf{K}_{**} - \mathbf{K}_*^{\mathbf{T}} \mathbf{K}^{\mathbf{-1}} \mathbf{K}_* \end{aligned}$$

Now we sample from the multivariate distribution $p(\mathbf{f}_* | \mathbf{X}*, \mathbf{X}, \mathbf{f}) \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{*}}, \mathbf{\Sigma}_*)$. Recall when we have a univariate Gaussian distribution $x \sim \mathcal{N}(\mu, \sigma^2)$, we sample in a way $x \sim \mu + \sigma \cdot \mathcal{N}(0, 1)$. The equivalent way of sampling from a multivariate distribution is: $\mathbf{f}_* \sim \boldsymbol{\mu} + \mathbf{B} \cdot \mathcal{N}(\mathbf{0}, \mathbf{I})$, where $\mathbf{B}\mathbf{B}^{\mathbf{T}} = \mathbf{\Sigma}_*$.

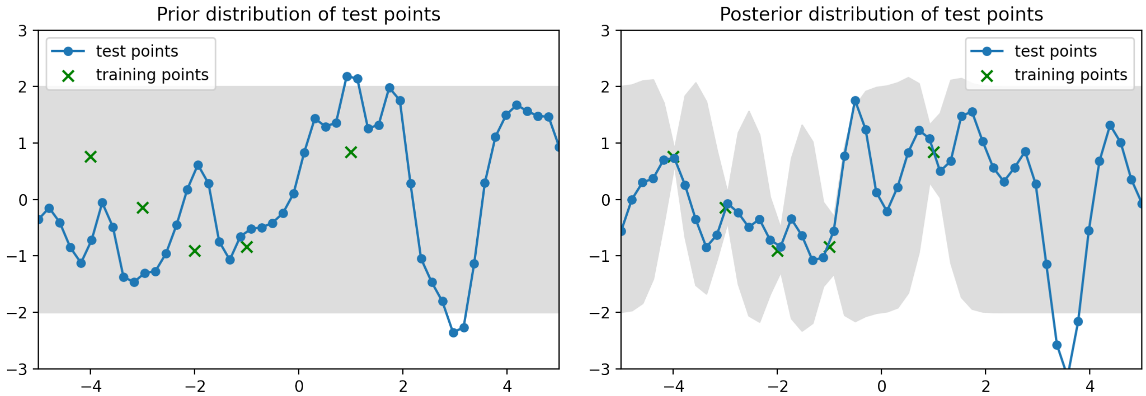

Figure 1. Prior and posterior distribution of Test set

```
#%%
import numpy as np
import matplotlib.pyplot as plt
np.set_printoptions(precision=3)
# plt.style.use('ggplot')


# KERNEL FUNCTION ==================== #
def kernel(a, b, param):
    """
    RBF kernel
    """
    sqdist = np.sum(a**2,1).reshape(-1,1) + np.sum(b**2,1) - 2*np.dot(a, b.T)
    return np.exp(-.5 * (1/param) * sqdist)
param = 0.1


# TEST DATA ========================== #
# PRIOR ============================== #
n = 50
Xtest = np.linspace(-5, 5, n).reshape(-1, 1)
K_ss = kernel(Xtest, Xtest, param)
stdv = np.diag(K_ss)
L = np.linalg.cholesky(K_ss)
f_prior = np.dot(L, np.random.normal(size=(n, 1)))


# TRAINING DATA ====================== #
Xtrain = np.array([-4, -3, -2, -1, 1]).reshape(5, 1)
ytrain = np.sin(Xtrain)
plt.plot(Xtest, f_prior)
plt.gca().fill_between(Xtest.flat,
                       0 - 2 * stdv,
                       0 + 2 * stdv,
                       color="#dddddd")
plt.scatter(Xtrain, ytrain, s=50, marker='x', c='green')
plt.axis([-5, 5, -3, 3])
```

```
    plt.title('Funtions from the GP prior')
    plt.savefig('prior.png', bbox_inches='tight', dpi=200)
    plt.show()

    # POSTERIOR ========================== #
    K = kernel(Xtrain, Xtrain, param)
    L = np.linalg.cholesky(K)

    K_s = kernel(Xtrain, Xtest, param)
    Lk = np.linalg.solve(L, K_s)    # L^-1 * Ks
    mu = np.dot(Lk.T, np.linalg.solve(L, ytrain)).reshape((n, ))
    # = (L^-1 * Ks)^T * L^-1 * f
    # = Ks^T*L^-T*L^-1*f
    # = Ks^T*(L*L^T)^-1*f
    # = Ks^T*K^-1*f

    s2 = np.diag(K_ss) - np.sum(Lk**2, axis=0)
    # = Kss - (L^-1 * Ks).**2 NOTE How?
    stdv = np.sqrt(s2)

    L = np.linalg.cholesky(K_ss - np.dot(Lk.T, Lk))
    # = Kss - (L^-1 * Ks)^T * (L^-1 * Ks)
    # = Kss - Ks^T * L^-T * L^-1 * Ks
    # = Kss - Ks^T * (L * L^T)^-1 * Ks

    f_post = mu.reshape(-1, 1) + np.dot(L, np.random.normal(size=(n, 1)))

    plt.plot(Xtest, f_post)
    plt.gca().fill_between(Xtest.flat,
                           mu - 2 * stdv,
                           mu + 2 * stdv,
                           color="#dddddd")
    plt.scatter(Xtrain, ytrain, s=50, marker='x', c='green')
    plt.axis([-5, 5, -3, 3])
    plt.title('Funtions from the GP posterior')
    plt.savefig('post.png', bbox_inches='tight', dpi=200)
    plt.show()
```