Gaussian Process Regression

Kyungjae Lee, Yunho Choi, Timothy Ha
RLLAB

[Exercise 1] Gaussian Process Regression

Let $y(x) = f(x) + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$.

Then $\mathbf{cov}(y(x_p), y(x_q)) = K(x_p, x_q) + \sigma_n^2 \delta_{pq}$ or in a matrix form

$$\mathbf{cov}(\mathbf{y}) = K(X, X) + \sigma_n^2 \mathbb{I}$$

The joint distribution between y and f_* is

$$\left[\begin{array}{c} \mathbf{y} \\ f_* \end{array} \right] \sim \mathcal{N}\left(0, \left(\begin{array}{cc} K(X,X) + \sigma_n^2 \mathbb{I} & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{array} \right)\right)$$

By conditioning, we get

$$\begin{aligned} f_* | X, \mathbf{y}, X_* &\sim \mathcal{N}(\bar{f}_*, \mathrm{cov}(f_*)), \\ \bar{f}_* &= K(X_*, X)\left(K(X,X) + \sigma_n^2 \mathbb{I}\right)^{-1} \mathbf{y} \\ \mathbf{cov}(f_*) &= K(X_*, X_*) - K(X_*, X)\left(K(X,X) + \sigma_n^2 \mathbb{I}\right)^{-1} K(X, X_*) \end{aligned}$$

[Exercise 1] Gaussian Process Regression

- Prediction
  - $f_* = K(x_*, X)(K(X,X) + \sigma_n I)^{-1} Y$
  - $cov(f_*) = K(x_*, x_*) - K(x_*, X)K(X,X)^{-1}K(X, x_*)$

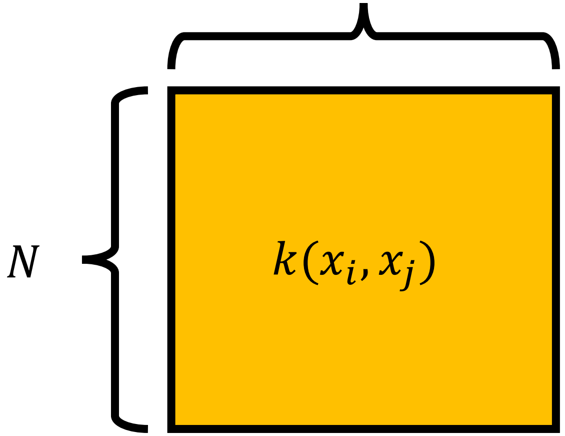
  - X : Input Data Y : Output Data

$$k(x_i, x_j) = \beta \exp(-0.5\lambda^{-1}(x_i - x_j)^2)$$

- Kernel Matrix

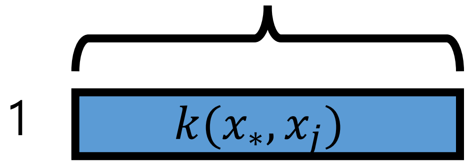
$K(X,X) = [k(x_i, x_j)]$

The number of data : N

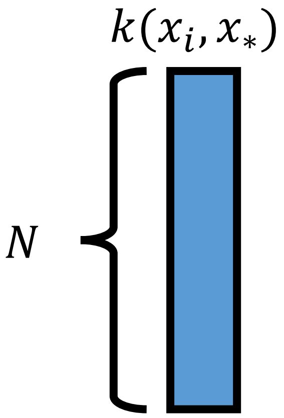


$K(x_*, X) = [k(x_*, x_j)]$

N



$K(X, x_*) = [k(x_i, x_*)]$

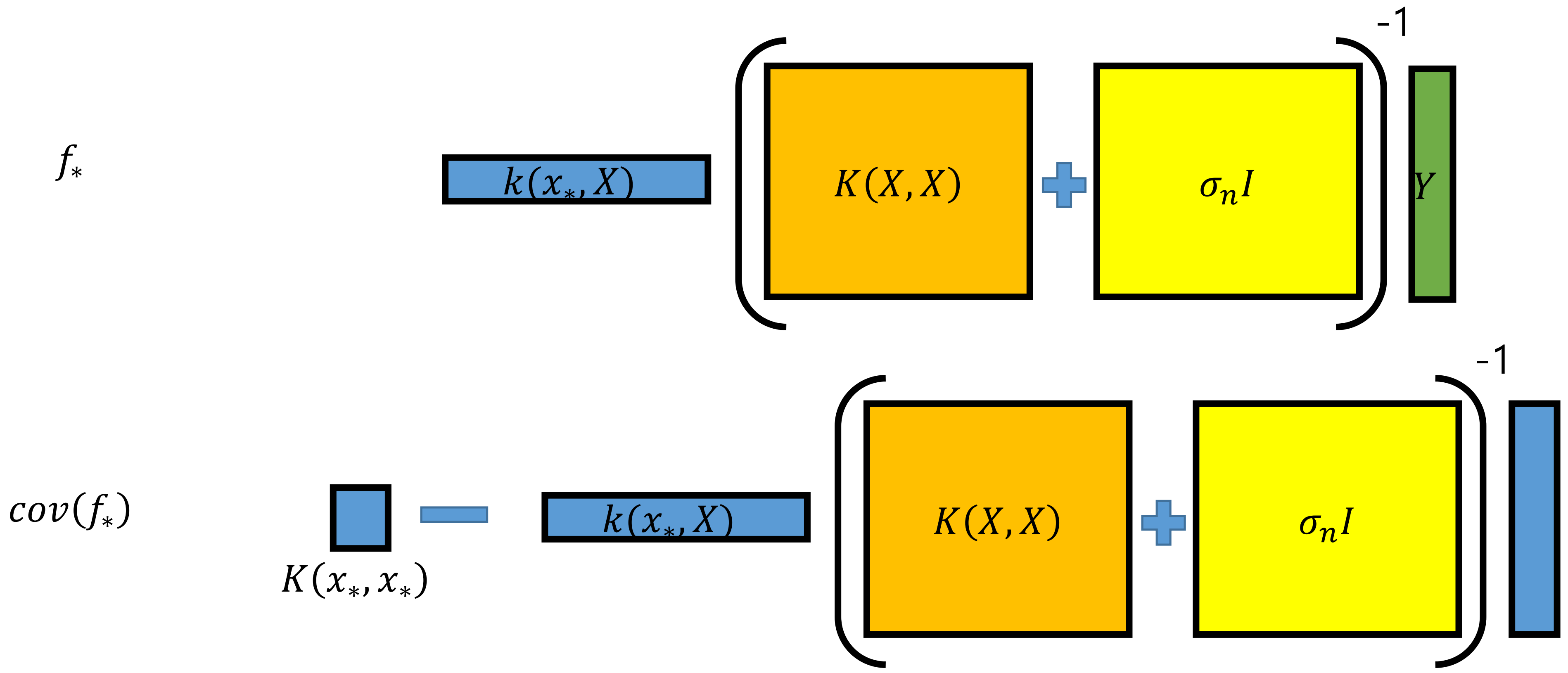


$K(x_*, X) = K(X, x_*)^t$

[Exercise 1] Gaussian Process Regression

- Prediction
  - $f_* = K(x_*, X)(K(X,X) + \sigma_n I)^{-1} Y$
  - $cov(f_*) = K(x_*, x_*) - K(x_*, X)(K(X,X) + \sigma_n I)^{-1} K(X, x_*)$

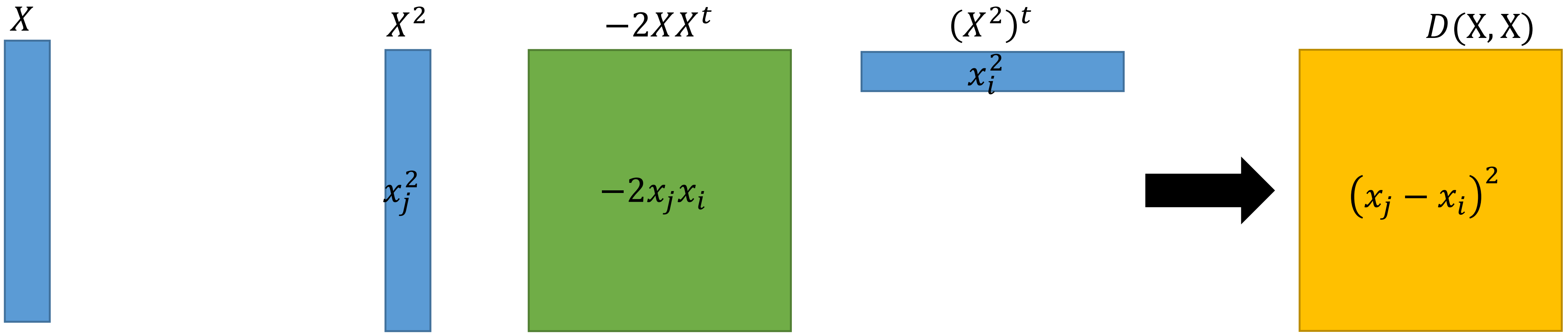
$$K(x_p, x_q) = \sigma_f^2 \exp\left(-\frac{1}{2\sigma_l^2}\|x_p - x_q\|^2\right)$$



[Exercise 1] Gaussian Process Regression

- Recall : How to compute Kernel matrix!?
  - First, compute distance matrix
  - Second, compute kernel matrix using distance matrix
    - $K(X,X) = \sigma_f^2 \exp\left(-\frac{D(X,X)}{2\sigma_l^2}\right)$

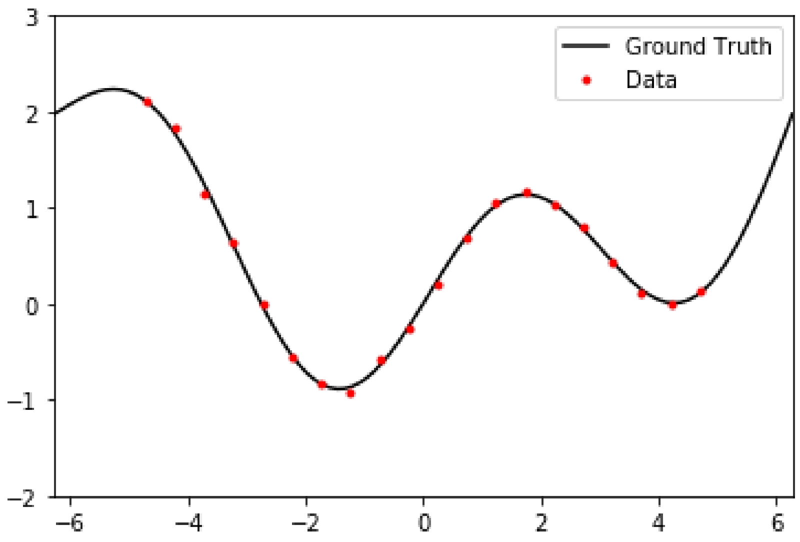
$$K(x_p, x_q) = \sigma_f^2 \exp\left(-\frac{1}{2\sigma_l^2}\|x_p - x_q\|^2\right)$$



[Exercise 1] Gaussian Process Regression

Data Generation

```python
def f3(x):
    return np.sin(x) + 0.05*x**2

x = np.linspace(-2*np.pi,2*np.pi,100)
y = f3(x)

n = 20
x_data = np.linspace(-1.5*np.pi,1.5*np.pi,n)
y_data = f3(x_data) + 0.05*np.random.randn(n)

plt.plot(x,y,"k-",label="Ground Truth")
plt.plot(x_data,y_data,"r.",label="Data")
plt.legend()
plt.xlim([-2*np.pi,2*np.pi])
plt.ylim([-2,3])
plt.show()
```



- 1-D Non-linear regression problem
- Goal
    - Construct kernel matrix in tensorflow graph
    - Run Gaussian process regression

[Exercise 1] Gaussian Process Regression

- Step 1
  - Define placeholder
  - x_star_ph : test point (unseen point)

- Step 2
  - Compute $K(X,X)$ and $K(x_*,X)$
  - Hint : see last exercise

Define kernel parameter and placeholder

```
tf.reset_default_graph()

inv_lambda = 4e0
beta = 1e0
sigma = 1e-4

x_ph = tf.placeholder(tf.float32,shape=(None,1))
y_ph = tf.placeholder(tf.float32,shape=(None,1))
x_star_ph = tf.placeholder(tf.float32,shape=(None,1))

inv_lambda_ph = tf.placeholder(tf.float32,shape=())
beta_ph = tf.placeholder(tf.float32,shape=())
sigma_ph = tf.placeholder(tf.float32,shape=())
```

Define GPR

$X^2 - 2XX' + X^2$

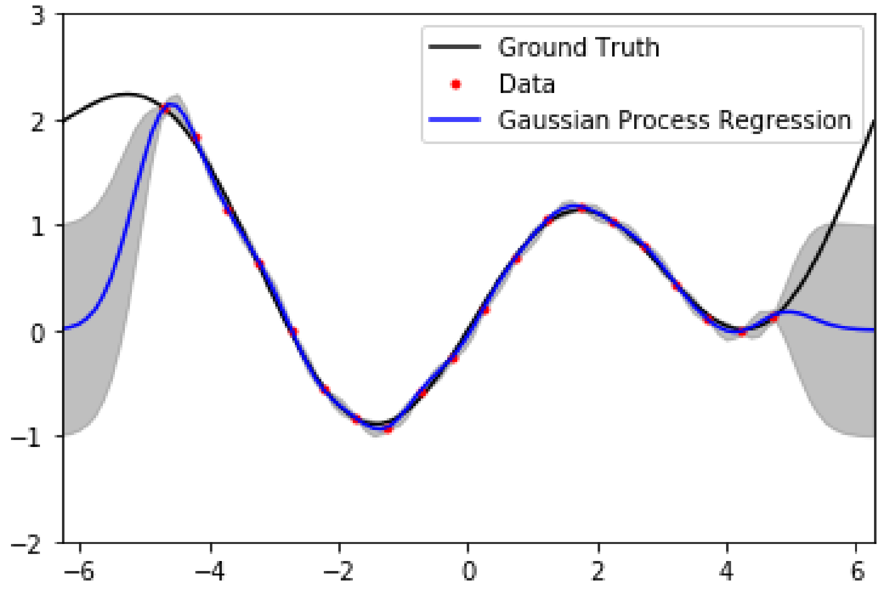
```
x_norm =
x_norm =

squared_dist_XX =
K_XX =

x_star_norm =
x_star_norm =

squared_dist_XstarX =
K_XstarX =

mean_y =

y_pred = tf.matmul(K_XstarX,tf.matmul(tf.linalg.inv(K_XX + sigma_ph*tf.identity(K_XX)),y_ph))
std_pred = tf.sqrt(beta-tf.diag_part(tf.matmul(K_XstarX,tf.matmul(tf.linalg.inv(K_XX + sigma_ph*tf.identity(K_XX)),tf.transpose(K_XstarX)))))
```

[Exercise 1] Gaussian Process Regression

- No need to optimize, run GPR!!

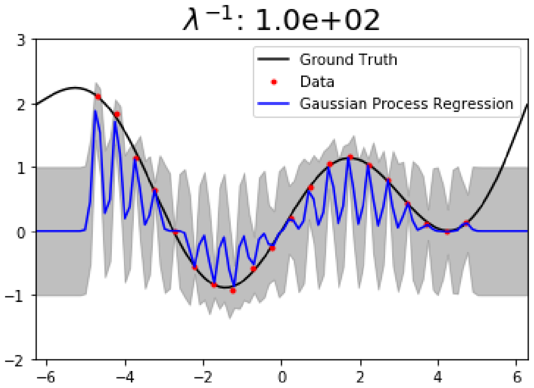
GPR

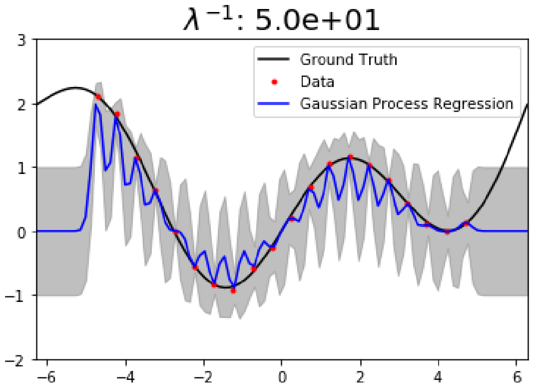
```
sess = tf.Session()

feed_dict = {x_star_ph:x[:,np.newaxis],x_ph:x_data[:,np.newaxis],y_ph:y_data[:,np.newaxis],
             inv_lambda_ph:inv_lambda,beta_ph:beta,sigma_ph:sigma}
y_pred_np,std_pred_np = sess.run([y_pred,std_pred],feed_dict=feed_dict)

plt.plot(x,y,"k-",label="Ground Truth")
plt.plot(x_data,y_data,"r.",label="Data")
plt.plot(x,y_pred_np,"b-",label="Gaussian Process Regression")
plt.fill_between(x, y_pred_np.flatten()-std_pred_np, y_pred_np.flatten()+std_pred_np, color='grey', alpha='0.5')
plt.legend()
plt.xlim([-2*np.pi,2*np.pi])
plt.ylim([-2,3])
plt.show()
```



[Exercise 2] Effect of Length Parameter

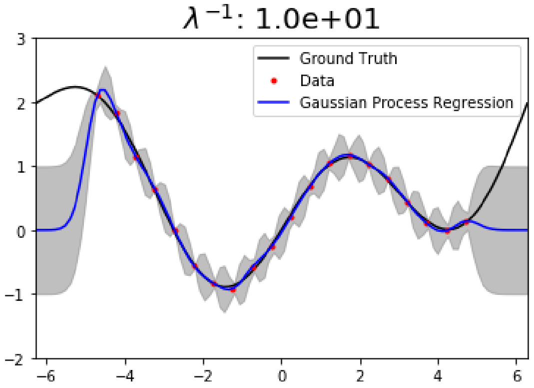
- Set various λ^{-1}
  - What is the best parameter? How to optimize it?

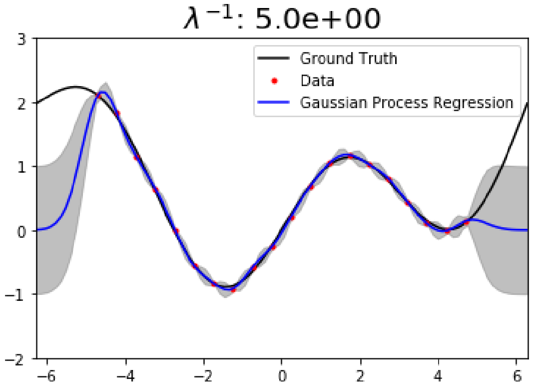


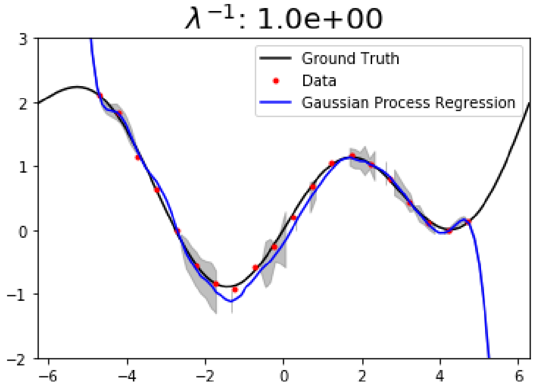


Effect of Scale Parameter λ^{-1}

```
inv_lambda1 =
inv_lambda2 =
inv_lambda3 =
inv_lambda4 = 5e0
inv_lambda5 =
```







[Exercise 2] Effect of Length Parameter

- Maximum marginal likelihood

Since $\mathbf{y} \sim \mathcal{N}(0, K + \sigma_n^2 \mathbb{I})$, the log marginal likelihood is

$$\log P(\mathbf{y}|X) = -\frac{1}{2}\mathbf{y}^T (K + \sigma_n^2 \mathbb{I})^{-1} \mathbf{y} - \frac{1}{2} \log |K + \sigma_n^2 \mathbb{I}| - \frac{n}{2} \log 2\pi,$$

which can be used to estimate σ_n^2 and parameters for the kernel function (using a gradient based method).

For example, if the following squared exponential kernel is used, the kernel parameters are (σ_f^2, σ_l^2).

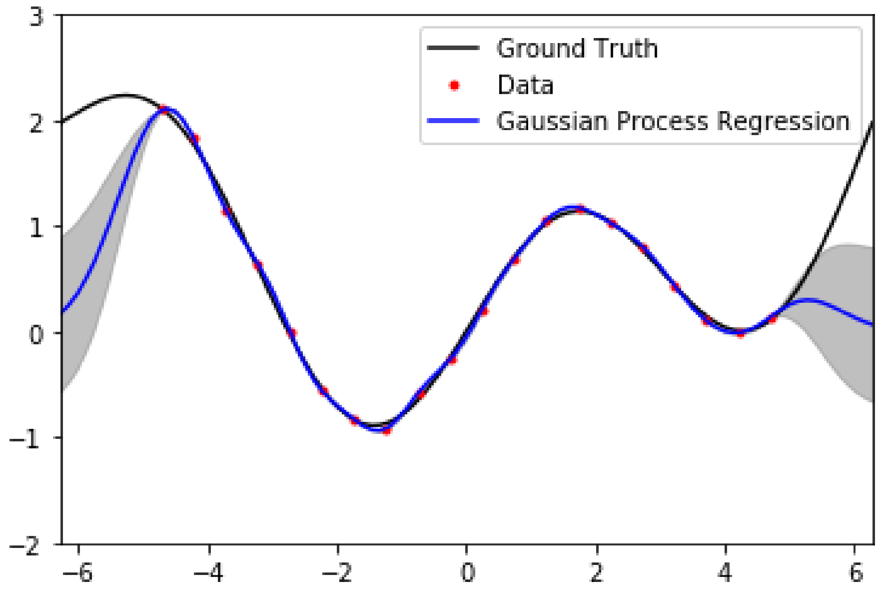
$$K(x_p, x_q) = \sigma_f^2 \exp\left(-\frac{1}{2\sigma_l^2} \|x_p - x_q\|^2\right)$$

In practice, selecting the right kernel for a given problem is also an important task.

[Exercise 2] Effect of Length Parameter

We can optimize the hyperparameter using sklearn package ¶

```
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF, ConstantKernel as C

kernel = C(1.0, (1e-3, 1e3)) * RBF(10, (1e-2, 1e2))
gp = GaussianProcessRegressor(kernel=kernel, n_restarts_optimizer=9)
gp.fit(x_data[:,np.newaxis], y_data[:,np.newaxis])

y_pred_sk, std_pred_sk = gp.predict(x[:,np.newaxis], return_std=True)

plt.plot(x,y,"k-",label="Ground Truth")
plt.plot(x_data,y_data,"r.",label="Data")
plt.plot(x,y_pred_sk,"b-",label="Gaussian Process Regression")
plt.fill_between(x, y_pred_sk.flatten()-std_pred_sk, y_pred_sk.flatten()+std_pred_sk, color='grey', alpha='0.5')
plt.legend()
plt.xlim([-2*np.pi,2*np.pi])
plt.ylim([-2,3])
plt.show()
```



[Exercise 2] Effect of Length Parameter

- To do
  - Define an arbitrary function
  - Generate noisy data
  - Defin GPR using sklearn packages
  - Fit hyperparameter
  - Plot the perdiction

Program your own example!

```
# Generate data

# Define gpr kernel

# Fit kernel hyperparameter

# Prediction and draw!
```