ML HW05 – Gaussian Processing

309553048_多工所_陳柏丞

- **Code Explanation：**

1. rational quadratic kernel (Xa, Xb, variance, alpha, lengthscale)

$$k(x_a, x_b) = \sigma^2 \exp\left(-\frac{\|x_a - x_b\|^2}{2\ell^2}\right)$$

```python
def RQkernel(xa, xb, var=1, alpha=1, lengthscale=1):
    return var * (1 + ((xa - xb) ** 2) / (2 * alpha * (lengthscale ** 2))) ** (-alpha)
```

2. negative marginal log-likelihood

$$\log p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^{\mathrm{T}}K^{-1}\mathbf{y} - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi.$$

```python
def GetLogLikelihood(args):
    var, alpha, lengthscale = args
    K = np.zeros((row, row))
    for i in range(row):
        for j in range(row):
            K[i][j] = RQkernel(X[i], X[j], var, alpha, lengthscale)

    return -(-0.5 * Y.T @ np.linalg.inv(K) @ Y -0.5 * np.log(np.abs(np.linalg.det(K))) - (n/2) * np.log(2 * np.pi))
```

3. read data & separate data to X, Y

```python
f = open("input.data")
input_data = f.read()
f.close()

row = 34
col = 2
n = 34
data = np.zeros((row, col))
count = 0
for xy in input_data.split("\n"):
    if xy != '':
        tmpX, tmpY = xy.split(" ")
        data[count] = [float(tmpX), float(tmpY)]
    count += 1

X = data[:, 0]
Y = data[:, 1]
```

4. calculate covariance matrix by rational quadratic kernel

```python
K = np.zeros((row, row))
var = 2.5
a = 500
l = 1

for i in range(row):
    for j in range(row):
        K[i][j] = RQkernel(X[i], X[j], var, a, l)
```

5. generate some sample & get new covariance matrix

$$\mathbf{C}_{N+1} = \begin{bmatrix} \mathbf{C} & k(\mathbf{x}, \mathbf{x}^*) \\ k(\mathbf{x}, \mathbf{x}^*)^\top & k(\mathbf{x}^*, \mathbf{x}^*) + \beta^{-1} \end{bmatrix}$$

get prediction of mean and variance (conditional gaussian)

$$\mu(\mathbf{x}^*) = k(\mathbf{x}, \mathbf{x}^*)^\top \mathbf{C}^{-1} \mathbf{y}$$

$$\sigma^2(\mathbf{x}^*) = k^* - k(\mathbf{x}, \mathbf{x}^*)^\top \mathbf{C}^{-1} k(\mathbf{x}, \mathbf{x}^*)$$

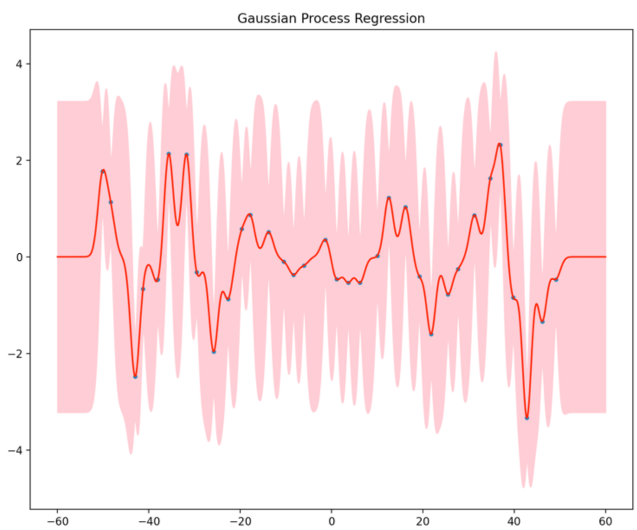
$$k^* = k(\mathbf{x}^*, \mathbf{x}^*) + \beta^{-1}$$

```
sample = np.linspace(-60, 60, 1000)
mean_y = []
var_y = []
Ks = np.zeros((row))
b = 5
for xs in sample:
    for i in range(row):
        Ks[i] = RQkernel(xs, X[i], var, a, l)

    Kss = RQkernel(xs, xs, var, a, l) + 1/b
    mean_s = Ks @ np.linalg.inv(K) @ Y
    var_s = Kss - Ks @ np.linalg.inv(K) @ Ks.T

    mean_y.append(mean_s)
    var_y.append(var_s)

mean_y = np.array(mean_y)
var_y = np.array(var_y)
```

6. plot gaussian processing

```
plt.title("Gaussian Process Regression")
plt.plot(X, Y, '.')
plt.plot(sample, mean_y, 'r')
plt.fill_between(sample, mean_y - 1.96*(var_y**0.5), mean_y + 1.96*(var_y**0.5), color='pink')
```

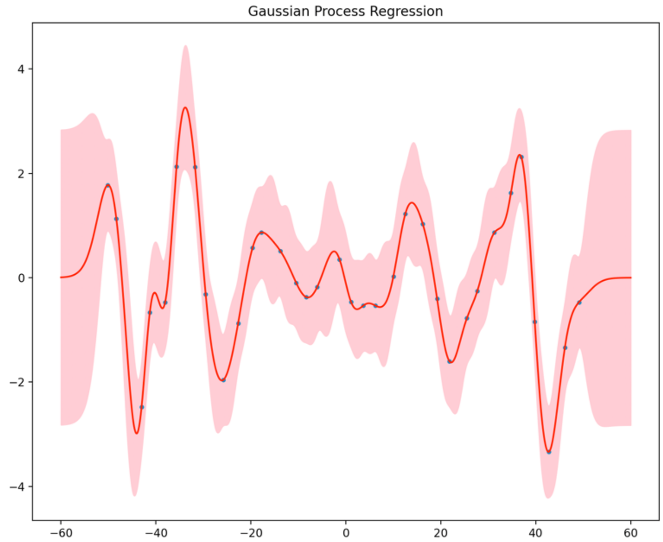
Result (variance = 2.5, alpha = 500, lengthscale = 1)



7. Minimize negative marginal likelihood

```
# Minimize negative marginal log-likelihood
begin = np.array([1, 1, 1])
res = minimize(fun=GetLogLikelihood, x0=begin, method='SLSQP')
print(f'Negative marginal log-likelihood : {res.fun}')
print(f'Variance : {res.x[0]}')
print(f'Alpha : {res.x[1]}')
print(f'Lengthscale : {res.x[2]}')
```

Optimize result (variance = 1.88, alpha = 8.51, lengthscale = 2.48)



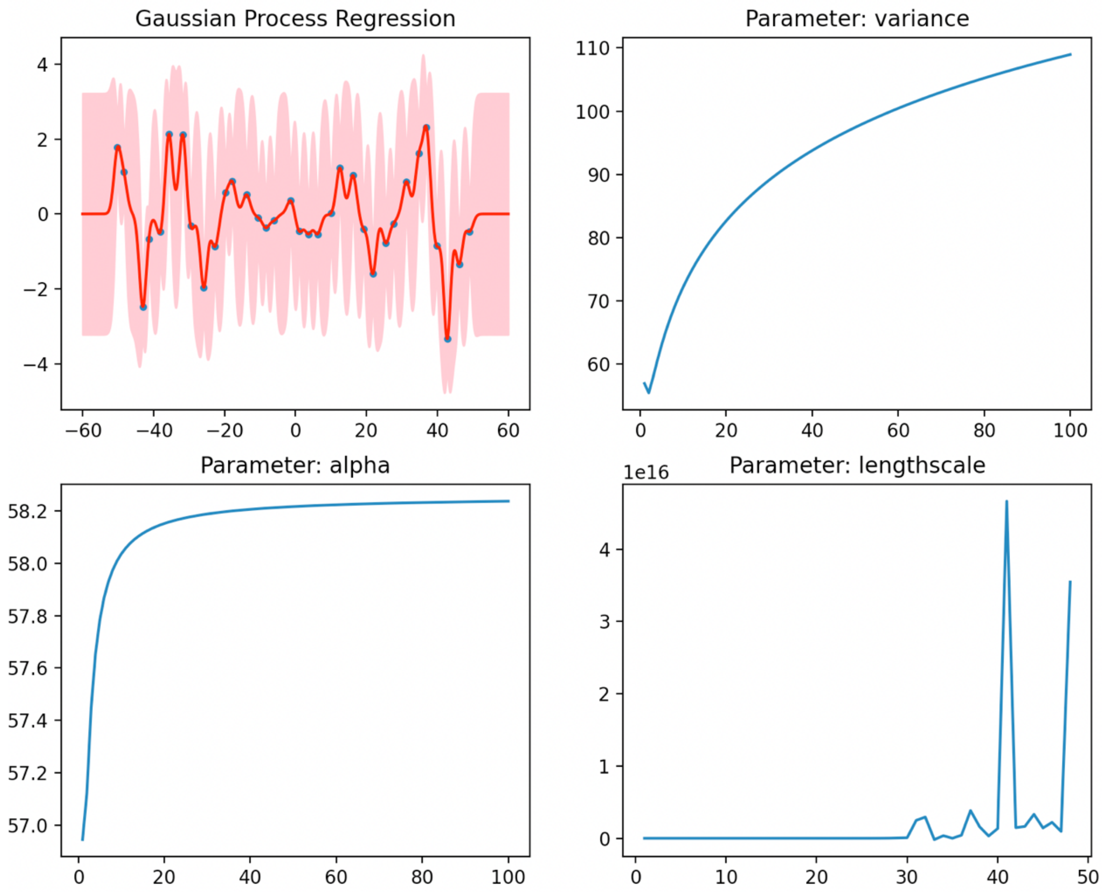
8. Testing other parameters; if a parameter set 1~100, others set 1. And get likelihood

```
x_m = np.arange(1, 101, 1)
L_var_m = []
L_alpha_m = []
L_length_m = []

for i in x_m:
    L = GetLogLikelihood([i, 1, 1])
    L_var_m.append(L)
    L = GetLogLikelihood([1, i, 1])
    L_alpha_m.append(L)
    L = GetLogLikelihood([1, 1, i])
    L_length_m.append(L)
```

Plot

```
plt.subplot(222)
plt.title("Parameter: variance")
plt.plot(x_m, L_var_m)

plt.subplot(223)
plt.title("Parameter: alpha")
plt.plot(x_m, L_alpha_m)

plt.subplot(224)
plt.title("Parameter: lengthscale")
plt.plot(x_m, L_length_m)
```

result



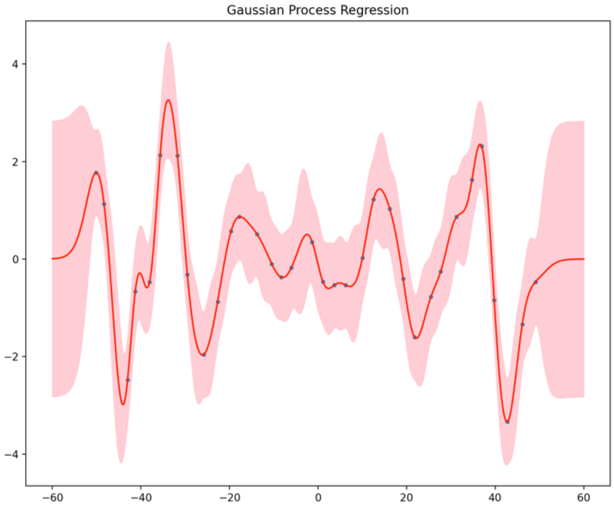
- **Experiment settings and results：**

1. At first, I'm not sure which parameters is best, so I set parameters in kernel by variance = 2.5, alpha = 500, lengthscale = 1.

   Result：

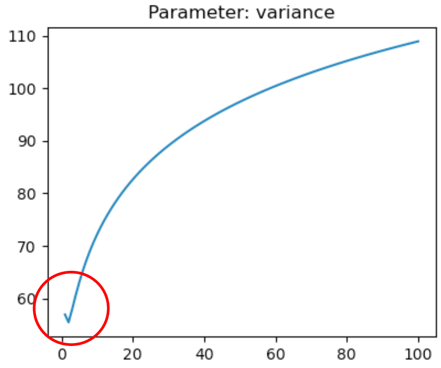
2. Then, I use the function, minimize(), and plotting with better parameter (variance = 1.88, alpha = 8.51, lengthscale = 2.48)

   Optimized result：



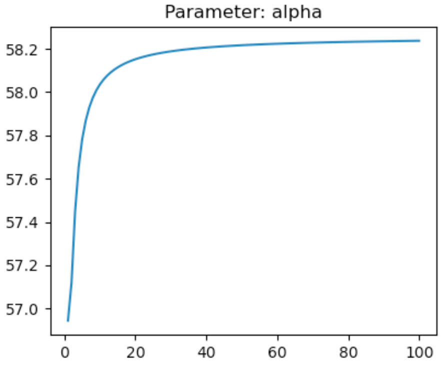
3. Furthermore, I want to know if I change one parameter in kernel and others are fixed, how likelihood could change.

   Variance line graph



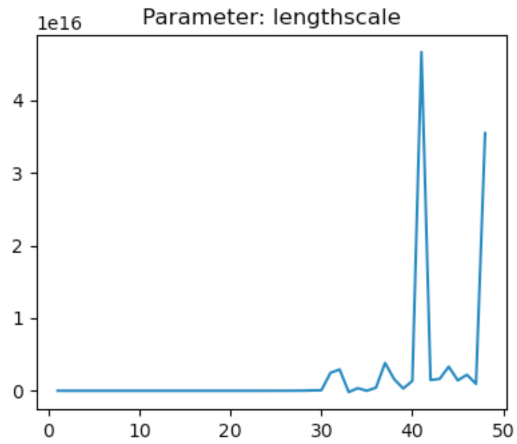
   Minimal likelihood is happened by variance around 2, and others are fixed to 1.

   Alpha line graph



   Obviously, likelihood in bigger alpha has ceiling when others are fixed to 1.

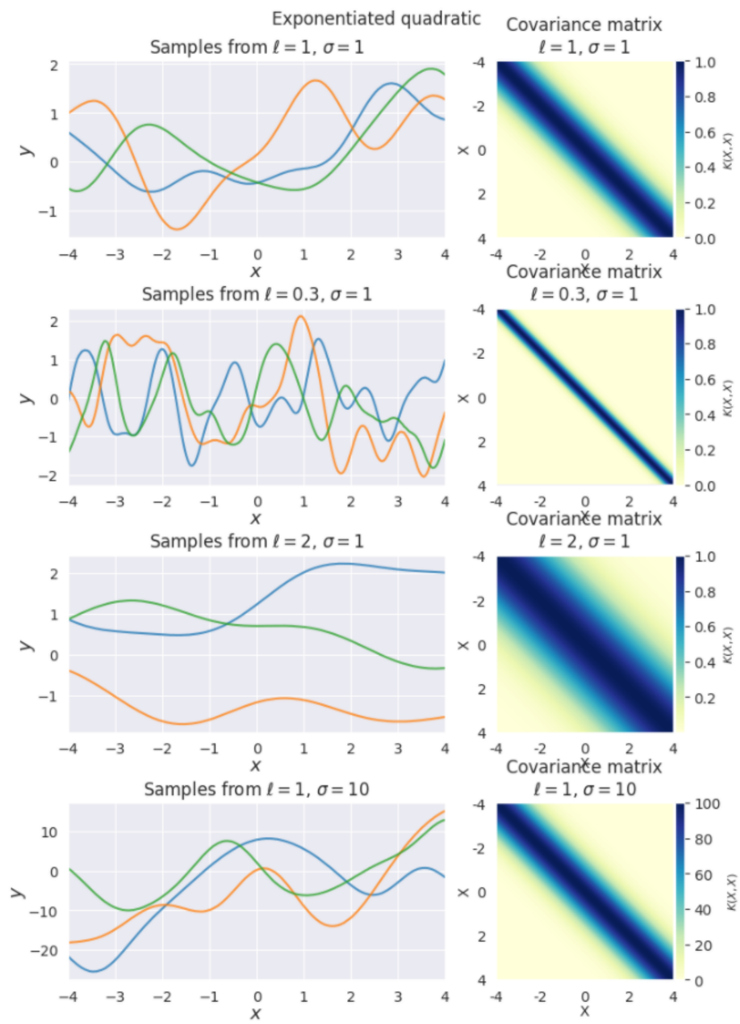
Lengthscale line graph



We couldn’t calculate likelihood when lengthscale over 50. It means the covariance matrix is singular, that is divide by zero.

- **Observations and discussion：**

We could find that which parameter in kernel is, determining the width of confidence interval.



Summary,

1. The larger variance is, the larger likelihood is. Then, the minimal likelihood happened by variance around 2, and others are fixed to 1.
2. The larger alpha is, the larger likelihood is. the likelihood has ceiling around 58.3.
3. With larger lengthscale, likelihood couldn’t be calculated because the covariance matrix is singular, that is, divided by zero.
4. If use minimize() to get minimal likelihood, our parameters are,
   Negative marginal log-likelihood：51.989679759470846
   Variance：1.8833709914285772
   Alpha：8.509966881390344
   Lengthscale：2.4762273397925236