ML Homework 5 – 1 Report

I. Gaussian Process

1. Code with detailed explanations

· Rational Quadratic kernel

The Rational Quadratic kernel is parameterized by a length scale parameter l>0 and a scale mixture parameter α>0. The kernel is given by:

$$k(x_i, x_j) = \left(1 + \frac{d(x_i, x_j)^2}{2\alpha l^2}\right)^{-\alpha}$$

where α is the scale mixture parameter, l is the length scale of the kernel and d(·,·) is the Euclidean distance.

According to the formula above, we define the kernel:

```
def rational_quadratic_kernel(xi, xj, l, alpha):
    return (1+cdist(xi, xj, 'sqeuclidean')/(2*alpha*l**2))**(-alpha)
```

Which sqeuclidean is the Squared Euclidean distance.

· Covariance Matrix

We have the marginal likelihood $p(y) = N(0, C)$, where the covariance matrix C has elements:

$$C(x_n, x_m) = k(x_n, x_m) + \beta^{-1}\delta_{nm}$$

According to the formula above, we calculate C:

```
C = rational_quadratic_kernel(X, X, l, alpha) + 1/beta*np.identity(X.shape[0])
```

· Conditional Distribution

We want to predict the distribution of new f^*, denote $y^* = f(x^*)$, the conditional distribution $p(y^*|y)$ is a Gaussian distribution with:

$$\mu(x^*) = k(x, x^*)^T C^{-1} y$$
$$\sigma^2(x^*) = k^* - k(x, x^*)^T C^{-1} k(x, x^*)$$

Where

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

So the code is:

```
k_x_xstar = rational_quadratic_kernel(X, Xstar, l, alpha)
kstar = rational_quadratic_kernel(Xstar, Xstar, l, alpha)+1/beta

mean = k_x_xstar.T.dot(np.linalg.inv(C).dot(Y))
var = kstar - k_x_xstar.T.dot(np.linalg.inv(C).dot(k_x_xstar))
```

· Optimize the kernel parameters

Consider the covariance matrix with hyper-parameters θ, the marginal likelihood is function of θ $p(\mathrm{y}|\theta) = \mathcal{N}(\mathrm{y}|0, \mathrm{C}_\theta)$,and the marginal log-likelihood is:

$$\ln p(\mathrm{y}|\theta) = -\frac{1}{2}\ln |\mathrm{C}_\theta| - \frac{1}{2}\mathrm{y}^\top \mathrm{C}_\theta^{-1}\mathrm{y} - \frac{N}{2}\ln (2\pi)$$

So we define the negative marginal log-likelihood:

```
def negative_marginal_likelihood(theta):
    C = rational_quadratic_kernel(X, X, theta[0], theta[1]) + 1/beta*np.identity(X.shape[0])
    res = 0.5*np.log(np.linalg.det(C))+0.5*Y.T.dot(np.linalg.inv(C).dot(Y))+X.shape[0]/2*np.log(2*math.pi)
```

And use scipy.optimize.minimize to calculate the hyper-parameters that minimize the negative marginal log-likelihood, then use the hyper-parameters to do gaussian process.

```
res = minimize(negative_marginal_likelihood, x0=[1,1])
Gaussian_Process(res.x[0], res.x[1])
```

- Visualization
  To mark the 95% confidence interval of *f*, the z-score is equal to 1.96, and

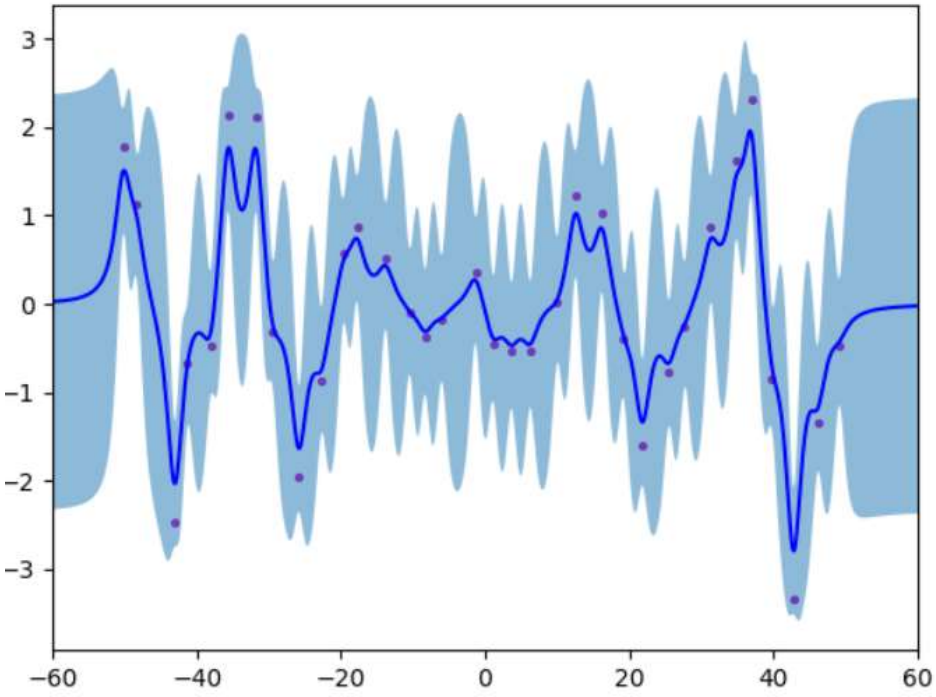
$$z = \frac{x - \mu}{\sigma}$$

,
so the upper bound and lower bound of *f* is:

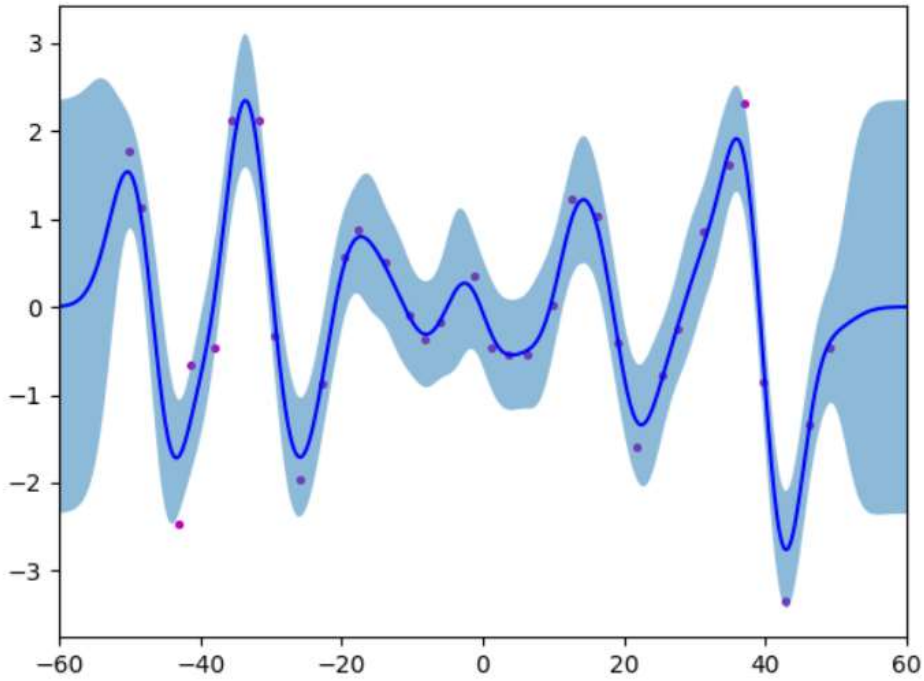
```
upper[i] = 1.96*var[i, i] + mean[i, 0]
lower[i] = 1.96*var[i, i] - mean[i, 0]
```

2. Experiments Settings and Results

- Part1:
  l = 1.0, α = 1.0



- Part2:
  l = 2.968833204734774, α = 893.8820888743509



3. Observations and Discussion

- Comparing the confidence interval with and without training data in the same figure, we can find that the confidence interval is smaller near the training data, and the confidence interval is larger when there is no training data(at both ends).

- Comparing the confidence intervals of unoptimized and optimized kernel parameters, we can find that the confidence interval after optimization is smaller, while the confidence interval of unoptimized words is larger.

- The execution time of optimized hyper-parameters is about 1.85 times that of none.

```
part1: 0.055847883224487305
part2: 0.10268473625183105
```