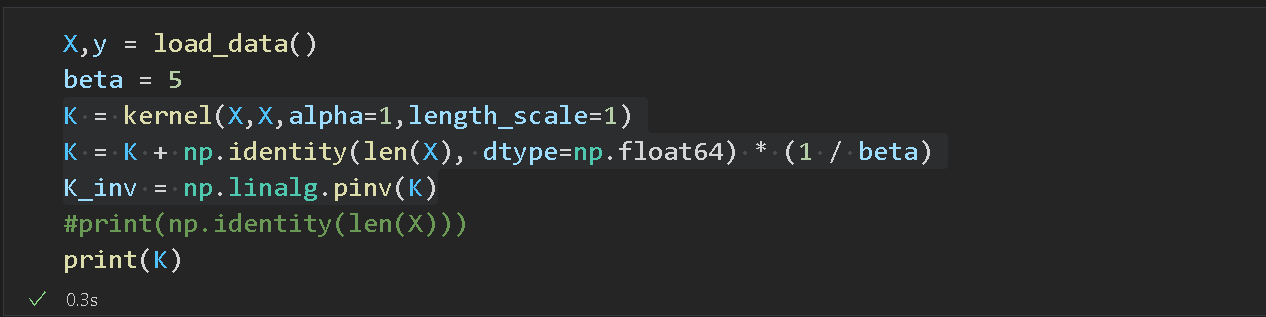
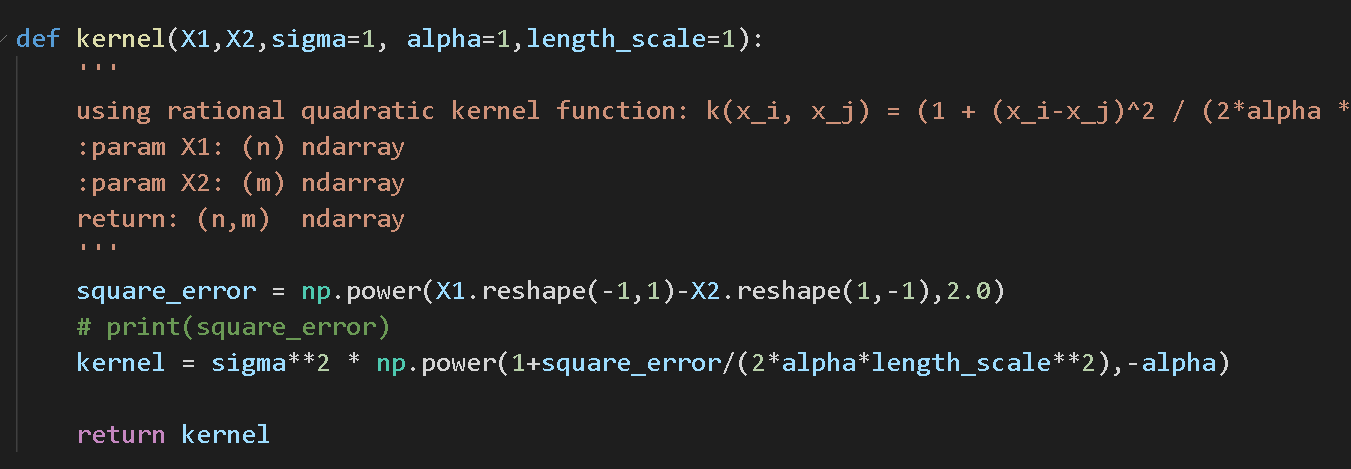
ML Lab5 Report

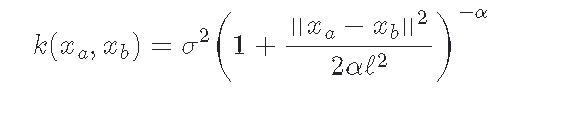
1. Gaussian Process
2. code with detailed explanations

Part1:





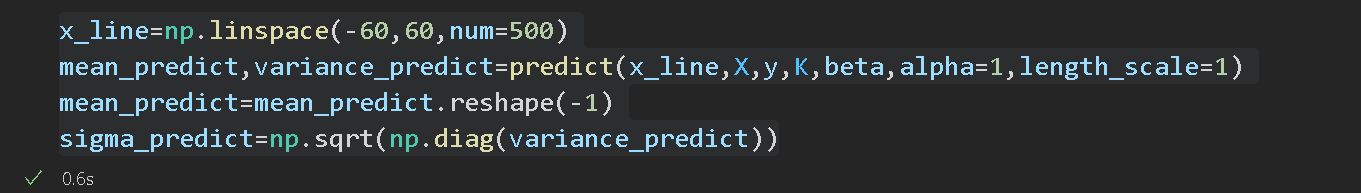
Rational quadratic kernel :

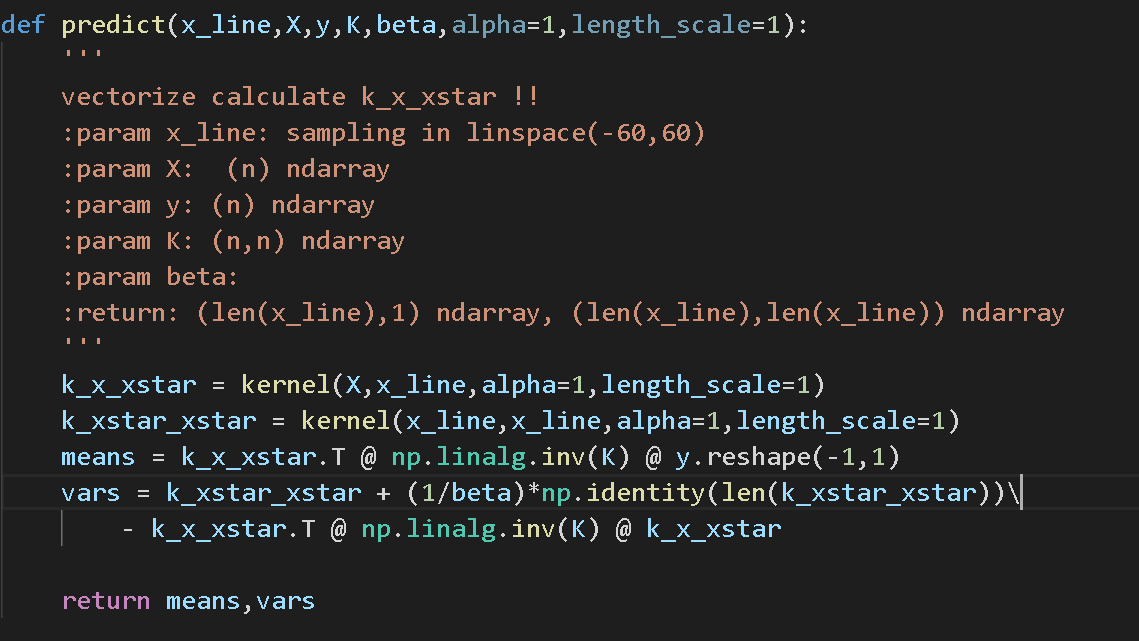


Covariance matrix:

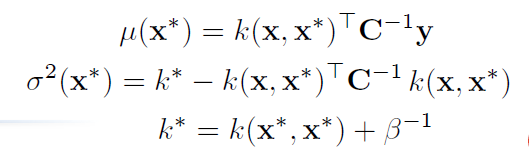


In this part, first we load data and build our rational quadratic kernel. We just initialize parameters all with 1. We use beta=5 in this implementation. Then we add some noise(1/beta) to the kernel to build covariance matrix based on the formula above.

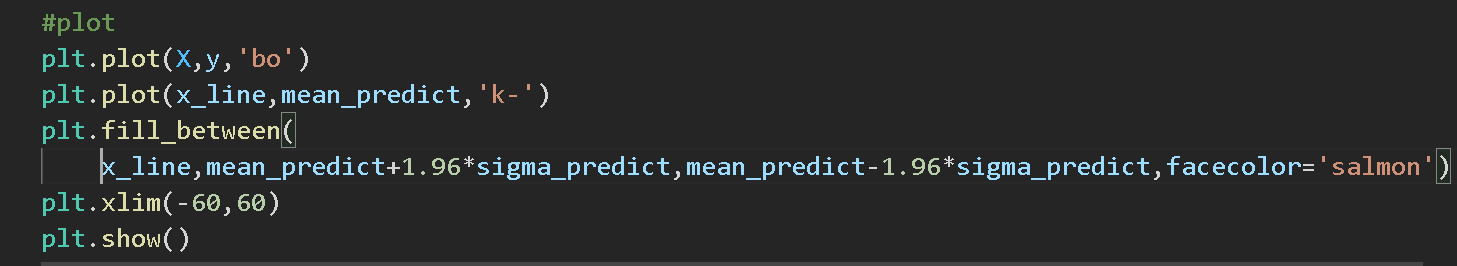




Mean and variance prediction formula:



We start to make predictions in this part. First we need to calculate the gram matrix of k(x\*, x) and k(x\*, x\*). After we have these values, then we start to predict the mean and variance of our testing data based on the formula above. Notice that we should extract the diagonal element of the N by N variance matrix to get the variance of each point in testing data.

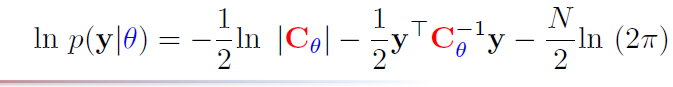


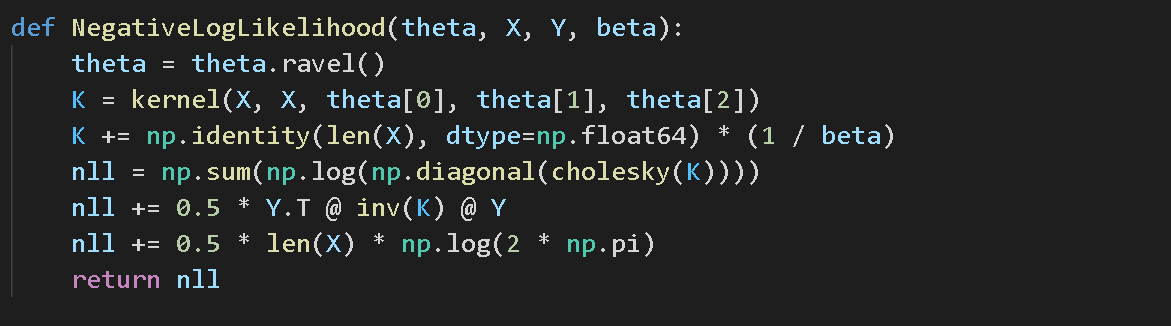
Finally, we visualize our result. 95% confidence interval is about 1.96\*sigma. The result will be showed in next section.

Part2:

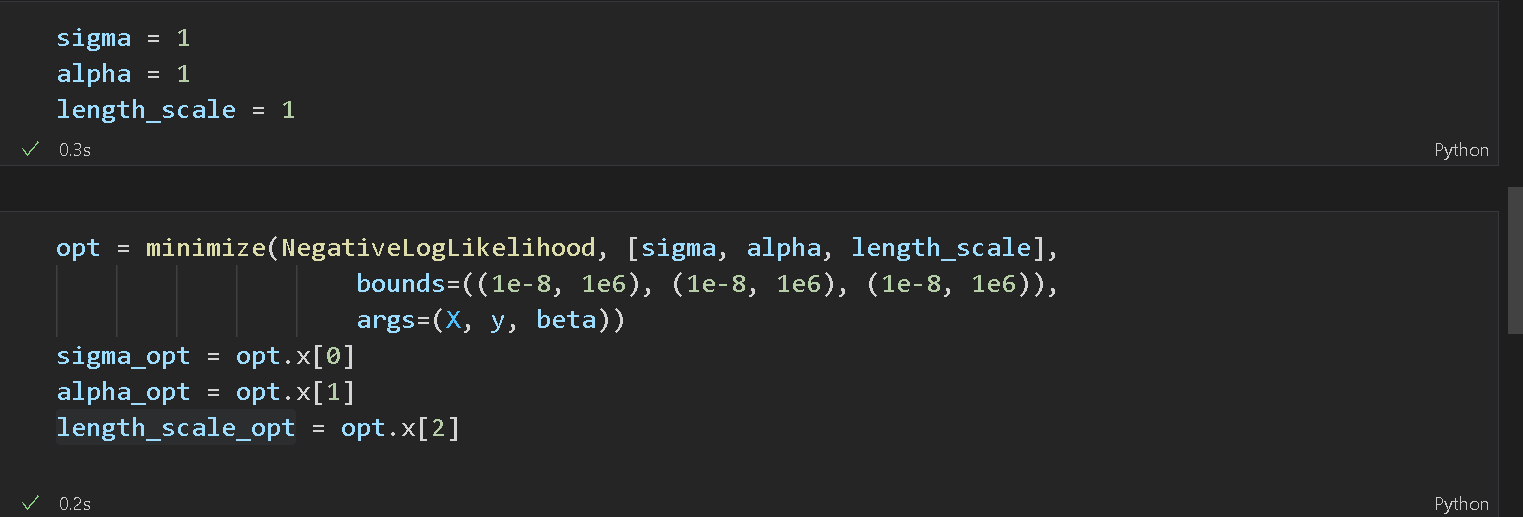
In this part, we need to use objective function to optimize our hyperparameter in kernel.

Marginal log likelihood:

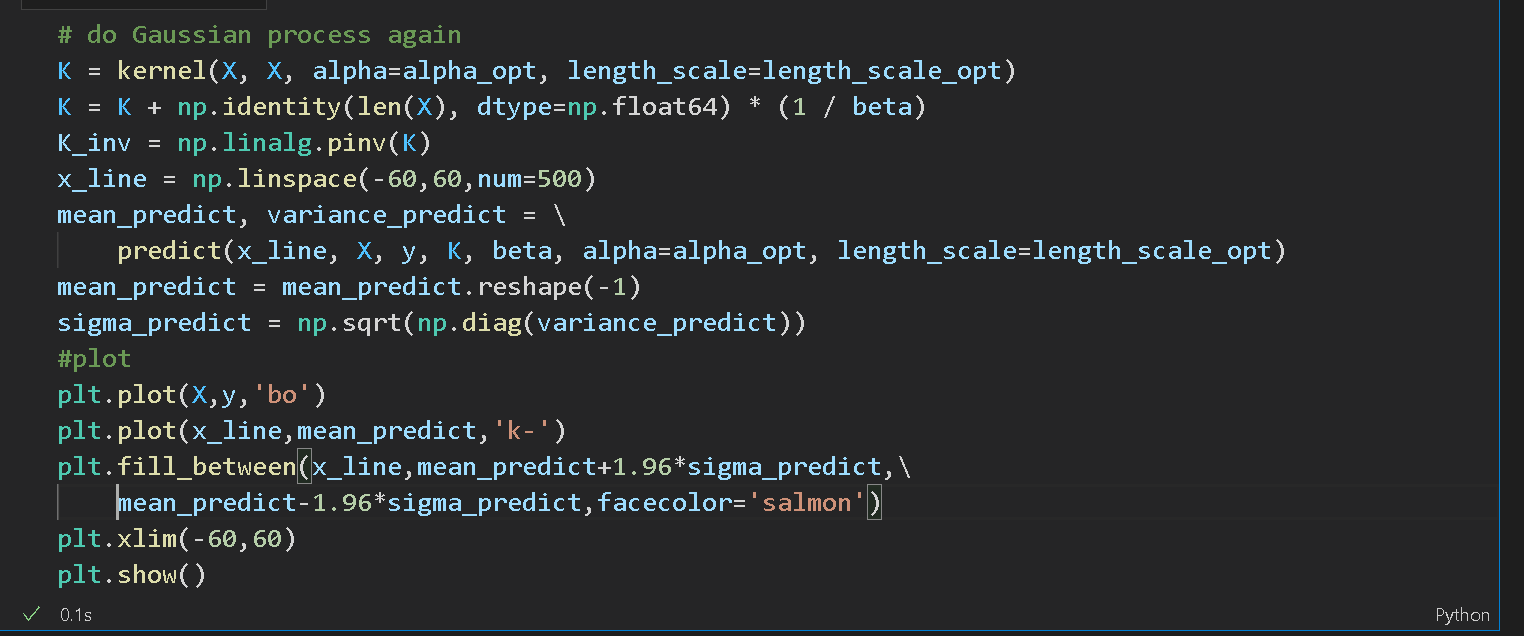




First we define our Negative Loglikelihood function. Notice that we get the determinant by calculate the product of all diagonal element of L(Cholesky(K)).



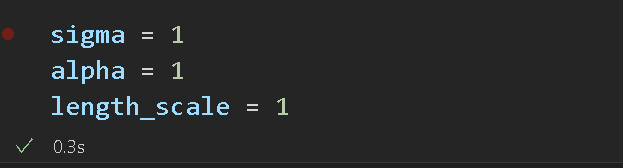
Then we use scipy.optimize.minimize to minimize our objective function. After minimize, we can get the optimize value of sigma, alpha and length\_scale.

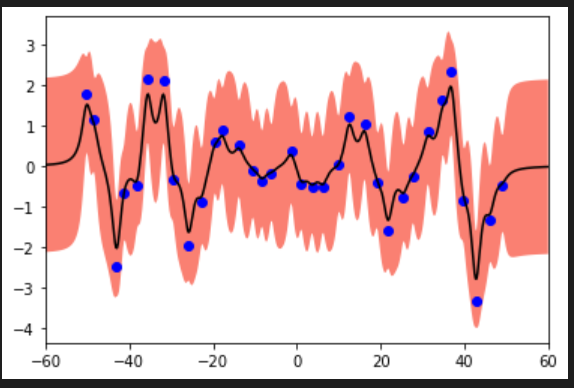


We can use the optimize values we got and do the same things like part 1. The result will be showed in next section.

1. experiments settings and results

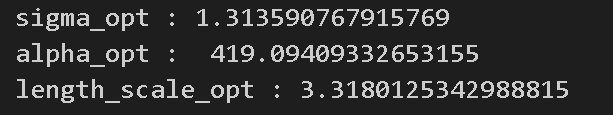
Part1



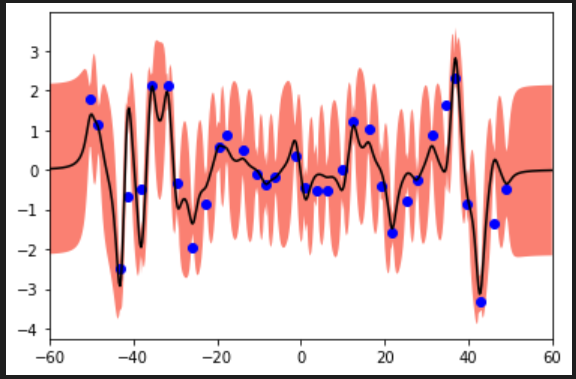


First we initialize the kernel parameters and get the result above.

Part2



In this part, we use our optimize parameters in kernel and visualize the prediction result.



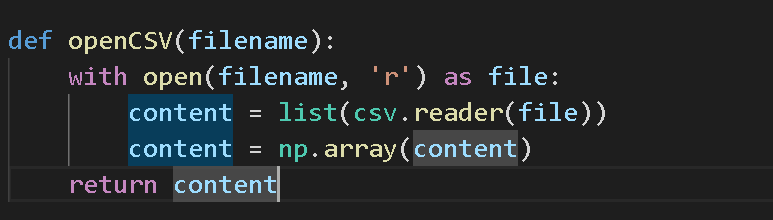
We can clearly see that the range of 95% confidence interval after parameter optimization is relatively small.

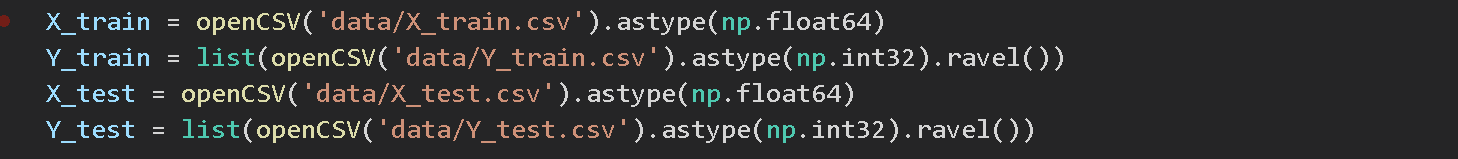
1. Observations and discussion

I found that under some situation variance calculated can be negative value. My current approach is to take absolute value of them.

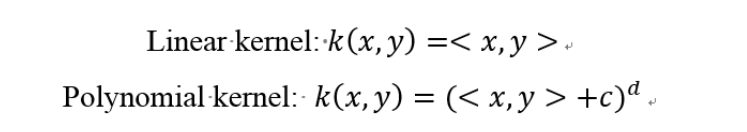
1. SVM
2. code with detailed explanations

○ Part1

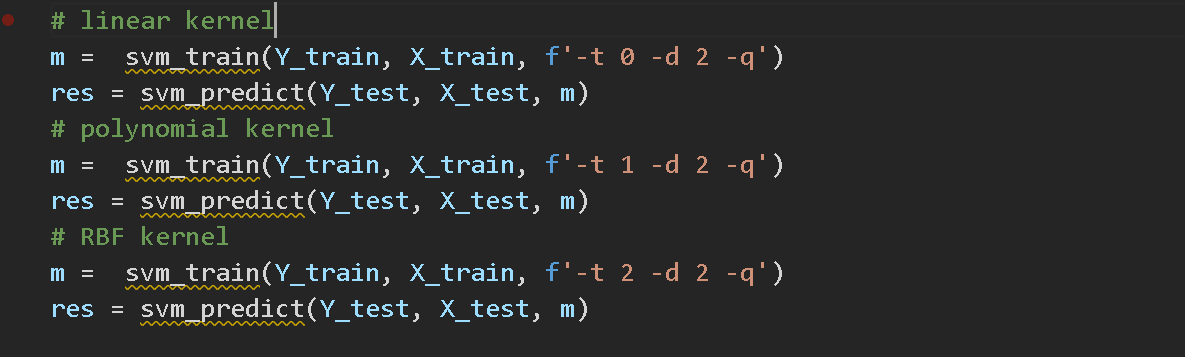




First we need to load our Mnist data to program.

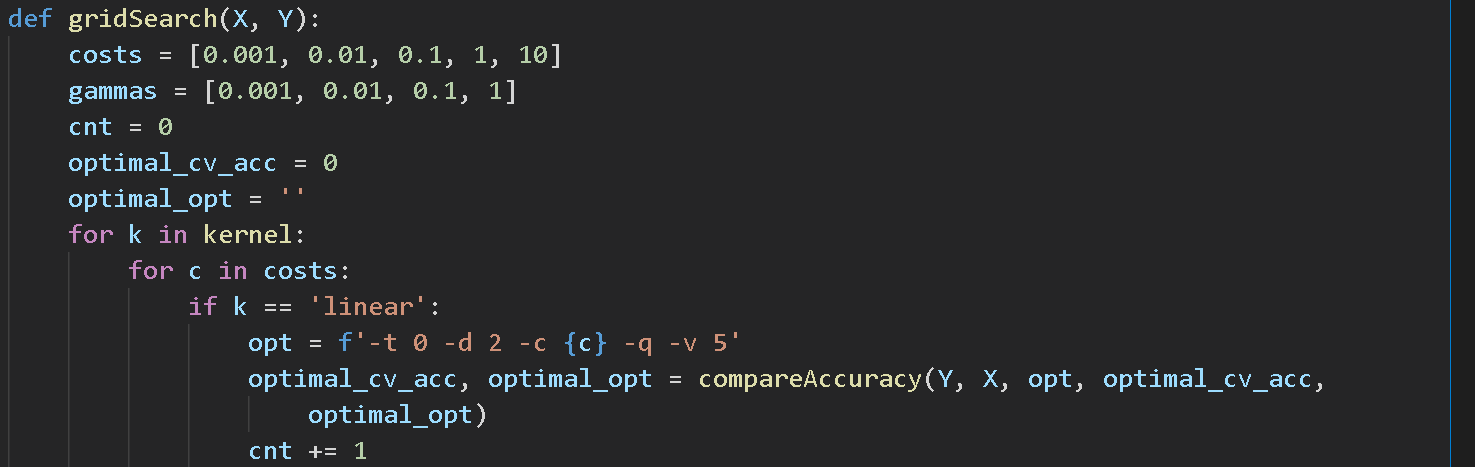


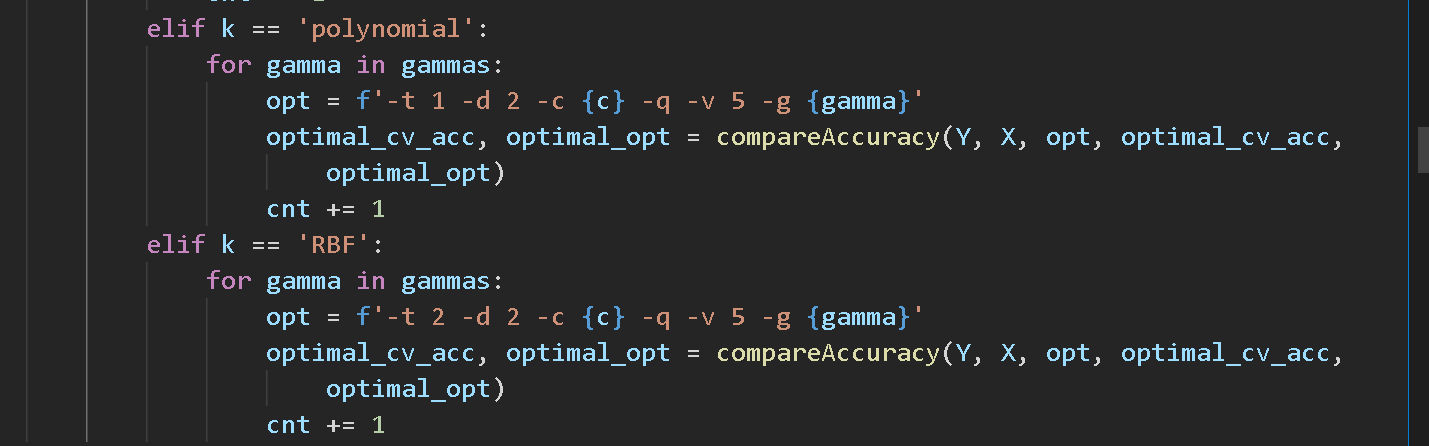


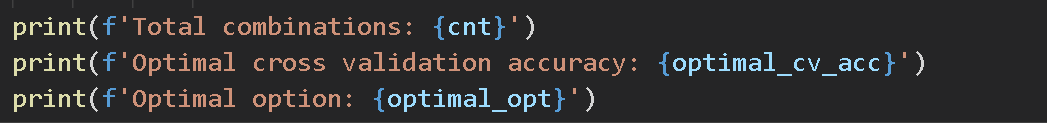


We then train our model with 3 different kernels above and compare their results. Output is in (b) section.

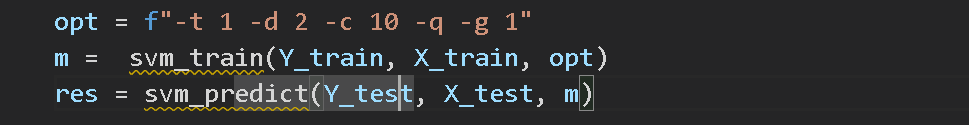
○ Part2





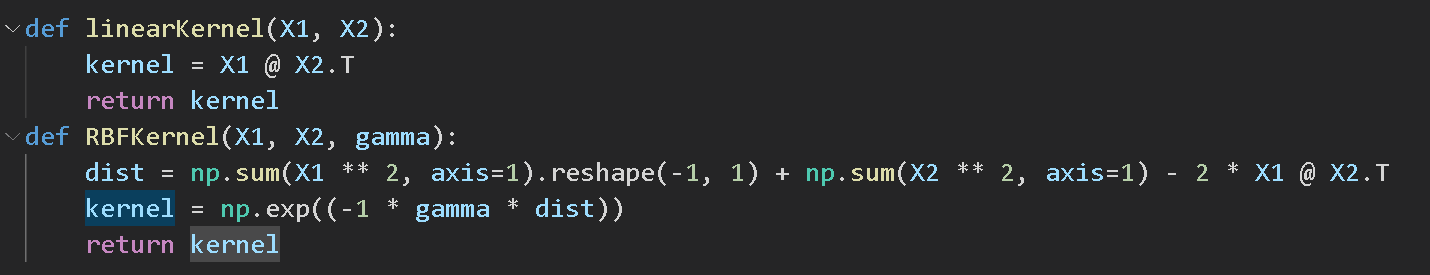


First we need to define our gridSearch function. We iterate over 3 different kernels, 4 different gammas, and 5 different costs value, so we have totally 45 combinations to try.

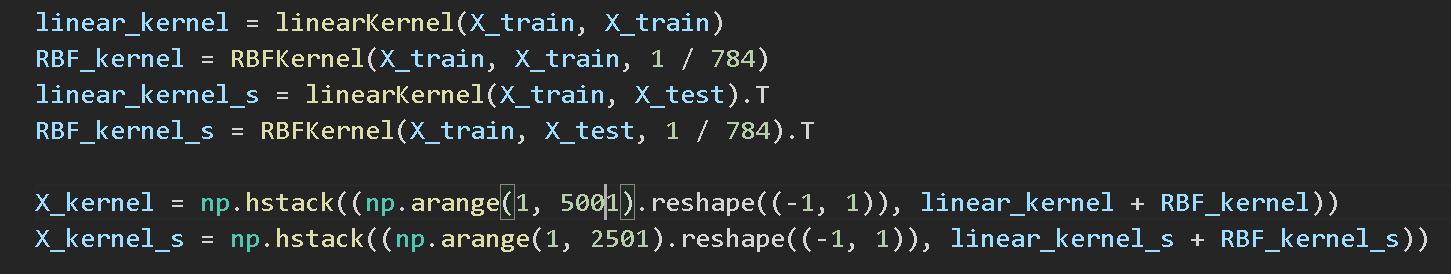


Then we use the best combination we found to train a model and validate on testing data.

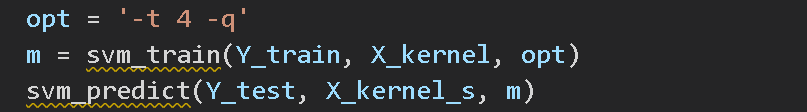
○ Part3



First we use the formula showed above to define 2 different kernels.



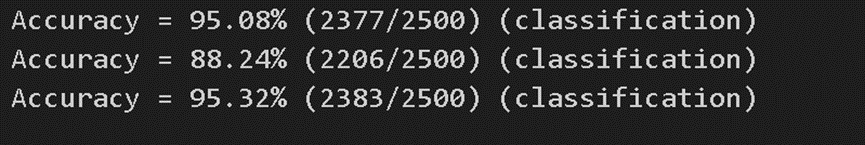
We first need to calculate the k(x, x) and k(x, x\*) and add index to it. We choose 1/num\_features as gamma value based on experience.



t=4 means precomputed kernel. We use this opt to train and validate.

1. experiments settings and results

○ Part1



- Linear Kernel: 95.08% (2377/2500)

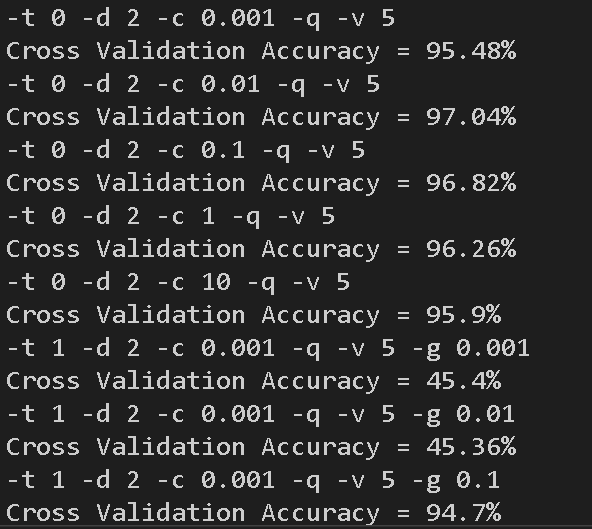
- Polynomial Kernel: 88.24% (2206/2500)

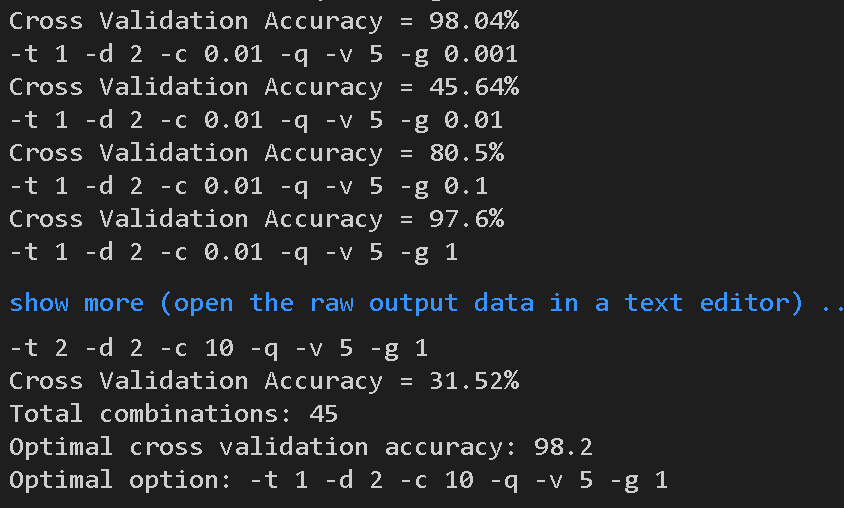
- Radial Basis Function Kernel: 95.32% (2383/2500)

RBF kernel seems the best for this case.

○ Part2

Output is like below.





After we iterate all these combinations, we can get the best combinations of parameters may be Poly kernel, c=10, gamma=1.



We can get 97.68% accuracy. It does surpass the result in part 1.

○ Part3



This result is good enough but didn’t surpass the result in part 2.

1. observations and discussion

Part 1:

The highest accuracy is RBF kernel and lowest is Poly kernel.

Part 2:

An interesting founding is in part 1, Poly kernel has worst result, but after our optimization with gridsearch, The best combination of parameters contains Poly kernel. All in all, as long as there is a good combination of parameters, every kernel should be able to achieve good results.

Part 3:

The result of combination of linear kernel and RBF kernel is same as linear kernel in part 1. It’s because maximum of RBF kernel is 1, but linear kernel can be very large. Because of the linear kernel dominate addition result, the result of this part is same as part 1.

To prove my idea, I try to first normalize linear kernel to (0, 1) then add RBF kernel. I got 95.88% of accuracy which really is different with previous results.