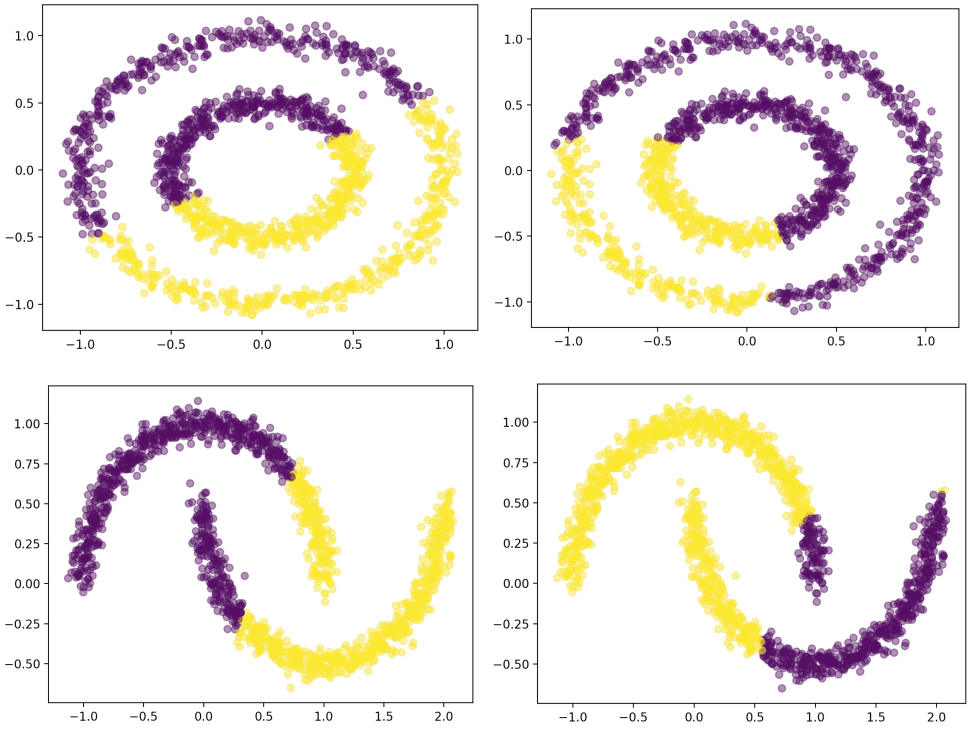
2018/05/24
0556623
Chieh Yu, Chen

HOMEWORK #1

1. FIRST PART

KMeans/ Kernel KMeans

A. You need to make videos showing the clustering procedure (visualize the cluster assignments of data points in each iteration, colorize each cluster with different colors) of your kmeans/kernel kmeans program

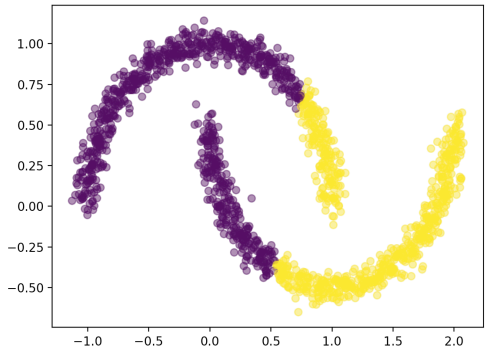


i. Neither linear_kernel(lhs) nor RBF_kernel(rhs) could successfully classify the data. There is no proper decision boundary in the feature space, which means the kernel function is not a proper distance/similarity matrix to evaluate this dataset.

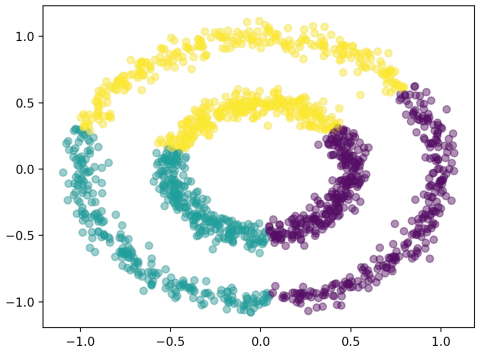
ii. Please see attached videos for the clustering procedure.

B. In addition to cluster data into 2 clusters, try more clusters (e.g. 3 or 4) and show results.

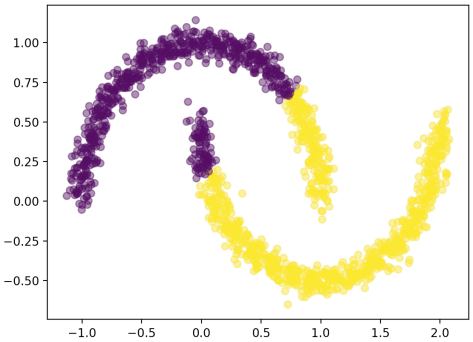
i. 3 Clusters (Kmeans, RBF Kernel KMeans)



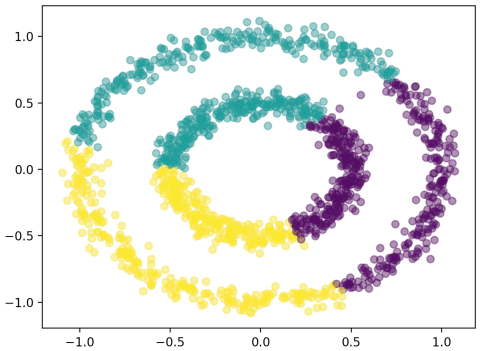
(Kmeans) Center:
[[-1.31881691e-01 5.46679797e-01]
[1.25771804e+00 -9.64700453e-02]
[9.87840031e+02 1.20543406e+02]]



(Kmeans) Center:
[[0.57759572 -0.36799092]
[-0.51821834 -0.41128139]
[-0.19577413 0.60910435]]

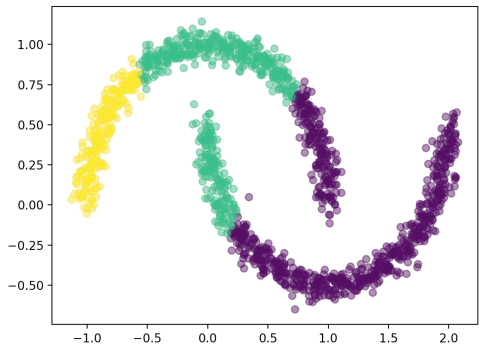


(RBF Kernel) Pseudo center:
with sigma=0.5
[[3.90640063e+02 2.14665072e+02]
[2.95188652e-01 6.51030321e-01]
[1.01975335e+00 8.38648195e-02]]

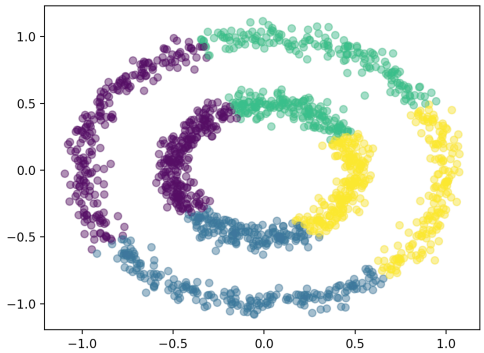


(RBF Kernel) Pseudo center:
with sigma=0.5
[[0.36878415 -0.0675722]
[-0.14780615 0.39857274]
[-0.19452496 -0.29489432]]

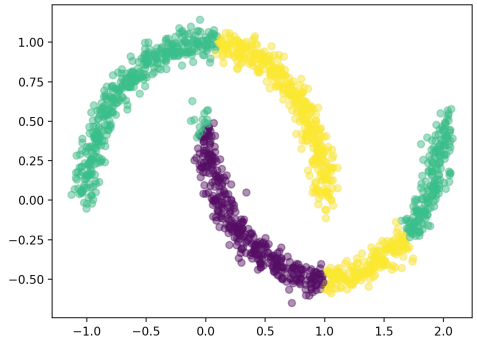
iv. 4 Clusters (Kmeans, RBF Kernel KMeans)



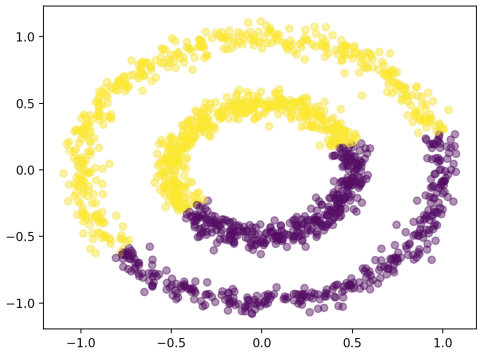
(Kmeans) Center:
[[2.38190644e+02 1.18368567e+02]
[1.28742257e+00 3.44860526e-02]
[2.44323352e+02 1.87113654e+02]
[1.28624525e-01 5.13485472e-01]]



(Kmeans) Center:
[[-0.64213 0.37167032]
[0.02130387 -0.94649517]
[0.25850042 0.68710766]
[0.6581767 -0.13564637]]



(RBF Kernel) Pseudo center:
with sigma=0.5
[[7.78227192e-01 1.03423470e-01]
[3.44241490e+02 1.21572307e+02]
[5.25031328e-01 5.15537265e-01]
[1.05105684e+00 3.90647773e-01]]



(RBF Kernel) Pseudo center:
with sigma=0.5
[[26.93568037 -112.12422513]
[0.17281825 -0.360483]
[-0.16246725 0.27233256]
[-1.48444556 18.29008097]]

2. SECOND PART

C-SVC(Soft-SVM) with LIBSVM library

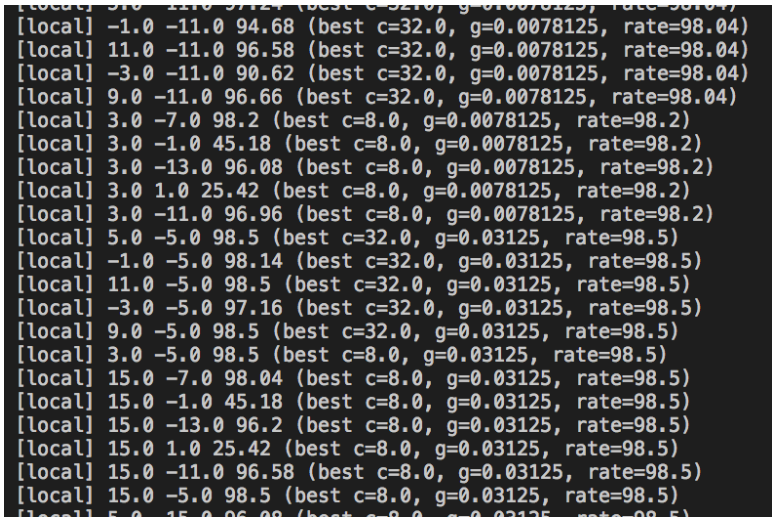
A. Use different kernel functions (linear, polynomial, and RBF kernels) and have comparison between their performance.

(Kernel)	linear	polynomial	RBF
Training accuracy	100.00%	34.34%	96.88%
Testing accuracy	95.08%	34.68%	95.32%

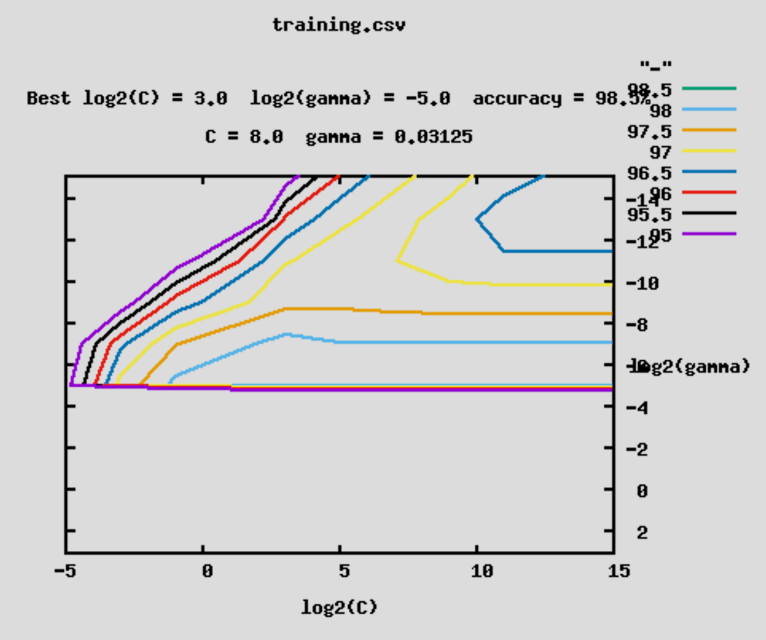
Note:
- polynomial (default: gamma=1/28,coef=0,degree=3)
- RBF (default: gamma=1/28)

B. Please do the grid search for finding parameters of best performing model.

i. By calling the API in "grid.py", I got the best accuracy=98.5% as below, and the first tag [local] is about the training device, three real number comes after the tag, which indicate the parameter C, parameter Gamma and the Accuracy separately, we can see several pairs of [C, Gamma] can achieve the best Accuracy. Note that the range of parameter C is [-5, 15] with step size 2, Gamma is [-15,3] with step size 2 (110 combinations in total), and I use 5-fold cross validation during the whole search process.



(grid search result for each hyperparameter pair of C and Gamma)



(grid search result with contour plot)

C. Use linear kernel+RBF kernel together (therefore a new kernel function) and compare its performance with respect to others.

i. Linear + RBF gains enhancement on accuracy as 95.64%.

(Kernel)	linear	polynomial	RBF	Linear + RBF
Training accuracy	100.00%	34.34%	96.88%	100.00%
Testing accuracy	95.08%	34.68%	95.32%	95.64%

D. Discussion

i. Code Explanation

1. *KMeans*

a. *My implementation of KMeans is in the file "KMeans.py", of Kernel KMeans is in the file "Kernel_KMeans.py"*

b. *There are three types of kernel functions as linear_kernel(), 'sigmoid_kernel(), RBF_kernel() in kernel kmeans implementation.*

c. *In the kmeans() function.*
Firstly, I randomly initialize the one_hot_table that indicate the cluster of each data.
Secondly, by following the class notes, the program will start the training process. In the beginning, I tally the whole gram matrix to save computational power which will be used for every training iterations. Then, update the one_hot_table by finding the smallest kernel distance.
In the end, I get the pseudo-center by mean over the data points within the same cluster. As the terminal condition, the program will break the loop when the pseudo-centers stop updating.

2. *SVM*
    a. *For better utilizing LIBSVM library, I preprocess the data into LIBSVM format in "preprocessing.py" as "data/training.csv" and "data/testing.csv"*
    b. *I implement three functions in "SVM.py".*
        i. test_kernel():
        Testing different kernel with default hyperparameter.
        ii. grid_search():
        By calling API find_parameters() in "grid.py".
        iii. precompute_kernel():
        By following the tutorial in README file, we can customize our own kernel by precomputing the gram matrix(n^2). And my customized kernel is built by summation of linear_kernel() and RBF_kernel() for every data pairs. Then feed them into the LIBSVM with flag isKernel=True.

ii. Observation

1. *KMeans*
Though kernel is powerful, there are few problems I met in this assignment.
    a. *The initial one_hot_table and number of cluster affect results.*
    b. *While implementing KMeans, the most difficult problem is there are too many possible reasons make training failed. Reasons includes bad initialization, hyperparameters, and bad kernel selection.*
    c. *For two dataset in this assignment, "moon.txt" and "circle.txt", both of them are failed even with RBF kernel function. In my opinion, the reason of such failure is because RBF kernel kmeans is alike kmeans, both of them are gaussian form. Though RBF kernel kmeans is nonlinear, it doesn't change the feature space too much, two cluster's center in feature space might be too close to separate. Maybe it could be right by proper initialization and parameter selection, but I didn't get it in this assignment.*
    d. *Kernel kmeans need to maintain the one_hot_table and gram matrix, which result of slower speed than vanilla kmeans.*
2. *SVM*
    a. *Linear SVM is fast and powerful, which can generate competitive performance for most cases.*
    b. *The hyperparameter will affect the result significantly.*
    c. *The gram matrix's complexity is O(n^2), so it cost a long time to build up the whole table with my notebook. I think this section could be paralleled for speeding up the precomputing process.*