2018/05/23

Homework #1

# First Part

KMeans/ Kernel KMeans

## 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

### Please see attached file. ()

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

### 3 Clusters (Linear Kernel, RBF Kernel) (Linear Kernel) Pseudo center: [[ 2.02984708e-01 4.94134478e-01]

### [ 2.53221081e+02 1.60056800e+02]

### [ 1.26789150e+00 1.45969530e-01]]

### 

### (Linear Kernel) Pseudo center: [[-0.10815412 0.42263357]

### [ 0.46638517 -0.01898857]

### [-0.34888111 -0.28362121]]

### (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]]

### 4 Clusters (Linear Kernel, RBF Kernel) (Linear Kernel) Pseudo 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]]

### (RBF Kernel) Pseudo center: with sigma=0.5 [[ 0.54216056 -0.05096302]

### [-0.336176 -0.34934078]

### [-2.50296903 0.20369893]

### [-0.18398494 0.45848129]]

### 

### (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]]

## For the initialization of kmeans clustering used in kmeans/kernel kmeans, try different ways and show corresponding results. TODO

### Uniform random initialization

### Zeros initialization

# Second Part

C-SVC(Soft-SVM) with LIBSVM library

## 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\*28,coef=0,degree=3) - RBF (default: gamma=1/28\*28)

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

### 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, and Gamma is [-15,3] with step size 2. (110 combinations in total)

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

### TODO TODO

## Discussion

### Code Explanation

#### KMeans

##### My implementation of KMeans is in the file “KMeans.py”

##### There are three types of kernel functions as linear\_kernel(), ‘sigmoid\_kernel(), RBF\_kernel().

##### 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.

#### SVM

##### For better utilizing LIBSVM library, I preprocess the data into LIBSVM format in “preprocessing.py”

##### I implement three functions in “SVM.py”.

###### test\_kernel():

###### Testing different kernel with default parameter.

###### grid\_search(): By calling API in “grid.py”.

###### precompute\_kernel(): By following the tutorial in README file, we can customize our own kernel by precomputing the gram matrix.

### Observation TODO

#### KMeans Though kernel is powerful, there are few problems I met in this assignment.

##### The hyperparameter will affect the result significantly.

##### The initial one\_hot\_table also affect result.

##### 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.

#### SVM

##### Linear SVM is fast and powerful, which can generate competitive performance for most cases.