Computational Machine Learning Homework4

Yun-shao Sung*

Abstract. This document served as the purpose of answering qestions from homework assignment, and also conclude the experiment observations.

1. Kmeans. In this section, I used the iris data set and run my regular kmeans. There is interesting thing I observed that as the k (number of clusters) gets large, there is certain pobility that the cluster cannot get points within that cluster. I think this is mostly due to the random process for picking the initial points. As there are too many clusters, or the data set is very similar spreading in limited dimension space, the cluster centroids will be too close and therefore there are situations cluster have no point. For this particular iris dataset and current random seed, cluster-no-point happened when k gets to 8. Since now I used random seed, the effect of low k was not able to observed. However, regular kmeans is randomly initialize the points, low k may yield a very different distortion results after running, and, therefore, it cannot guarantee you the best representive of centroid with minimum distortion.

2. mykmeans-multi. Function mykmeans-multi is inherent from regular kmeans, but will save the distortion and centroids during each iteration. When finising running, with return the best centroid with lowest distortion. In this way, this function will make sure return the best minimum to represent the data distribution. It's hard to say whether this is more preferable than regular kmeans, since the data set is relative small in only 4-dimention space. However, I think using mykmeans-multi is always better than regular kmeans, such that we won't get the non-optimon result during the run. As making the plot for distortions and centroids trajectory, we can see from the distortions that it improve very fast during the first few iteration, and improvement become flat between iteration 3 to 5, which imply the it is now within a low slope region. Then distortion got a bit more improve and remained all the same.

3. mykmeans++. Function mykmeans++ is doing more delicate considerations when initialize the points. Insteresting enough, although at the very fist point is still based on random, all the remaining points were pickup base on the possibility porpotional to the distance to the closet centroid, therefore we can see the initial distortion is higher than mykmeans-multi, but the final result is much better than kmeans-multi. I think this is due to the nature of kmeans++ that trying to pick the points that are far enough, and we can skip the situaion that get stock in local minimum. Furthermore, the cluster-no-point sitation will not happened even the k=50, and the distortion can get even improved from 80(k=10) to 34(k=50).

4. sklearn.cluster.KMeans tricks. While reading the code in sklearn.cluster.KMeans, I noticed it save many computations when handeling the for-loop, mostly by Cython. Therefore, I tried to improve my code when running for loop by either lamda or put it in numpy.array to compute.

*yss265@nyu.edu

5. Bag-of-words tutorial. I think the tutorial of Bag-of-words is very key to understand to last part of assignment. In the Bag-of-words tutorial, we imported the movie reviews, and use CountVectorizer to vectorize the counts of every token. Then feed into TfidfTransformer to have the idea of important key words or not that importants words. Therefore, this will be the information when train the classifier. Furthermore, I noticed the choise of classifier will also yeild noticable difference. I got the prediction of 77.4% from naive bayes to 82.4% from SVM.

6. Reading Scikit-Learn. Since I am useing kmeans++ as the base-method finding centroids and it's performance and accuracy will very affect out classification result, I have checked the code of MiniBatchKMeans, especially the method finding the initial centroids. In the method called _init_centroids, there is trick parameter init_size, which is to lower down the number of samples to randomly sample for speeding up the initialization, but this will sometimes at the expense of accuracy. Also this method was also observed in _k_init that instead of the whole samples, it used n_local_trials (default: 2+log(k)) to do sampling. _k_init is the method used to get the init centers fro kmeans++, and seems there is also a trick that ckecking whether point space X is sparse, and make it toarray and get the specific point if X is sparse, otherwise just keep as it is to get the point (X[center_id]). Also seem there is a faster way called pairwise.euclidean_distances that can directly compute the distance matrix between each pair of vectors, and the advantage will be computationally efficient when dealing with sparse data.

7. Project. So here I have used the same learnvocabulary and getbof method that have been used previously and applied to very small set of our music data, so it's easiler to my testing purpose and easier to debug. So the small data set is 10 genres, and 8 songs per genre. In notebook kmeans_music.ipynb, I used MultinomialNB as my classifier and the fit and training process went well, but got 0 prediction score. Then in KNeighborsClassifier_music.ipynb I used KNeighborsClassifier combined with grid_search to train the classifier. The best parameter for n_neighbors: 6, weights: distance, and algorithm: auto. Then the prediction score I got is 0.2 which is better than MultinomialNB. The phenenome is also observed in my notebook BagWordsTutorial.ipynb that SVM has better prediction than MultinomialNB. Below are the experiments that I would like to test

1. The effect if I reuse the same data for training classifier
2. The effect transpose the data point