ARTIN 2020/10/14 Lab　M1 JEMARO　name: Yuki Saito　student number: 200305D

3: Euclidean space clustering

1.. Use the kmeans algorithm (sklearn.cluster.KMeans)

please look Jupyter notebook.

2. what is the AMI for the synthetic and real datasets ?

For synthetic dataset, we use kmeans clustering with k = 10. The AMI score is 1.0. Also, for real dataset, we use kmeans with k = 8. The AMI score is 0.916. Synthetic dataset have small number of data, 1000 and its dimension is also small 2. So, kmeans can perfectly cluster the data as its groundtruth. On the other hand, Real dataset have 2400 data and its dimension is as much as 1024. So compared with synthetic data, it is considered to have lower AMI score because of its complication.

4 Model selection

1. Implement the gap statistic criterion. You can use the inertia given by kmeans as the loss function.

please look Jupyter notebook.

2. Verify that the statistic performs well on the synthetic dataset.

Figrue1 shows the gap statistic result of kmeans for synthetic dataset. Top shows the relationship between cluster size k and gap statistic result. Bottom shows the relationship between k and AMI scores of kmeans. k is in range 1 to 20. We can see the gap statistic plot has the highest values at k = 10. This means k = 10 is the suitable cluster size in terms of gap statistic. Also, k = 10 have the highest AMI score among k = 1, 2, 3, … 20. We can say the gap statistic is a effective validation for deciding the suitable cluster size for sure.

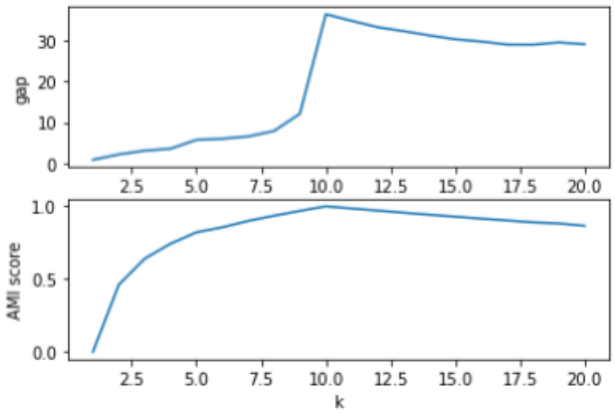

Figure1: the relationship between cluster size k and gap statistic result of kmeans for synthetic dataset

(Top: gap statistic , Bottom; AMI score)

3. Increase the overlapping between the clusters. Is the statistic robust ?

From Figure1 above, there is a huge gap between k <= 9 and k >= 10. To increase the overlapping, we have to set the smaller cluster size. So, if we decrease cluster size k from 10 to 1 for example, the gap indicator drops significantly at k = 10->9 and then, slowly decreasing. Increase the overlapping lead to increasing the cluster inertia (loss function) , and the accuracy is decreasing as shown in Figure1 bottom graph. The gap statistic can predicts this phenomena in small k , so the statistic is considered to be robust enough.

4. Apply the statistic to detect the number of clusters for the real data. Is the detected number of clusters correct ?

Figrue2 shows the gap statistic result of kmeans for real dataset. Top shows the relationship between cluster size k and gap statistic result. Bottom shows the relationship between k and AMI scores of kmeans. k is in range 1 to 50. In contrast to synthetic data, the gap statistic plot is almost increasing from k = 1 to 50. so by using the derivatives of this plot(curve), the slop starts convergence around k = 9, 10. Since, I predict k = 10 is the suitable size.

As shown in Bottom graph, the AMI score has the highest value at k = 7. So, the my prediction is not so far from the correct. But cannot detect the most preferable cluster size. Complicated data like this real dataset, the gap statistic cannot always detect the correct size though we can detect the correct k for synthetic dataset.

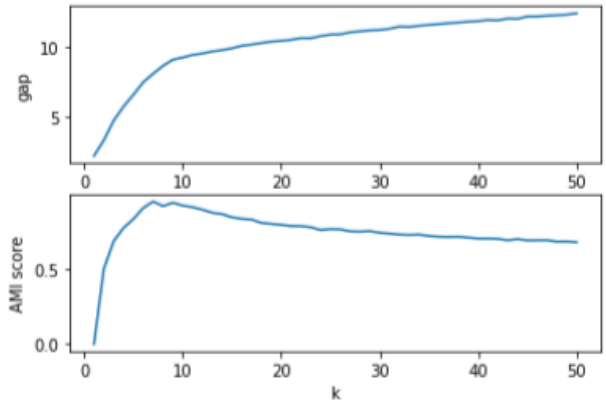

Figure2: the relationship between cluster size k and gap statistic result of kmeans for real dataset

(Top: gap statistic , Bottom; AMI score)

5 Non Euclidian clustering

1. Consider a radial basis function to compute the dissimilarity matrix:

please look Jupyter notebook.

2. Implement the spectral clustering algorithm using the eigenvalue and the kmeans implementation.

please look Jupyter notebook.

3. Verify that the eigenvector decomposition of the laplacian of the synthetic dataset is step wise.

please look Jupyter notebook.

4. Is it the same for the real dataset ?

please look Jupyter notebook.

5. Compare the results with the reference implementation available sklearn.cluster.SpectralClustering.

I compare my implementation with scikit-learn Spectral Clustering with gamma = 1, affinity='rbf'. I used synthetic dataset and the cluster size k is in range 1 to 30. Figure3 left shows the prediction accuracy of both implementation. Both looks like almost all the same. There is a little bit difference from k >= 10 and the scikit-learn is decreasing compared with my implementation. There may be some difference between them , for example "(a)(b) compute the weight matrix W using the radial basis function". For my implementation,

Function : nearest_neighbor_graph(X, gamma=1 makes the Euclidian distance matrix and applies rbf for the whole matrix. But as we saw in the lecture slide, the weight matrix should only have rbf values for neighbor nodes and the other should be zero.

Then, Figure3 right shows the AMI score for apply rbf for only neighborhood node of Euclid distance matrix. I set the neighbor node num is 100. Compared with Figre3 left, the difference of my implementation and scikit-learn implementation is smaller.

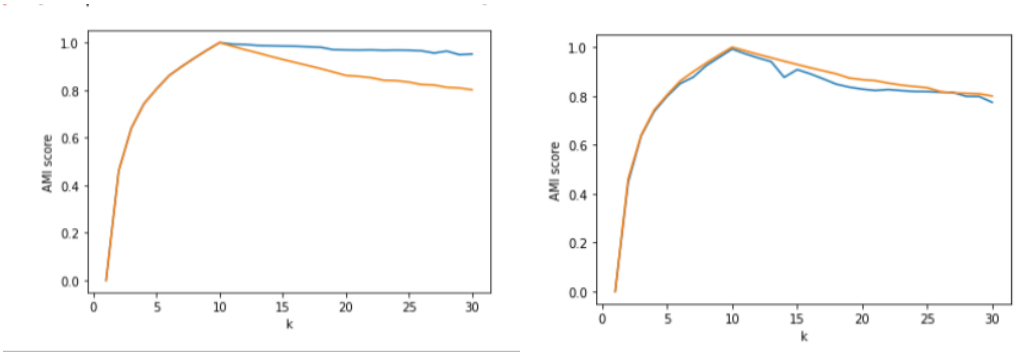

Figure3: the relationship between cluster size k and AMI score of spectral clustering for synthetic dataset Left is the result when we apply rbf for whole Euclid distance matrix, and the Right is the result when we apply rbf for only neighborhood node of Euclid distance matrix. (blue: my implementation, orange: scikit learn implementation)

6. Optimize gamma for best performance.

Figure4 shows the relationship between gamma values of rbf function and AMI score of spectral clustering. I used my implementation and kmeans clustering part with k = 10, for both synthetic dataset and real dataset.

I set gamma from 0.1, 0.2, 0.3, …2.0 in range 0~2 and search the optimized gamma. Figure4 left shows the synthetic dataset result and right shows real dataset result. For synthetic dataset, the AMI score has highest values at 0.6 < gamma < 1.0. For real dataset, it has maximum values at 0.5 < gamma < 1.0. So, to get best performance, we should set gamma around 0.6 ~ 1.0.

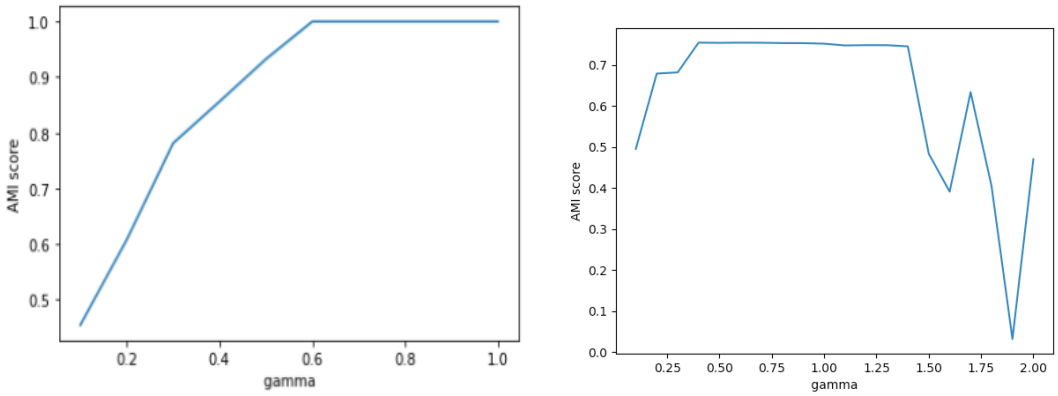

Figure3: the relationship between sigma of rbf function and the AMI score of spectral clustering for synthetic dataset and real dataset. Left: synthetic dataset, Right: real dataset.

6 Conclusion

1. Present a table compiling the performance achieved by the different clustering algorithms on the two datasets.

Table 1 shows the comparison of different clustering algorithms on two dataset in this lab.

At first, the normal kmeans algorithm perform well on synthetic dataset, which only have 1000 data with dimension 2. By using the gap statistics, we can predict the suitable cluster size. But, for real dataset, which is more complicated than synthetic, it is more difficult to apply suitable kmeans. If we set larger cluster size for real dataset, the AMI scores are decreasing and we have to select k = 10 for best performance even though gap statistic is increasing all the way. Also, kmeans is not so robust to some outliers in the input data since this algorithm calculate gravity point of each clusters.

Secondary, the Spectral Clustering can perform well on synthetic dataset. Spectral clustering use feature reduction and then, apply kmeans, which gives us data smoothing benefits. So, this algorithm require less assumption of input data form. However, we have to tune hyperparameter k , sigma of rbf etc… for best performance.

Table 1: Comparison of different clustering algorithms on real and synthetic dataset

Dataset type	data num	data dimension	Kmeans	Spectral Clustering
synthetic	1000	2	・perform well on simple form data. ・Gap statistic is valid. Gap statistic is valid.	・perform well . ・we have to select suitable k and sigma for rbf.
real	2400	1024	・perform worse than synthetic data, since it has more complicated form. ・Gap statistic is less effective than synthetic data.	・perform worse than synthetic data and normal kmeans ・we have to select suitable k and sigma for rbf.

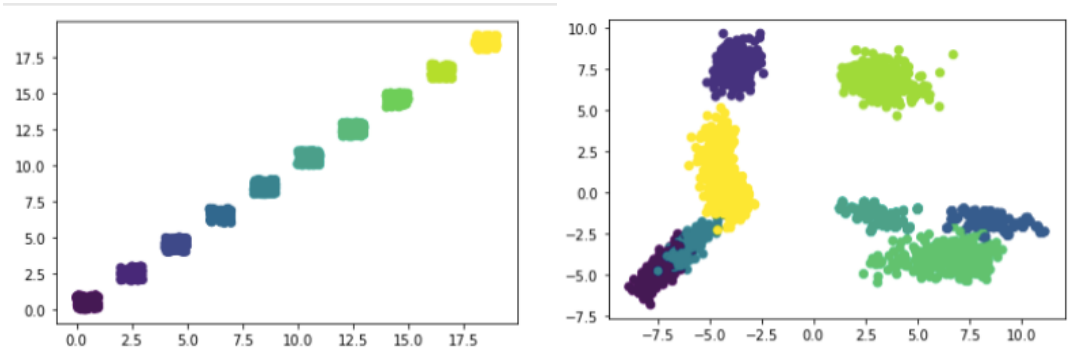

Figure4 data form of synthetic and real dataset. (real dataset is projected to 2dims using PCA)

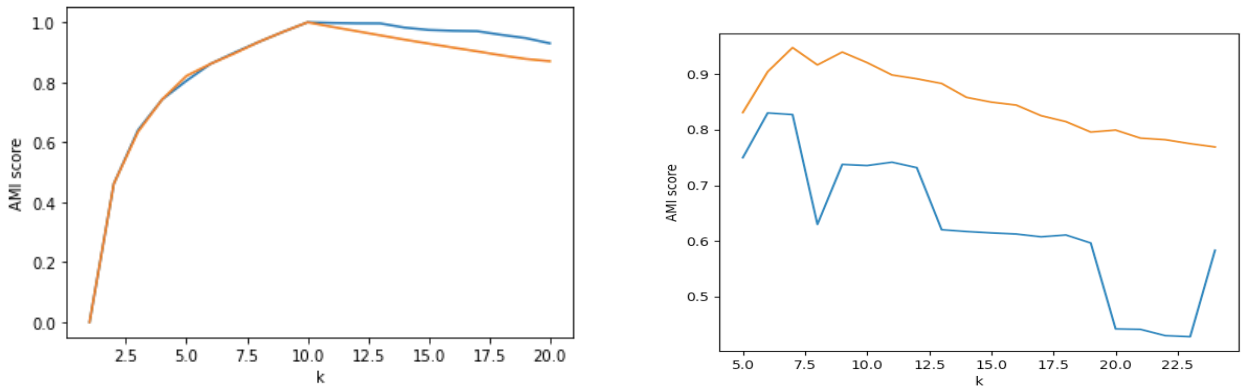

Figure5: the comparison of performance on synthetic and real dataset with 2 different clustering algorithm. Left graph shows synthetic data result and right shows real data. Vertical axis is AMI score and horizontal is cluster size k.

(blue: Spectral clustering by my implementation, orange: kmeans clustering by scikit-learn implementation)