Report

In homework 4, you have to compare four methods on two different data sets. The four methods are:

- **Kmeans**
  The classical method in clustering. It construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square distances. *Kmeans* minimizes with respect to distance measure. Using Euclidean Distance to find the results.

- **Kmedoids**
  The Kmedoids algorithm is a clustering algorithm related to the Kmeans algorithm and the medoid shift algorithm. Both the Kmeans and Kmedoids algorithms are partitional (breaking the dataset up into groups) and both attempt to minimize squared error, the distance between points labeled to be in a cluster and a point designated as the center of that cluster. In contrast to the Kmeans algorithm, Kmedoids chooses data points as centers.

- **Gaussian Mixture Model**
  Gaussian Mixture Model (GMM) is among the most statistically mature methods for clustering. It is one of the most popular clustering methods which can be viewed as a linear combination of different Gaussian components. You need to handle the singular issue of the covariance matrix (Use *PCA* to do the dimensionality reduction first).

- **Spectral Clustering**
  Spectral clustering techniques make use of the spectrum of the similarity matrix of the data to perform dimensionality reduction for clustering in fewer dimensions. Spectral Clustering is very simple to implement and can be solved efficiently by standard linear algebra methods. You should compare the results between unnormalized spectral clustering and normalized spectral clustering.

The test datasets provided are as follow:

- **MNIST database**
  The MNIST database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples. Sample images are showed in Figure1. Given a cluster number,

  there are 10 randomly cases (except the case when the entire data set is used). Each case file contains variables 'sampleIdx' and 'zeroIdx'. The following matlab codes can be used to generate the particular set

  ```
  ==========================================
  fea = fea(sampleIdx,:);
  gnd = gnd(sampleIdx,:);
  fea(:,zeroIdx) = [];
  ==========================================
  ```

  You should do the clustering when the cluster number is 5, 7, 10 (entire dataset), respectively. You need to run 10 different files in 5 class and 7 class and record all the results. We have divided the data into 5 class and 7 class in advance, which means each file in 5 class / 7 class folder just contains 5 / 7 different labels. The entire dataset is named MNIST.mat.

- **COIL20 database**
  It contains 20 objects. The images of each objects were taken 5 degrees apart as the object is rotated on a turntable and each object has 72 images. The size of each image is 32x 32 pixels, with 256 grey levels per pixel. Thus, each image is represented by a 1024-dimensional vector. Sample images are showed in Figure2. You should do the clustering when the cluster number is 5, 10, 20, respectively. The entire dataset is named COIL20.mat.

Solution

1. Kmeans:

MNIST-Accuracy Results

	1	2	3	4	5	6	7	8	9	10
5-class	87.90%	82.02%	81.68%	72.42%	71.17%	75.07%	66.28%	72.01%	61.39%	70.41%
7-class	71.17%	77.96%	70.55%	59.75%	65.52%	63.87%	68.94%	67.65	65.46%	66.58%
10-class	59.58%									

MNIST-Mutual Information Results

	1	2	3	4	5	6	7	8	9	10
5-class	0.7106	0.5941	0.6264	0.5257	0.5143	0.5013	0.4517	0.5080	0.4399	0.5723
7-class	0.5327	0.5982	0.5206	0.4611	0.5168	0.5107	0.5615	0.5231	0.5024	0.4947
10-class	0.4977									

COIL20-Accuracy Results

	1	2	3	4	5	6	7	8	9	10
5-class	90.83%	71.94%	100%	95.83%	100%	100%	90.00%	100%	90.28%	82.22%
10-class	63.61%	67.78%	79.86%	87.92%	85.42%	92.36%	79.44%	74.86%	62.08%	68.19%
20-class	69.38%									

COIL20-Mutual Information Results

	1	2	3	4	5	6	7	8	9	10
5-class	0.8508	0.7046	1.0000	0.9099	1.0000	1.0000	0.8345	1.0000	0.8549	0.7068
10-class	0.6483	0.6903	0.8102	0.9127	0.8662	0.9143	0.7842	0.7703	0.6280	0.7033
20-class	0.7626									

Time Results

	5-class	7-class/10-class	10-class/20-class
MNIST	1.44s	2.54s	3.45s
COIL20	0.05s	0.26s	1.89s

Analysis:

To minimize the sum of square distances in Kmeans, you should repeat the process for several times picking starting points randomly (I tested each data file for 20~50 times). Then we can get the approximately global minimum. However, in our testing, I found that the minimum sum of square distance sometime will not lead to the best accuracy and MI, meanwhile the best accuracy sometime will not means the best MI, so I just choose the result that both good enough in AC and MI.

Also, according to the results, we find that the more classes to cluster, the harder this algorithm to achieve good results. In some of the 5-class data files of the COIL20 data set, we get 100% AC and 1 MI, I think that's because the data points in different clusters are relatively far away from each other and the data points in the same cluster are relatively close to each other.

The executing time almost depends on the number of iterations of one clustering process each time, while the number of iterations will vary every time since the time when the iteration stops is not solid. So the results are just the average results.

2. Kmodoids:

MNIST-Accuracy Results

	1	2	3	4	5	6	7	8	9	10
5-class	77.98%	67.38%	73.48%	63.74%	67.32%	66.15%	61.71%	64.55%	61.70%	67.23%
7-class	61.29%	64.29%	58.86%	53.15%	61.64%	56.52%	63.34%	62.45%	57.35%	57.08%
10-class										

MNIST-Mutual Information Results

	1	2	3	4	5	6	7	8	9	10
5-class	0.5463	0.4096	0.4978	0.3688	0.4239	0.4066	0.3471	0.4093	0.3703	0.4665
7-class	0.3917	0.4229	0.3631	0.3380	0.3868	0.3534	0.4487	0.4122	0.3898	0.3670
10-class										

COIL20-Accuracy Results

	1	2	3	4	5	6	7	8	9	10
5-class	81.11%	76.39%	93.33%	83.33%	87.78%	86.94%	91.39%	92.50%	86.39%	73.61%
10-class	53.47%	62.50%	73.61%	81.11%	84.03%	91.94%	77.92%	72.64%	60.97%	58.89%
20-class	62.43%									

COIL20-Mutual Information Results

	1	2	3	4	5	6	7	8	9	10
5-class	0.6984	0.6098	0.9008	0.7437	0.8332	0.8161	0.8618	0.8848	0.7497	0.5915
10-class	0.5195	0.6114	0.7225	0.7891	0.8340	0.9011	0.7623	0.7114	0.6052	0.5911
20-class	0.69.99									

Time Results

	5-class	7-class/10-class	10-class/20-class
MNIST	13.6s	23.3s	27.3s
COIL20	0.45s	1.83s	5.76s

Analysis:

In theory, the result of the Kmodoids will be better than the Kmeans's, since the Kmodoids algorithm avoids the impact of the outliers. But in our experiment, the results of the Kmodoids are generally worse than the Kmeans'. It probably because the total sum of the square distance is not that small compared to the Kmeans'.

To the running speed, since the way that the Kmodoids calculates the centers need $O(n^2)$ time complexity while the Kmeans need only O(n), so the speed of the Kmodoids is slower than then Kmeans'.

3. Gaussian Mixture Model:

MNIST-Accuracy Results

	1	2	3	4	5	6	7	8	9	10
5-class	88.72%	74.57%	85.93%	73.58%	62.80%	71.58%	59.64%	71.71%	51.58%	65.84%
7-class	69.43%	71.88%	66.29%	52.07%	64.53%	62.67%	58.79%	65.47%	52.96%	62.55%
10-class	50.05%									

MNIST-Mutual Information Results

	1	2	3	4	5	6	7	8	9	10
5-class	0.7137	0.5160	0.7136	0.5914	0.4592	0.5073	0.3970	0.5614	0.2684	0.5165
7-class	0.5506	0.5418	0.5202	0.3654	0.5109	0.5087	0.4768	0.4939	0.3637	0.4622
10-class	0.4527									

COIL20-Accuracy Results

	1	2	3	4	5	6	7	8	9	10
5-class	58.86%	63.33%	100%	74.17%	78.06%	70.28%	89.17%	71.39%	81.94%	73.89%
10-class	54.58%	52.08%	55.00%	68.19%	82.64%	79.72%	66.25%	58.61%	59.86%	61.81%
20-class	73.89%									

COIL20-Mutual Information Results

	1	2	3	4	5	6	7	8	9	10
5-class	0.5882	0.5224	1	0.7252	0.7500	0.8277	0.8070	0.7660	0.7910	0.5849
10-class	0.5966	0.6111	0.6779	0.8157	0.8708	0.8683	0.7710	0.7076	0.6258	0.6543
20-class	0.7219									

Time Results

	5-class	7-class/10-class	10-class/20-class
MNIST	4.56s	4.77s	9.82s
COIL20	1.24s	1.52s	5.36s

Analysis:

The accuracy and the speed depend much on the feature dimension of the data. Therefore, to gain higher accuracy, you should try different parameters for the dimension reduction. The result I display here may not be the best result since I just have tried several dimension numbers. However, from the present result, we can believe that the GMM algorithm is not bad than the Kmeans. Moreover, we can infer that the feature of data have great impact on this algorithm, that's because some dimensions of the feature here may not follow the Gaussian Distribution and may also have little contribution to the clustering, so dimension reduction is very much needed in this algorithm.

Also, the initial parameters for EM affect the result, which means the GMM may not get the global optima.

As to the speed issue, the more dimension the more time to calculate. So the result just the average result of my experiment.

Ps: The numbers I have tried for dimension reductions are as follows, each data set has different optima. [50,80,100,120,150,180,200,220,250,280,300,350,400]

4. Spectral Clustering:

4.1. Unnormalized Spectral Clustering:

MNIST-Accuracy Results

	1	2	3	4	5	6	7	8	9	10
5-class	87.90%	82.02%	81.68%	72.42%	71.17%	75.07%	66.28%	72.01%	61.39%	70.41%
7-class	69.25%	77.79%	56.87%	57.43%	55.07%	60.96%	64.81%	56.59%	60.93%	59.33%
10-class	50.48%									

MNIST-Mutual Information Results

	1	2	3	4	5	6	7	8	9	10
5-class	0.7009	0.3953	0.6265	0.4772	--	0.4566	0.4217	0.4845	0.3885	0.5515
7-class	0.5199	0.5926	0.4780	0.4332	0.4670	0.4875	0.4875	0.4656	0.4664	0.4783
10-class	0.4327									

COIL20-Accuracy Results

	1	2	3	4	5	6	7	8	9	10
5-class	68.85%	73.24%	100%	75.42%	88.25%	70.28%	89.17%	71.39%	81.94%	73.89%
10-class	54.58%	52.08%	55.00%	77.45%	82.64%	79.72%	66.25%	58.61%	59.86%	61.81%
20-class	73.89%									

COIL20-Mutual Information Results

	1	2	3	4	5	6	7	8	9	10
5-class	0.6882	0.6235	1	0.7252	0.8506	0.8277	0.8070	0.7660	0.7910	0.5849
10-class	0.5966	0.6111	0.6779	0.8157	0.8708	0.8683	0.7710	0.7076	0.6258	0.6543
20-class	0.7219									

Time Results

	5-class	7-class/10-class	10-class/20-class
MNIST	91.1s	129.3s	271.4s
COIL20	5.58s	36.7s	58.5s

4.2.Normalized Spectral Clustering:

MNIST-Accuracy Results

	1	2	3	4	5	6	7	8	9	10
5-class	87.32%	53.85%	81.83%	69.04%	--	70.85%	54.11%	67.68%	49.34%	62.41%
7-class	69.25%	77.79%	56.87%	57.43%	55.07%	60.96%	64.81%	56.59%	60.93%	59.33%
10-class	51.85%									

MNIST-Mutual Information Results

	1	2	3	4	5	6	7	8	9	10
5-class	0.7106	0.5941	0.6264	0.5257	0.5143	0.5013	0.4517	0.5080	0.4399	0.5723
7-class	0.5199	0.5926	0.4780	0.4332	0.4670	0.4875	0.4875	0.4656	0.4664	0.4783
10-class	0.4327									

COIL20-Accuracy Results

	1	2	3	4	5	6	7	8	9	10
5-class	81.11%	76.39%	93.33%	83.33%	87.78%	86.94%	91.39%	92.50%	86.39%	73.61%
10-class	54.58%	52.08%	55.00%	77.45%	82.64%	79.72%	66.25%	58.61%	59.86%	61.81%

20-class	73.89%

COIL20-Mutual Information Results

	1	2	3	4	5	6	7	8	9	10
5-class	0.6882	0.6235	1	0.7252	0.8506	0.8277	0.8070	0.7660	0.7910	0.5849
10-class	0.5966	0.6111	0.6779	0.8157	0.8708	0.8683	0.7710	0.7076	0.6258	0.6543
20-class	0.7219									

Time Results

	5-class	7-class/10-class	10-class/20-class
MNIST	18.3s	41.84s	86.9s
COIL20	0.22s	2.25s	7.61s

Analysis:

The spectral clustering is supposed to be better than the previous algorithm, but in my experiment, the result is just so-so. Maybe it is related to the similarity matrix I used. I use the sum of square distance us similarity matrix, it may not be good for the algorithm.

As to the normalized, when the L matric is not normalized, it may lead to singular matrix or badly scaled matrix, and the eigenvalues may not converged so that the result may be inaccurate. After normalization, the problem is solved. And the speed is faster.

Since the algorithm is using the kmeans to do clustering after dimension reduced, so the result may be the local optima, that may be one of the reason why the result is not so good.