Machine Learning Homework 2

Sun Hao
Student ID: 516030910362

1 PCA algorithm

Give at least two algorithms that could take data set $X = \{x_1, x_2, \cdots, x_N\}, x_t \in \mathbb{R}^{n\times 1}, \forall t$ *as input, and output the first principal component* $\mathbf{w}$*. Specify the computational details of the algorithms, and discuss the advantages or limitations of the algorithms*

Solution

Let us formally describe the problem. **Suppose that X has been decentralized.** The first principal component unit vector is $\mathbf{v}$, which means that $\mathbf{w} = \left(\mathbf{v}^T\mathbf{x}_i\right)\mathbf{v}$. The object function (maximum variance direction, equal to minimum squared error) and the constraint are as below

$$\begin{array}{ll} \max\limits_{\mathbf{v}} & \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{v}^T\mathbf{x}_i\right)^2 = \frac{1}{n}\mathbf{v}^T\mathbf{X}\mathbf{X}^T\mathbf{v} \\ \text{s.t.} & \mathbf{v}^T\mathbf{v} = 1 \end{array} \tag{1}$$

1.1 Eigen Decomposition

1. Use Langragian method to solve problem 1, note that removing the coefficient of object function gets the same result. Then the Langragian form is

$$\mathcal{L}_{\mathbf{v}} = \mathbf{v}^T\mathbf{X}\mathbf{X}^T\mathbf{v} + \lambda\left(1 - \mathbf{v}^T\mathbf{v}\right) \tag{2}$$

   When Partial derivative of $\mathcal{L}$ with respect to $\mathbf{v}$ is 0, $\mathbf{v}$ gets its extremum.

$$\frac{\partial\mathcal{L}_{\mathbf{v}}}{\partial\mathbf{v}} = \mathbf{X}\mathbf{X}^T\mathbf{v} - \lambda\mathbf{v} = 0 \tag{3}$$

2. From Equation 3, we have $\mathbf{X}\mathbf{X}^T\mathbf{v} = \lambda\mathbf{v}$, which means that λ is eigen value of matrix $\mathbf{X}\mathbf{X}^T$, $\mathbf{v}$ is the eigen vector of corresponding eigen value.

   We do eight decomposition to matrix $\mathbf{X}\mathbf{X}^T$, to gain the largest eigen value, then compute the corresponding eigen vector.

$$\left|\mathbf{X}\mathbf{X}^T - \lambda\mathbf{I}\right| = 0 \tag{4}$$

   where $\mathbf{I}$ represents the unit matrix.

3. Solve Equation 4, and we gets the largest λ and then gets corresponding $\mathbf{v}$

   Compute the first principal component $\mathbf{w} = \left(\mathbf{v}^T\mathbf{x}_i\right)\mathbf{v}$

1.1.1 Advantages

The algorithm is accord with intuition. And the algorithm is easy to understand, which is also the original realization version for PCA.

1.1.2 Limitations

The algorithm cost lots of time especially the progress of computing $\mathbf{X}\mathbf{X}^T$The complexity of PCA in eigen decomposition method is computed as below:

Covariance matrix computation is $O(p^2n)$, and its eigen-value decomposition is $O(p^3)$. where n represents the number of data points, p represented the features of each data points. Therefore, the complexity of PCA is $O(p^2n + p^3)$.

1.2 Singular Value Decomposition (SVD)

1. Compute the covariance matrix of $\mathbf{X}$, formally, compute $\frac{1}{m}\mathbf{X}\mathbf{X}^T$.

2. We do singular value decomposition to $\mathbf{X}$, formally,

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{5}$$

   where $\mathbf{U}$ is called left singular vectors, $\mathbf{V}$ is called right singular vectors, and $\mathbf{D}$ is a matrix full of zeros except the diagonals. $\mathbf{U}$ and $\mathbf{V}$ are both orthonormal basis, which means $\mathbf{U}^T\mathbf{U} = 1$,$\mathbf{V}^T\mathbf{V} = 1$

3. Therefore we have

$$\begin{aligned}\mathbf{X}\mathbf{X}^T &= \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)^T \\ &= \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{U}^T\end{aligned} \tag{6}$$

   And we find that $\mathbf{U}$ is just the eigen vector in Eigen Decomposition. $\mathbf{\Sigma}\mathbf{\Sigma}^T$ is just the matrix full of zeros except the diagonals, and the diagonals are the sorted eigen values $\{\lambda\}$ in Eigen Decomposition. Formally,

$$\mathbf{\Sigma}\mathbf{\Sigma}^T = \begin{bmatrix} \lambda_1 & & & & & & \\ & \lambda_2 & & & & & \\ & & \ddots & & & & \\ & & & \ddots & & & \\ & & & & \ddots & & \\ & & & & & \lambda_{n-1} & \\ & & & & & & \lambda_n \end{bmatrix} \tag{7}$$

   where $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_{n-1} \geq \lambda_n$.

4. Thus we without the eigen decomposition, get the eigen vectors and eigen values by SVD. The left singular vectors $\mathbf{U}$ is just the eigen vectors. Compute the first principal component $\mathbf{w} = \left(\mathbf{U}^T\mathbf{x}_i\right)\mathbf{U}$

1.2.1 Advantages

Much faster than eigen decomposition, we don't need to compute the covariance matrix. In addition, when the problem becomes reducing dimension as specific value instead of 1, SVD don't need a loop to get the highest λ. The matrix Σ tells us all.

The time complexity of SVD is $O(npk)$. where we want the first k eigen value, and n is the number of data points, p is the dimension (features) of each data point.

1.2.2 Limitations

Though it is much faster than eigen decomposition. SVD still faces the problem that time cost much. And When running PCA in SVD, the memory (space complexity) is also bottleneck.

2 Factor Analysis (FA)

Calculate the Bayesian posterior $p(y|x)$ of the Factor Analysis model $\mathrm{x} = \mathrm{Ay} + \mu + \mathrm{e}$*, with* $p(\mathrm{x}|\mathrm{y}) = G\left(\mathrm{x}|\mathrm{Ay} + \mu, \Sigma_e\right)$,$p(\mathbf{y}) = G\left(\mathbf{y}|0, \Sigma_y\right)$,*where* $G(\mathrm{z}|\mu, \Sigma)$ *denotes Gaussian distribution density with mean* μ *and covariance matrix* Σ.

Solution

$$\begin{aligned}E(\mathrm{x}) &= E(\mathrm{Ay} + \mu + \mathrm{e}) \\ &= E(\mathrm{Ay}) + \mu + E(\mathrm{e}) \\ &= \mu\end{aligned} \tag{8}$$

$$
\begin{aligned}
cov(x) = \Sigma_{xx} &= \mathrm{E}\left[(x - \mathrm{E}[x])(x - \mathrm{E}[x])^T\right] \\
&= \mathrm{E}\left[(\mu + \mathrm{A}z + \mathrm{e} - \mu)(\mu + \mathrm{A}z + \mathrm{e} - \mu)^T\right] \\
&= \mathrm{E}\left[\mathrm{A}zz^T\mathrm{A}^T + \mathrm{e}z^T\mathrm{A}^T + \mathrm{A}z\mathrm{e}^T + \mathrm{e}\mathrm{e}^T\right] \\
&= \mathrm{AE}\left[zz^T\right]\mathrm{A}^T + \mathrm{E}\left[\mathrm{e}\mathrm{e}^T\right] \\
&= \mathrm{A}\mathrm{A}^T + \Sigma_e
\end{aligned} \tag{9}
$$

so that $\mathrm{x} \sim \mathcal{N}(\mu, \mathrm{A}\mathrm{A}^T + \Sigma_e)$, $p(\mathrm{x}) = G(\mathrm{x}|\mu, \mathrm{A}\mathrm{A}^T + \Sigma_e)$

In *Gaussian processes*, the author *Chuong B. Do* gives a formula to compute The mean and variance of the conditional distribution.

$$
\mathrm{x}_1|\mathrm{x}_2 \sim \mathrm{N}\left(\mu_{1|2}, \Sigma_{1|2}\right)
$$

where we have

$$
\begin{aligned}
\mu_{1|2} &= \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2) \\
\Sigma_{1|2} &= \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}
\end{aligned} \tag{10}
$$

According to the formula we can compute that

$$
\begin{aligned}
\mu_{y|x} &= \mathrm{A}^{\mathrm{T}}\left(AA^T + \Sigma_e\right)^{-1}\mathrm{x} \\
\Sigma_{y|x} &= \Sigma_y - \mathrm{A}^{\mathrm{T}}\left(\mathrm{A}\mathrm{A}^{\mathrm{T}} + \Sigma_e\right)^{-1}\mathrm{A}
\end{aligned} \tag{11}
$$

$$
\begin{aligned}
p(\mathrm{y}|\mathrm{x}) &= G(\mathrm{y}|\mu_{y|x}, \Sigma_{y|x}) \\
&= G\left(\mathrm{y}|\mathrm{A}^{\mathrm{T}}\left(AA^T + \Sigma_e\right)^{-1}\mathrm{x}, \Sigma_y - \mathrm{A}^{\mathrm{T}}\left(\mathrm{A}\mathrm{A}^{\mathrm{T}} + \Sigma_e\right)^{-1}\mathrm{A}\right)
\end{aligned}
$$

3 Independent Component Analysis (ICA)

Explain why maximizing non-Gaussianity could be used as a principle for ICA estimation.

Solution

- The Central Limit Theorem, a classical result in probability theory, tells that the distribution of a sum of independent random variables tends toward a Gaussian distribution, under certain conditions.
- Thus, a sum of two independent random variables usually has a distribution that is closer to Gaussian than any of the two original random variables.
- Formally, we have

  $$
  y = \mathbf{w}^T\mathbf{x} = \mathbf{w}^T A\mathbf{s} = \left(\mathbf{w}^T A\right)\mathbf{s} = \mathbf{z}^T\mathbf{s} \tag{12}
  $$

  where $\mathbf{z}^T\mathbf{s}$ is more Gaussian than any of the s_i, and becomes least Gaussian when it infact equals one of the s_i. (Note that this is strictly true only if the s_i have identical distributions, as we assumed here.) distributions, as we assumed here.)
- We do not in practice know the values of $\mathbf{z}$, but we do not need to, because $\mathbf{z}^T\mathbf{s} = \mathbf{w}^T\mathbf{x}$ by the definition of $\mathbf{w}$. We can just let $\mathbf{w}$ vary and look at the distribution of $\mathbf{w}^T\mathbf{x}$.
- Therefore,we could take as $\mathbf{w}$ as a vector that *maximizes the nongaussianity* of $\mathbf{w}^T\mathbf{x}$. Such a vector would necessarily correspond to a $\mathbf{z} = \mathbf{A}^T\mathbf{w}$, which has only one nonzero component. This means that $y = \mathbf{w}^T\mathbf{x} = \mathbf{z}^T\mathbf{s}$ equals one of the independent components! Maximizing the nongaussianity of $\mathbf{w}^T\mathbf{x}$ thus gives us one of the independent components.

Table 1: The random variables and their ranges.

Variable	Description	Range
n_samples	the number of samples	$(0, 1000]$
m	the dimension of x	$[2, 10]$
n	the dimension of y	$[m+1, 20]$
noise_level	the variance of noise σ^2	$[0, 0.3]$
mu	mean of x	$[0, 20]$
A	the transform matrix	$\{a_{ij}\} \sim U(0, 1)$

4 Dimensionality Reduction by FA

Solution

4.1 Dataset Generation

4.2 Experiments

The experiments includes two parts.

- The first part, I generate lots of datasets with **all** setting values randomly set as table 1, then I compute the accuracy of AIC and BIC predicting the correct(optimal) m (the number of components). By this part, we can explore which method is better and how better the method is when the dataset is totally unknown.
- The second part, I use the **control variable method**. First I control one variable to linearly increase in its range, and the rest of variables are randomly set as table 1. I will in turn control each variable. By this part, we can explore the influence degree of each factor (variable) and compare among these factors. In this way, we can choose the correct information criterion when faced with a dataset with known factors.

4.2.1 Precision Test

I generate 1000 dataset as table 1, the final accuracy is shown as table 2.

Information Criterion	Accuracy
AIC	0.786
BIC	**0.856**

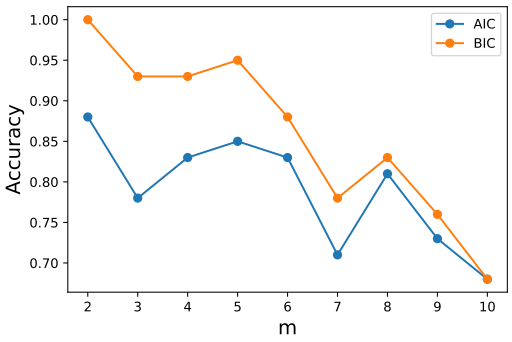
Table 2: The accuracy of getting the optimal number of component m of AIC and BIC, which is computed by generating 1000 random dataset

From the table result, we can get the conclusion that, for choosing model parameters in Factor Analysis model, BIC performs much better than AIC, under the condition that the dataset is totally random.

4.2.2 Part 2

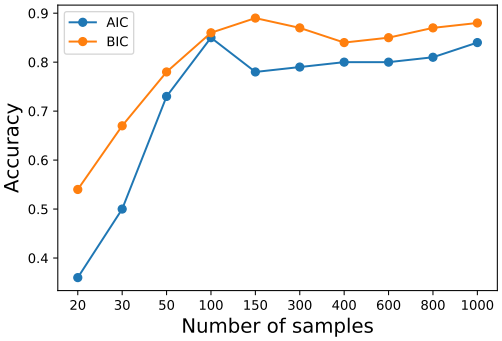
Control dimension of latent factors

m	AIC	BIC	m	AIC	BIC
2	0.88	1.00	7	0.71	0.78
3	0.78	0.93	8	0.81	0.83
4	0.83	0.93	9	0.73	0.76
5	0.85	0.95	10	0.68	0.68
6	0.83	0.88			



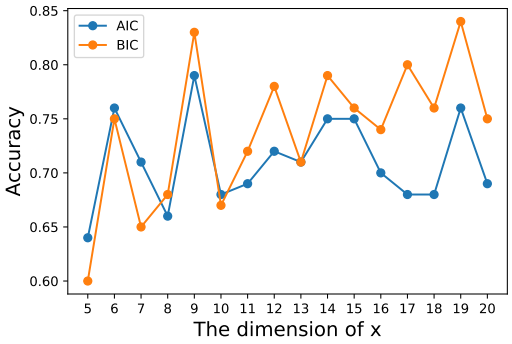
Control number of samples x

N	AIC	BIC	N	AIC	BIC
20	0.36	0.54	300	0.79	0.87
30	0.50	0.67	400	0.80	0.84
50	0.73	0.78	600	0.80	0.85
100	0.85	0.86	800	0.81	0.87
150	0.78	0.89	1000	0.84	0.88



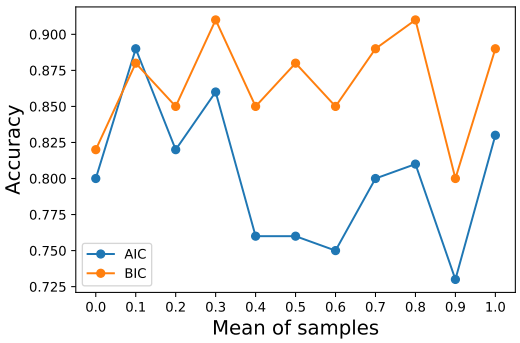
Control dimension of samples x

n	AIC	BIC	n	AIC	BIC
5	0.64	0.60	13	0.71	0.71
6	0.76	0.75	14	0.75	0.79
7	0.71	0.65	15	0.75	0.76
8	0.66	0.68	16	0.70	0.74
9	0.79	0.83	17	0.68	0.80
10	0.68	0.67	18	0.68	0.76
11	0.69	0.72	19	0.76	0.84
12	0.72	0.78	20	0.69	0.75



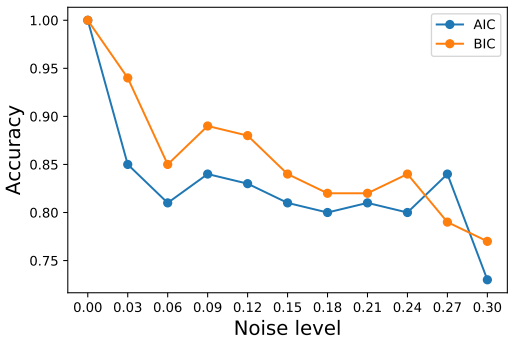
Control mean of samples x

μ	AIC	BIC	μ	AIC	BIC
0	0.80	0.82	0.6	0.75	0.85
0.1	0.89	0.88	0.7	0.80	0.89
0.2	0.82	0.85	0.8	0.81	0.91
0.3	0.86	0.91	0.9	0.73	0.80
0.4	0.76	0.85	1.0	0.83	0.89
0.5	0.76	0.88			



Control noise level (variance) σ^2

σ^2	AIC	BIC	σ^2	AIC	BIC
0	1.00	1.00	0.18	0.80	0.82
0.03	0.85	0.94	0.21	0.81	0.82
0.06	0.81	0.85	0.24	0.80	0.84
0.09	0.84	0.89	0.27	0.84	0.79
0.12	0.83	0.88	0.30	0.73	0.77
0.15	0.81	0.84			



4.3 Comparison summary

1. I analyze five factors. And under almost all circumstances, BIC gets the more accurate result than AIC.

2. When m, the dimension of latent factors y increases, the accuracy of AIC and BIC both decrease. BIC always keeps higher accuracy in the dimension range $2 \sim 10$. However, we can find that the gap between AIC and BIC is smaller and smaller when m increases.

3. When the number of samples are small to 20, the accuracy of both AIC and BIC can be very low. This is normal because FA model cannot fit well (under-fitting). When the number of samples increases gradually, the accuracy of AIC and BIC both ascend and then become steady. In addition, under all circumstances of N, the accuracy of BIC is a little bit higher than AIC's, with the similar trend.

4. Things got a little more complicated in controlling n, dimension of samples. We can see that when n is small, AIC performs better than BIC, however, when n increases to larger than 10, BIC starts to perform better than AIC, and, the gap between BIC and AIC is bigger and bigger. Besides the curves are not steady at all.

5. The curves are the most steady when controlling μ, mean of samples among these five factors. We can see that the accuracy of AIC is up and down in 0.8 while the accuracy of BIC is up and down in 0.85. In a overview, BIC performs still much better than AIC, though AIC performs a little bit better than BIC when $\mu = 0.1$.

6. The final factor is the noise level [2]. As I guess, the accuracy of both information criterion is very sensitive to the noise level. When noise level is small or even zero, the accuracy of AIC and BIC is pretty high (accuracy of both is 100% when $\sigma^2 = 0$). When σ^2 becomes larger, both information decreases gradually. Although under some special circumstances AIC may have higher accuracy, BIC performs still a liitle bit better than AIC.

4.4 Strategy of choosing information criterion

From section 4.2.1 and section 4.3, I sum up the strategy of choosing information criterion (AIC or BIC) as follows.

- Gnerally speaking, BIC performs better than AIC. If we don't know well the dataset or we want a guaranteed good result, BIC will be better.

- When n, the dimension of original dataset, is small, and when m, the dimension of latent factors is large, and when the noise level is pretty high. AIC method may have a better result.

- Other than the circumstances above, BIC is highly recommended.

5 Spectral clustering

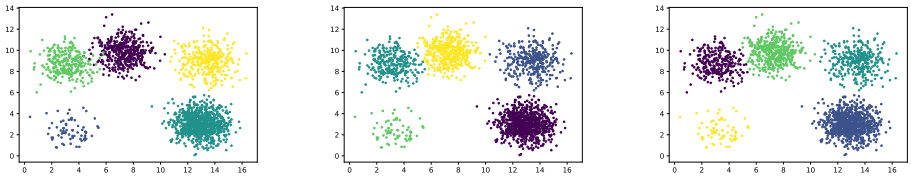
In this section I experimented with the shapes of datasets.

I made some special famous datasets to test the Spectral clustering. And I find that if I adjust parameters well enough, the performance can be pretty great. For comparison, I will also give the label of one dataset and the performance of another method to cluster like kmeans, DBSCAN.

5.1 When does SC work well

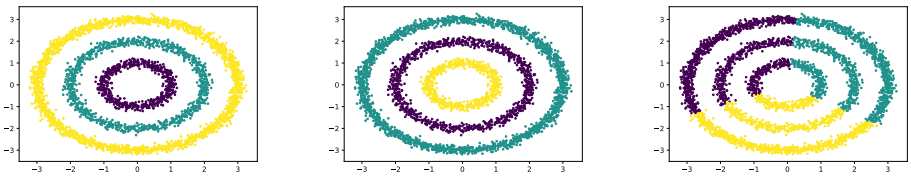
Spectral Clustering can adjust its parameter: affinity matrix to adapt almost all shapes of SC. Therefore, SC can beat Kmeans on some density-based datasets, which is more suitable for DBSCAN, a method of determining the result using the data density and radius of the circle.

Here I will list some dataset to show the comparison.



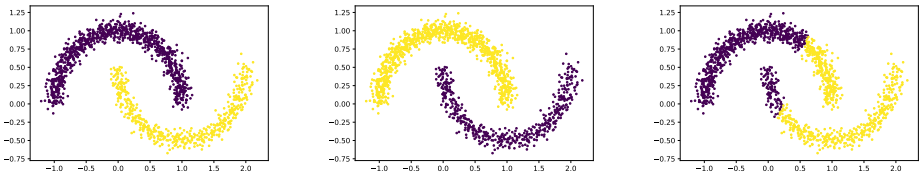
(a) the Dataset, colored by label (b) the result of Spectral Cluster, affinity=rbf (c) the result of Kmeans

Figure 1: The most common condition of five clusters, and we can see that Spectral clustering performs very well as kmeans, a traditional and canonical method to solve this kind of dataset



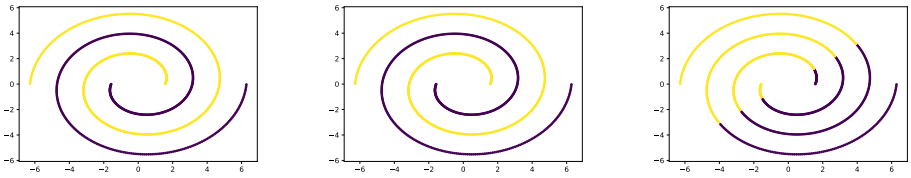
(a) the Dataset, colored by **label** (b) the result of Spectral Cluster,affinity=nearest_neighbors (c) the result of Kmeans

Figure 2: three concentric circles dataset, and we can see kmeans cannot do with this dataset, DBSCAN can handle the dataset actually. However, when I set the parameter affinity of SC as nearest neighbors. Spectral clustering still work as well as DBSCAN.



(a) the Dataset, colored by **label** (b) the result of Spectral Cluster,affinity=nearest_neighbors (c) the result of Kmeans

Figure 3: Two moon dataset, and we can see that like kmeans, this dataset is not suitable for kmeans, and from sklearn I got to know that many methods such as meanshift and Gaussian Mixture Model all cannot solve this problem. However, SC works very well in this dataset, only if we set the parameters appropriately.

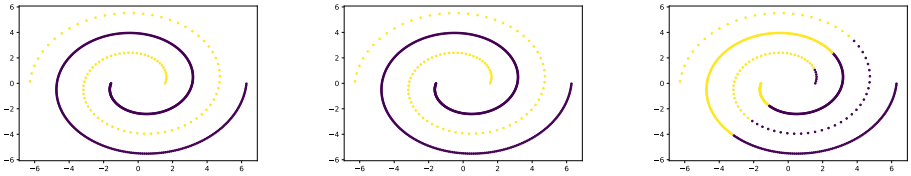


(a) the Dataset, colored by **label**

(b) the result of Spectral Cluster,affinity=nearest_neighbors

(c) the result of Kmeans

Figure 4: Spiral dataset, a very interesting dataset. It is the hard version of the two moon dataset. Many methods cannot handle this dataset like kmeans, GMM, and Birch, Spectral clustering works very well.



(a) the Dataset, colored by **label**

(b) the result of Spectral Cluster,affinity=nearest_neighbors

(c) the result of Kmeans

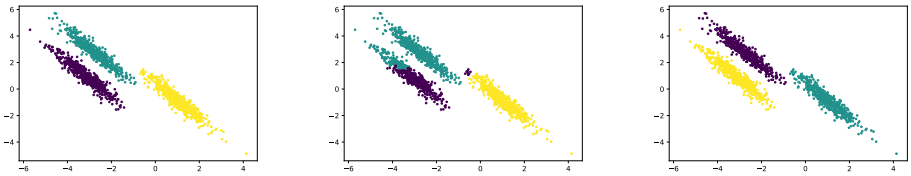
Figure 5: Un balanced spiral dataset, note that the dataset is different from the previous dataset, the density of yellow points is smaller than blue one. Actually only if I set the number of neighbors as 3, the result is satisfactory, instead of a very large range on other datasets. Therefore, the unbalanced dataset enlarge the parameters sensitivity of Spectral clustering.

5.2 When does SC fail?

Actually the condition is pretty infrequent. I found two conditions , which I name them as "Near GMM" and "Unbalanced Scale".

5.2.1 Near GMM

Spectral cannot handle the data whose distribution is Gaussian Mixture very well as GMM, especially the mixtures are pretty near. Here I will list some situations to show.



(a) the Dataset, colored by **label**

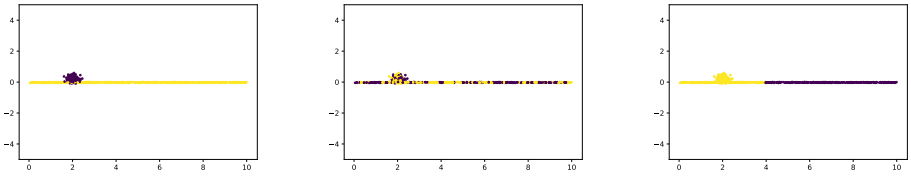
(b) the result of Spectral Cluster,affinity=nearest_neighbors

(c) the result of GMM

Figure 6: In this dataset generated by GMM. However we adjust the parameters of SC, we cannot handle very well. we can see that in the figure there are two mixtures very near, which makes SC perform not well.

5.2.2 Unbalanced Scale

Now let's see what causes this reason. Let me enlarge the scale of axis y, and the figure is as below 8.

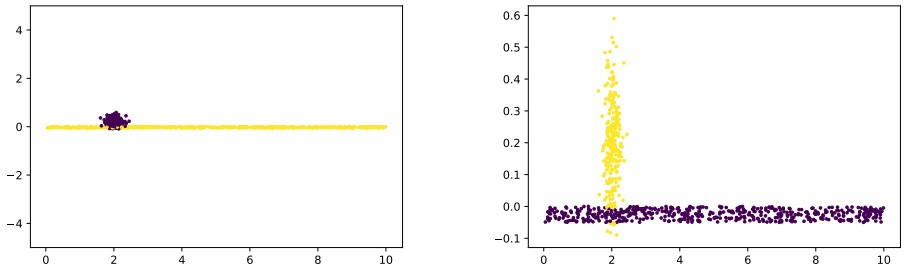


(a) the result of GMM

(b) the result of Spectral Cluster,affinity=nearest_neighbors

(c) the result of Spectral Cluster,affinity=rbf

Figure 7: In this dataset, the scale of axis x and axis y is very different. and the dataset is also unbalanced. GMM can handle this well. However, whether setting affinity as nearest_neighbors or rbf, the result is not satisfactory neither,



(a) Normal view

(b) Enlarged view of axis y

Figure 8: Note that these two sub-figures are the same actually, with the right figure the scale of y enlarged. we can see that this is actually a GMM, though from the scale of axis x, we think the dataset colored by yellow is just like circle distribution.

5.3 Summary

5.3.1 When does Spectral clustering work well

- From the experiments we can see that Spectral clustering can Work for most shapes include normal balls, spirals, moons. Compared with other traditional clustering method such as kmeans, DBSCAN, the generalization of Spectral clustering is much higher.
- The reason of great generalization of Spectral clustering is that the affinity is self-defined. Therefore, we can use many kinds of kernels. When affinity is rbf, the effect is like kmeans. When affinity is nearest_neighbors, the effect is like DBSCAN. Meanwhile, the distance metric can be also defined. So that we can choose a better metric like cosine similarity and mahalanobis distance.

5.3.2 When does Spectral clustering work well

- **Dataset in GMM** Although Spectral clustering can handle a few dataset generated by GMM, it performs not well enough as GMM. As we have experimented, when two mixtures in GMM is near, SC performs badly. Maybe one may think near mixtures is not common. However, the experiments is in two dimension, in many dimensions and many components, the probability of the condition is pretty common.
- **Different scale in different dimension** As we have experimented, when we are faced with a dataset whose dimension is not decentralized (or rather, the importance of different dimension is not the same). Maybe we can choose a better distance metric like mahalanobis distance to overcome the distinction of different dimension.
- **High dimension, many components dataset** The two conditions above are very common in High dimension dataset. The scale is always unbalanced in high dimension dataset.

Meanwhile, Spectral clustering is not very fit for a dataset that has many components. Many components make the condition of near components easier.