CS420 Homework 2

CS420-Machine Learning , Shikui Tu, Spring 2019.

Name: Jiayi Xu   Student ID: 516021910396   Email: xujiayi925@sjtu.edu.cn

1 PCA Algorithm

In this section, I will give 2 algorithms to calculate the first principal component and discuss the advantages and limitations of them.

1.1 Eigen decomposition

Suppose the projection matrix is $\boldsymbol{V}$, then each data point $\boldsymbol{x_i}$ is projected to $\boldsymbol{V}^T\boldsymbol{x_i}$ in the projection space. Since we want the data set $\boldsymbol{X} = \{\boldsymbol{x_1}, \cdots, \boldsymbol{x_N}\}$ to be as separated as possible, we should make sure that the variance of the projected data points are the largest. The variance of the projected data points is:

$$\frac{1}{N-1}\sum_{i=1}^{N}(\boldsymbol{x}_i^T\boldsymbol{V})^2 = \frac{1}{N-1}\sum_{i=1}^{N}\boldsymbol{V}^T\boldsymbol{x}_i\boldsymbol{x}_i^T\boldsymbol{V} = \boldsymbol{V}^T(\frac{1}{N-1}\sum_{i=1}^{N}\boldsymbol{x}_i\boldsymbol{x}_i^T)\boldsymbol{V}$$

It's known that $\frac{1}{N-1}\sum_{i=1}^{N}\boldsymbol{x}_i\boldsymbol{x}_i^T$ is the covariance matrix, we can represent it as Σ.

We denote the variance of the projected data points as $\boldsymbol{\lambda}$, then:

$$\boldsymbol{\lambda} = \boldsymbol{V}^T\Sigma\boldsymbol{V}$$

Since $\boldsymbol{V}$ is composed of standard orthogonal basis, therefore, $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}$, then:

$$\boldsymbol{V}\boldsymbol{\lambda} = \boldsymbol{\lambda}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{V}^T\Sigma\boldsymbol{W} = \Sigma\boldsymbol{V}$$

that is:

$$\Sigma\boldsymbol{V} = \boldsymbol{\lambda}\boldsymbol{V}$$

Therefore, $\boldsymbol{\lambda}$ is composed of the eigenvalues of Σ, and $\boldsymbol{V}$ is composed of the eigenvectors. The first principal component $\boldsymbol{w}$ is calculated using the eigenvector with largest eigenvalue. The computational details can be seen as Alg. 1.

Algorithm 1: Eigen decomposition for getting the first principal component

```
  Input  : Data points $\boldsymbol{X} = \boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_N}$;
  Output: The first principal component $\boldsymbol{w}$;
1 Normalize the data set $\boldsymbol{X}$ (that is $\sum_i \boldsymbol{x}_i = \boldsymbol{0}$);
2 Calculate the covariance matrix of $\boldsymbol{X}$, which is denoted as $\Sigma$:
      $\Sigma = \frac{1}{N-1}\sum_{i=1}^{N}\boldsymbol{x}_i\boldsymbol{x}_i^T;$
3 Calculate the eigenvalues $\lambda$ and corresponding eigenvectors $\boldsymbol{V}$;
4 Choose the largest eigenvalue $\lambda_{max}$, suppose its corresponding eigenvector is $\boldsymbol{v}_{max}$;
5 Calculate the first principal component $\boldsymbol{w}$:
      $\boldsymbol{w} = \boldsymbol{v}_{max}^T\boldsymbol{X}.$
```

Advantages:

- PCA using eigen decomposition has simple logic and is easy to understand and implement.
- This algorithm is derived from the theory of 'maximum separability', therefore it is reasonable.

Limitations:

- PCA using eigen decomposition needs to calculate the covariance matrix Σ, so it is time consuming when encountering large amount of data.
- This algorithm requires a large amount of computation when calculating the eigenvalues and eigen vectors of the covariance matrix.

1.2 Singular value decomposition

According to singular value decomposition (SVD), if the dataset $\boldsymbol{X}$ can be decomposed to $\boldsymbol{UDV}^T$, that is:

$$\boldsymbol{X} = \boldsymbol{UDV}^T,$$

then, since $\boldsymbol{U}^T \cdot \boldsymbol{U} = \boldsymbol{I}$. $\boldsymbol{V}^T \cdot \boldsymbol{V} = \boldsymbol{I}$ and $\boldsymbol{D}$ is a diagonal matrix:

$$\boldsymbol{X}^T\boldsymbol{X} = (\boldsymbol{UDV}^T)^T \cdot \boldsymbol{UDV}^T = \boldsymbol{VD}^T\boldsymbol{U}^T \cdot \boldsymbol{UDV}^T = \boldsymbol{VD}^T \cdot \boldsymbol{DV}^T = \boldsymbol{VD}^2\boldsymbol{V}^T,$$

then:

$$\boldsymbol{X}^T\boldsymbol{X} \cdot \boldsymbol{V} = \boldsymbol{VD}^2\boldsymbol{V}^T \cdot \boldsymbol{V} = \boldsymbol{VD}^2 = \boldsymbol{D}^2\boldsymbol{V}$$

Therefore, we can see that $\boldsymbol{V}$ is composed of the eigenvectors of Σ, and $\boldsymbol{D}^2$ is composed of the eigenvalues of Σ. The first principal component $\boldsymbol{w}$ is calculated using the eigenvector with largest eigenvalue. The computational details can be seen as Alg. 2.

Algorithm 2: Eigen decomposition for getting the first principal component

```
Input  : Data points $\boldsymbol{X} = \boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_N}$;
Output: The first principal component $\boldsymbol{w}$;
1 Normalize the data set $\boldsymbol{X}$ (that is $\sum_i \boldsymbol{x}_i = \boldsymbol{0}$);
2 Conducting SVD on the data set $\boldsymbol{X}$:
      $\boldsymbol{X} = \boldsymbol{UDV}^T$
3 The eigenvalues is $\boldsymbol{D}^2$, and the corresponding eigenvectors is $\boldsymbol{V}$;
4 Choose the largest eigenvalue $\lambda_{max}$, suppose its corresponding eigenvector is $\boldsymbol{v}_{max}$;
5 Calculate the first principal component $\boldsymbol{w}$:
      $\boldsymbol{w} = \boldsymbol{v}_{max}^T\boldsymbol{X}$.
```

Advantages:

- PCA using SVD does not need to calculate the covariance matrix Σ, which requires a large amount of computation.
- PCA only needs to calculate the eigenvalues and eigenvectors of $\boldsymbol{X}$ rather than $\boldsymbol{X}^T\boldsymbol{X}$, which will save computation.

Limitations:

- The same as all PCA algorithms, PCA using SVD neglects the label information, which contains important separability information.
- The same as all PCA algorithms, the principal components have no direct explainability.

2 Factor Analysis

According to problem formulation, $p(x|y) = G(x|\boldsymbol{A}y + \mu, \Sigma_e)$, $p(y) = G(y|0, \Sigma_y)$, $E(e) = 0$, then:

$$E(x) = \mu,$$

$$\begin{aligned}\Sigma(x) &= E[(x - E(x))(x - E(x))] \\ &= E[(\mu - \boldsymbol{A}y + e - \mu)(\mu - \boldsymbol{A}y + e - \mu)] \\ &= E[\boldsymbol{A}yy^T\boldsymbol{A}^T + ey^T\boldsymbol{A}^T + \boldsymbol{A}ye^T + ee^T] \\ &= \boldsymbol{A}E(yy^T)\boldsymbol{A}^T + E(ee^T) \\ &= \boldsymbol{A}\Sigma_y\boldsymbol{A}^T + \Sigma_e,\end{aligned}$$

That is, $p(x) = G(x|\mu, \boldsymbol{A}\Sigma_y\boldsymbol{A}^T + \Sigma_e)$.

Therefore, according to Bayesian fomula, the Bayesian posterior can be calculated as follows:

$$\begin{aligned}p(y|x) &= \frac{p(x|y)p(y)}{p(x)} \\ &= \frac{G(x|\boldsymbol{A}y + \mu, \Sigma_e)G(y|0, \Sigma_y)}{G(x|\mu, \boldsymbol{A}\Sigma_y\boldsymbol{A}^T + \Sigma_e)}\end{aligned}$$

3 Independent Component Analysis

Independent component analysis (ICA) is a computational method to separate a multivariate signal into additive subcomponents. ICA is done by assuming that the subcomponents are non-Gaussian signals and that they are statistically independent from each other. ICA is a special case of blind source separation.

Suppose the data are represented by the observed vectors $\boldsymbol{x} = (x_1, x_2, \cdots, x_m)^T$, and the hidden components are $\boldsymbol{s} = (s_1.s_2, \cdots, s_n)^T$. ICA is to use a linear static transformation $\boldsymbol{W}$ to transform the observed data $\boldsymbol{x}$ into hidden components $\boldsymbol{s}$, $\boldsymbol{s} = \boldsymbol{W}\boldsymbol{x}$, $s_i = w_i^T x_i$.

Suppose each $\boldsymbol{s}_i$ has probability density $p_s(s_i)$, then the joint distribution of $\boldsymbol{s}$ is:

$$p(s) = \prod_{i=1}^{n} p_s(s_i)$$

According to the assumption that $\boldsymbol{s}$ are independent of each other, $p(\boldsymbol{x})$ can be calculated as:

$$p(\boldsymbol{x}) = p_s(\boldsymbol{W}\boldsymbol{x})|\boldsymbol{W}| = |\boldsymbol{W}|\prod_{i=1}^{n} p_s(w_i^T x_i)$$

According to probability theory, the probability density function $p(x)$ is the derivative of cumulative distribution function (CDF) $F(x)$. The CDF function $F(x)$ should satisfy the following two conditions: monotone increasing and ranging from 0 to 1. Therefore, sigmoid function is suitable for describing the CDF function. If we do not know the distribution of $\boldsymbol{s}$, we assume that the CDF function of $\boldsymbol{s}$ conforms to sigmoid function:

$$g(\boldsymbol{s}) = \frac{1}{1 + e^{-\boldsymbol{s}}},$$

after derivation:

$$P_s(\boldsymbol{s}) = g'(\boldsymbol{s}) = \frac{e^{\boldsymbol{s}}}{(1 + e^{\boldsymbol{s}})^2}$$

Then, we can estimate $\boldsymbol{W}$ by maximizing logarithmic likelihood:

$$\mathcal{L}(\boldsymbol{W}) = \sum_{i=1}^{m}(\sum_{j=1}^{n} log\ g'(w_j^T x_i) + log|\boldsymbol{W}|)$$

However, the hidden component $\boldsymbol{s}$ cannot agree with Gaussian distribution. This is due to the uncertainty of ICA.

Suppose the hidden components agree with Gaussian distribution, that is $\boldsymbol{s} \sim N(0, \boldsymbol{I})$. Then the observed components $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}$ $(\boldsymbol{A}^{-1} = \boldsymbol{W})$ agree with Gaussian distribution as well, $E(\boldsymbol{x}) = 0$, $Cov(\boldsymbol{x}) = E[\boldsymbol{x}\boldsymbol{x}^T] = E[\boldsymbol{A}\boldsymbol{s}\boldsymbol{s}^T\boldsymbol{A}^T] = \boldsymbol{A}\boldsymbol{A}^T$.

Suppose there is a orthogonal matrix $\boldsymbol{R}$, $\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I}$, we can get $\boldsymbol{A}' = \boldsymbol{A}\boldsymbol{R}$. If we replace $\boldsymbol{A}$ with $\boldsymbol{A}'$, then $\boldsymbol{x}' = \boldsymbol{A}'\boldsymbol{s}$. In this case, $E(\boldsymbol{x}') = 0$, $Cov(\boldsymbol{x}') = E[\boldsymbol{x}'(\boldsymbol{x}'^T)] = E[\boldsymbol{A}'\boldsymbol{s}\boldsymbol{s}^T(\boldsymbol{A}')^T] = E[\boldsymbol{A}\boldsymbol{R}\boldsymbol{s}\boldsymbol{s}^T(AR)^T] = \boldsymbol{A}\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{A}^T = \boldsymbol{A}\boldsymbol{A}^T$.

From above, we can see that whether the mixture matrix is $\boldsymbol{A}$ or $\boldsymbol{A}'$, the distribution of observed components $\boldsymbol{x}$ is the same, so the mixture matrix is not unique, we cannot obtain the mixture matrix $\boldsymbol{A} = \boldsymbol{W}^{-1}$ through ICA, therefore, we cannot determine the original hidden component. In conclusion, we cannot estimate with Gaussian distribution.

4 Dimension Reduction by FA

Dimension reduction by FA is done based on the assumption that the observed dataset can be generated though a latent dataset with a lower dimension. Then whether the estimated dimension of the latent dataset is accurate can be essential. This problem is to verify the accuracy of FA to estimate the dimension of latent dataset y in different settings.

4.1 Effect of sample sizes

To test the effect of sample sizes, I fix $n = 4$, $m = 3$, noise level $\sigma^2 = 0.1$, and vary the the sample sizes from 10 to 1000. The log-likelihood equals to the sum of log-likelihood of all samples, that is the average score of all samples multiply the number of samples. I set the Aic score equal to the log-likelihood minus the value of m, the Bic score equal to the log-likelihood minus the value of $m * \ln N/2$. The log-likelihood, aic score and bic score are shown as Fig. 1.

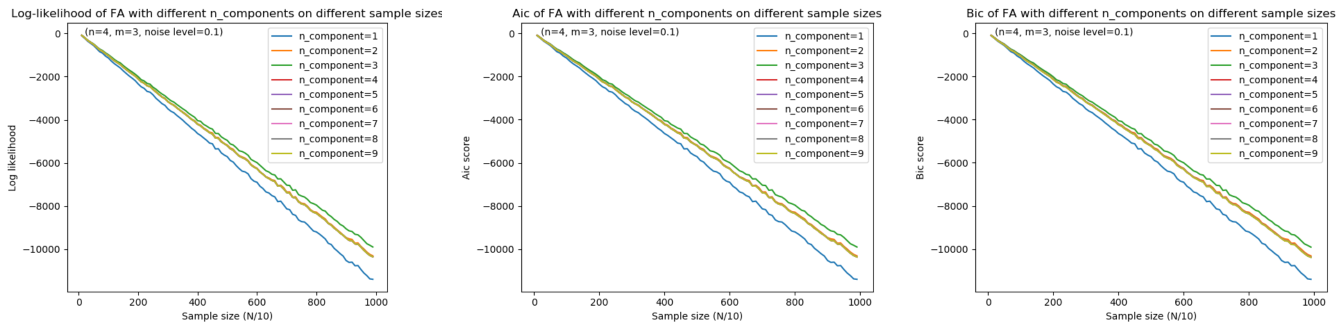


Figure 1: Effect of sample sizes on log-likelihood and Bic score

As we can see from the pictures, all of the three model selection methods, log-likelihood, Aic and Bic can select the proper m, $m = 3$. The Aic score and Bic score have no big difference to the log-likelihood, that is because the value of m is much smaller than the log-likelihood on magnitude. Therefore, the penalizing term has little effect. With sample sizes increasing, the log-likelihood, Aic score and Bic score are decreasing, that does not mean that FA performs worse with larger sample size, because the average log-likelihood, Aic score, Bic score are similar under different sample sizes.

4.2 Effect of the dimension of observed dataset n

To test the effect of dimension of the observed dataset n, I fix sample size $N = 1000$, $m = 3$, noise level $\sigma^2 = 0.1$, and vary the dimension of n from 1 to 20. The calculation of log-likelihood, Aic score and Bic score are the same as Sec. 4.1. The log-likelihood, aic score and bic score of different n are shown as Fig. 2.

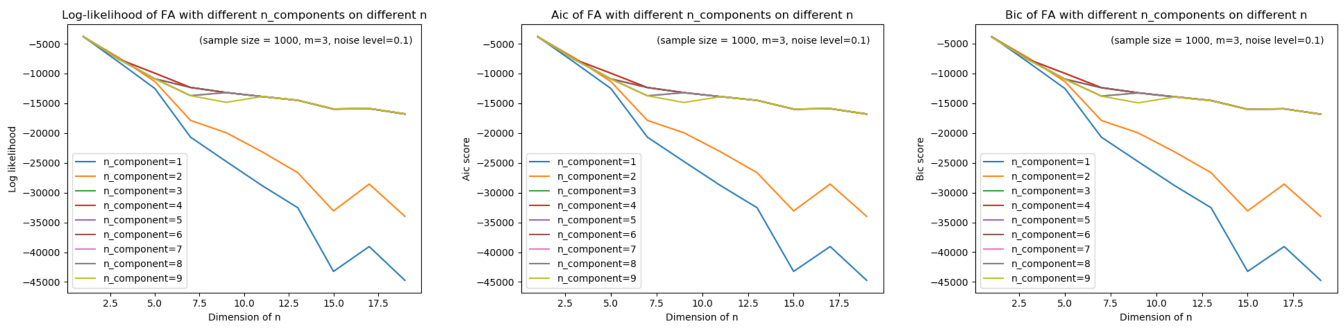


Figure 2: Effect of dimension n on Bic score

As we can see from the pictures, the dimension $m = 3$ has a high score no matter which model selection method is adopted.

Besides, we can notice that with higher dimension of n, the log-likelihood, aic score and bic score of n_component= 1 and n_component= 2 will all drop even if aic and bic prefer simpler model. Maybe that is because larger n will help reveal the fact that the observed dataset cannot be generated through latent dataset with only one or two dimensions.

Although it is hard to tell from the pictures, in all of these three model selection methods, Aic and Bic perform better than log-likelihood due to the penalization to m. In the model selection method using log-likelihood, FA cannot differentiate $m = 3$ from $m >= 4$. However, Aic and Bic will select the proper m with n which is big enough.

This group of experiments show that if the dimension of the observed dataset x is not bigger enough than the dimension of the latent dataset y, FA will not perform well in determining the dimension of latent dataset m, that is FA cannot determine the proper dimension to reduce the observed dataset to. Besides, with bigger n, both log-likelihood and Aic will prefer bigger m, but Bic will choose the proper m due to its penalization term.

The m that log-likelihood, Aic, and Bic choose is shown as Tab. 1. The correct m is 3, we can see that Aic and Bic have higher probability to select the proper $m = 3$.

Table 1: The m that log-likelihood, Aic, and Bic choose.

The actual m	3	3	3	3	3	3	3	3	3	3
The m log-likelihood chooses	1	7	3	6	8	9	9	9	9	9
The m Aic chooses	1	2	3	3	3	4	4	8	8	9
The m Bic chooses	1	2	3	3	3	3	4	3	4	3

4.3 Effect of the dimension of the latent dataset m

To test the effect of dimension of the latent dataset m, I fix sample size $N = 1000$, noise level $\sigma^2 = 0.1$, $n = 20$, and vary the dimension of m from 1 to 20. I set the calculation of log-likelihood, Aic score and Bic score the same as Sec. 4.1. The log-likelihood, aic score and bic score of different m are shown as Fig. 3.

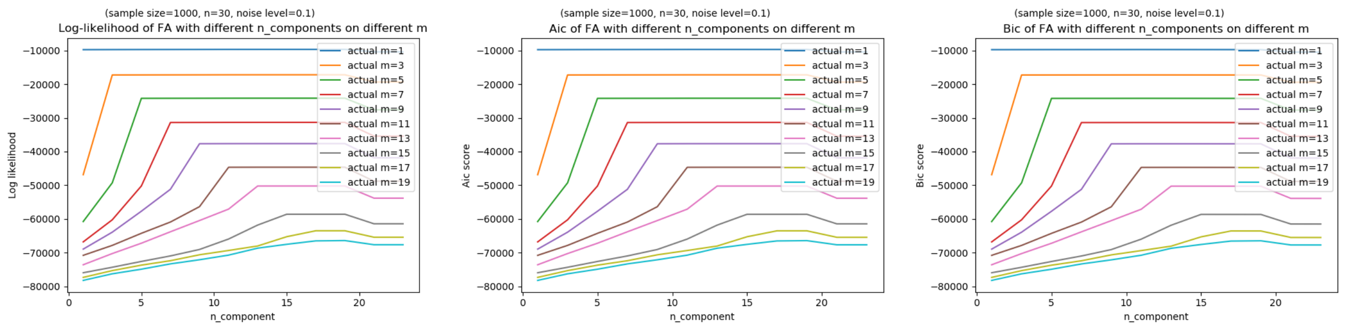


Figure 3: Effect of dimension m on different model selection methods. Each line represents the score of all n_components on the data generated on the same actual m.

The logic of this set of pictures is a little bit different from the previous ones. Each line represents the result score of different n_component (m to be tested) on the data generated by the same m. Therefore, the peak of the line is the m chosen by the model selection method.

The reason why I choose $n = 20$ is that the largest m is equal to 19, since Factor analysis is an algorithm to do dimension reduction, we should set $n > m$. The reason is quite intuitional as well because it is easier to reduce the dimension of data than raise the dimension of data.

As we can see from the pictures, when $n = 20$, all of the three model selection methods perform not bad. n_component $= 3$ all get high score.

In addition, we can find the result contains the shape of trapezoid. That means although the model selection methods assign the accurate n_component with high score, it assigns high score to others as well, thus it cannot differentiate them and choose the proper n_component. However, Aic and Bic penalize the model complexity, so it can better help FA select the proper n_component.

The m that log-likelihood, Aic and Bic choose is shown as Tab. 2.

Table 2: The m that log-likelihood, Aic, and Bic choose. ($n = 30$)

The actual m	1	3	5	7	9	11	13	15	17	19	error
The m log-likelihood chooses	19	17	19	19	19	19	19	19	19	19	88
The m Aic chooses	13	13	13	13	15	15	15	15	17	19	48
The m Bic chooses	11	11	11	11	11	13	13	15	17	19	32

The ‘error’ in the table is calculated according to the total absolute error between the actual m and the m selected by the model selection method.

From this table, we can see that, Aic and Bic perform better than log-likelihood, and Bic is better than Aic. Maybe that is because the penalization term of Bic considers the magnitude of sample size, thus the penalization term is more reasonable.

4.4 Effect of the noise level

To test the effect of noise level on the selection of m, I fix sample size $N = 1000$, $n = 10$, $m = 3$, and vary the noise level from 0 to 1. I set the calculation of log-likelihood, Aic score, and Bic score the same as Sec. 4.1. The log-likelihood, aic score and bic score of different n_components on different noise level are shown as Fig. 4.

As we can see from the pictures, with noise level increasing, the log-likelihood, aic score and bic score will decrease. Maybe that is because noise will make it harder for FA to estimate the proper n_component. Besides, we can see that Aic and Bic can select the proper n_component$= 3$ better than log-likelihood.

The m that log-likelihood, Aic and Bic choose is shown as Tab. 3.

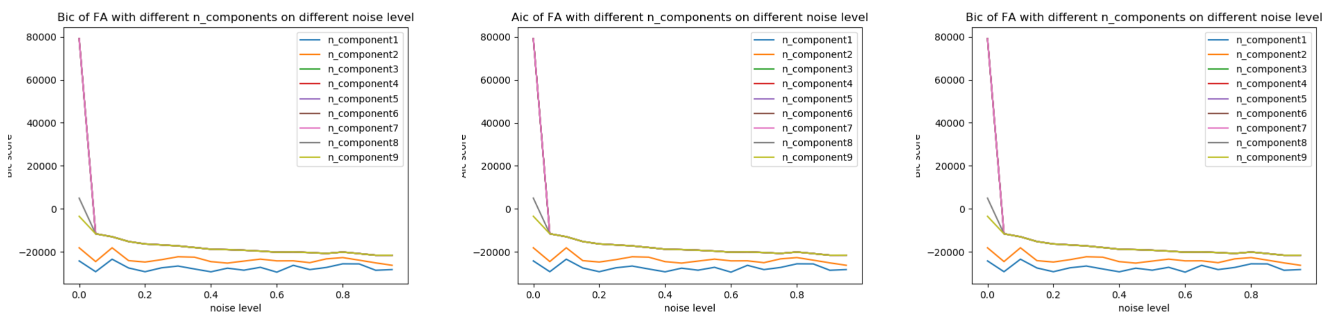


Figure 4: Effect of noise level

Table 3: The m that log-likelihood, Aic, and Bic choose. ($n = 20$)

The actual m	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	error
The m log-likelihood choose	6	8	9	9	7	9	9	8	9	9	9	9	9	9	9	8	9	9	9	8	111
The m Aic choose	6	6	6	6	6	5	6	6	5	5	6	5	5	5	5	6	6	6	5	6	52
The m Bic choose	6	4	4	5	4	4	3	3	4	4	5	4	3	4	5	4	3	3	3	4	31

The 'error' in the table is calculated according to the total absolute error between the actual m and the m selected by the model selection method, which is the same as Sec.4.3.

From this table, we can see that although Bic cannot always choose the right m, it has much smaller m selection error than log-likelihood and Aic, which proves that Bic is a better model selection method.

According to the above four sets of experiments, we can see that Bic performs the best all the time. That proves that when we should do model selection, using Bic is a good idea.

5 Spectral Clustering

To test the performance of spectral clustering, I use toy dataset, which contains 5 interesting 2D datasets. The first 4 datasets are generated by algorithms tuned to produce good clustering results, the last dataset contains homogeneous data, and there is no good clustering for it. I vary the parameters of spectral learning, such as 'eigen_solver' and 'assign_labels', the results are shown as Fig. 5.

We can see from the pictures that no matter the parameter setting, spectral clustering always get good clustering results. The GMM we use in homework 1 cannot achieve good clustering result in the first and second datasets, two circles and two moon, but spectral clustering can. Besides, spectral clustering performs well in the third, fourth and fifth dataset which GMM is good at. Therefore, we can tell spectral clustering is a better clustering method than GMM from this set of experiments.

Since the data used in toy dataset is 2D, it cannot reveal spectral clustering's performance on high dimensional data. Therefore, I generate some datasets by my own.

First, I generate datasets using Gaussian distribution. To make comparison easier, I fix cluster number to 3, data points in each cluster to 1000, and vary the dimension from 2 to 6. The results are shown as Fig.6. The first row is the ground truth clustering results, each color represents a cluster. The second row is the clustering result using spectral clustering, the title shows the clustering accuracy.

To evaluate the performance of spectral clustering, I come up with a criterion 'accuracy'. The accuracy of clustering cluster 1 is equal to the max number of data points originally in cluster 1 assigned to the same cluster by spectral clustering divide cluster 1's size. The accuracy of other

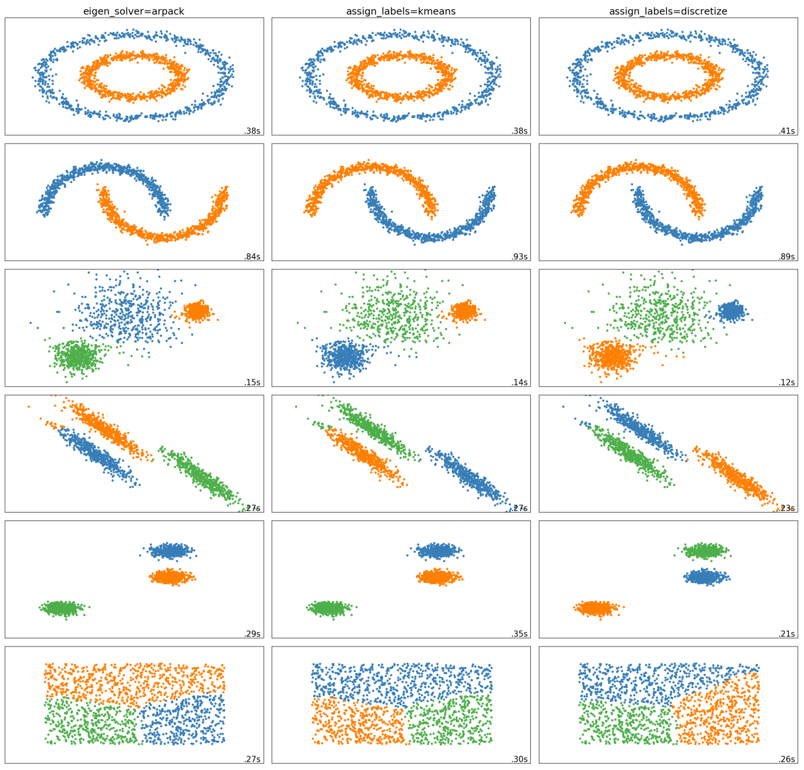
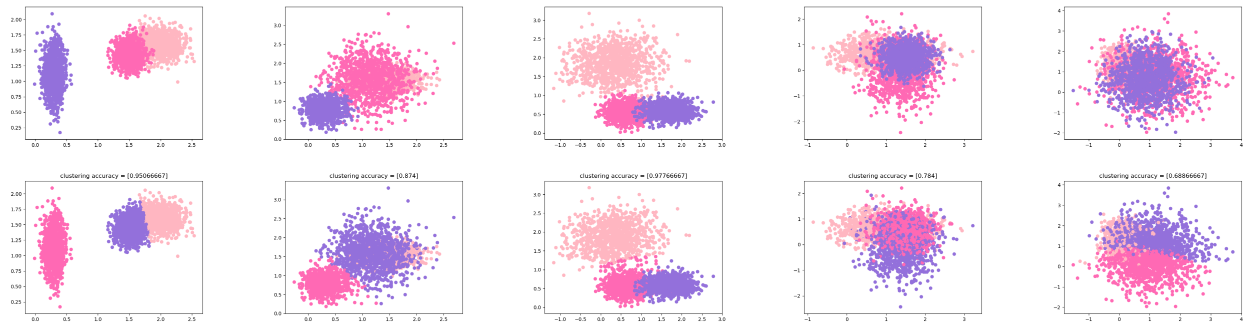


Figure 5: Spectral learning on toy datasets

clusters is the same as that of cluster 1. The total accuracy is the average of all clusters' accuracy.

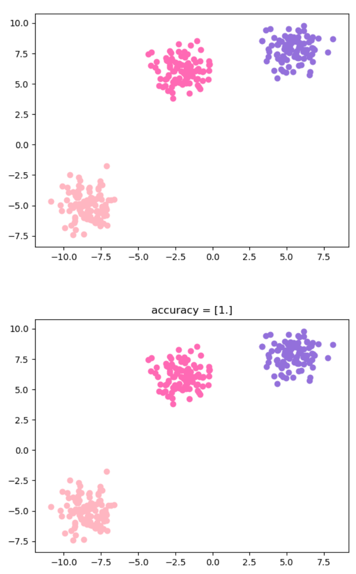


(a) dimension = 2 (b) dimension = 3 (c) dimension = 4 (d) dimension = 5 (e) dimension = 6

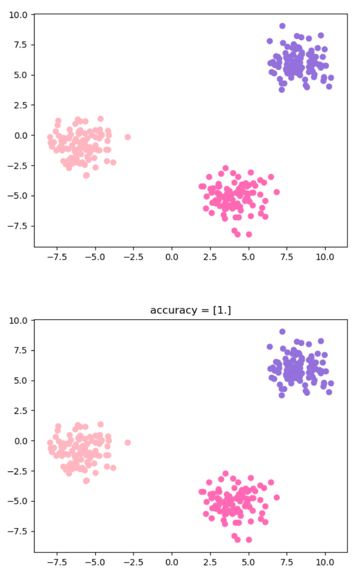
Figure 6: Spectral learning on Gaussian distribution datasets. The first row is the ground truth clustering results, each color represents a cluster. The second row is the clustering result using spectral clustering, the title shows the clustering accuracy.

As we can see from the pictures, dimension is not the decisive factor of spectral clustering. The initial cluster centers are an influential factor. if the center of clusters if a little bit close, such as the second set of pictures 'dimension = 3' spectral clustering can still get good clustering results. However, if the cluster centers are too close, such as the fourth and fifth set of pictures, spectral clustering will not achieve good results. Maybe that is because spectral clustering calculates the distance between data points and then do clustering, but the data points of different clusters are too close to each other, thus making spectral clustering results wrong.

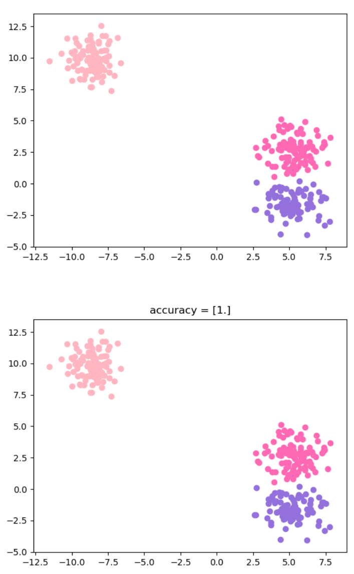
Second, I generate datasets using 'make_blob'. The toy dataset already test 'make blob' with dimension of 2, so I 'make_blob' with higher dimensions. To make comparison easier, I fix cluster number to 3, the number of all data points to 300, and vary the dimension from 2, 3 to 5. The results are shown as Fig.7. The first row is the ground truth clustering results, each color represents a cluster. The second row is the clustering result using spectral clustering, the title shows the clustering accuracy.



(a) dimension = 2



(b) dimension = 2



(c) dimension = 2

Figure 7: Spectral learning on blob datasets. The first row is the ground truth clustering results, each color represents a cluster. The second row is the clustering result using spectral clustering, the title shows the clustering accuracy.

Since the center of blobs are more disperse than Gaussian distribution data, spectral clustering performs quite well, it achieve 100% accuracy on all datasets.

In all of the parameter settings, I set the n_component of spectral clustering to the actual 3. We can see from the results that spectral clustering performs quite well except the clusters overlap with each other. In conclusion, spectral clustering works well when the data points in the same cluster is close to each other while data points in different clusters far away from each other; spectral clustering fails when the distances of the data points in the same cluster and in different clusters are similar.

6 Conclusion

In this project, I revised the algorithms for PCA: eigen decomposition and SVD; I revised the principle and algorithm of FA, and implement FA using sklearn; I revised the principle and algorithm of ICA, and implement it using sklearn.FastICA; I revised the principle and algorithm of spectral learning, and implement it using sklearn. I use AIC and BIC to select the best n_component, and I learnt to calculate AIC score and BIC score by myself. This homework help me get a deeper understanding of what I learnt in machine learning lectures. My code can be found on github: https://github.com/JiayiXuDaisy/Machine-Learning.