Homework 2

CS420, Machine Learning, Shikui Tu, Spring 2020

1 PCA algorithm

1.1 PCA based on eigenvalue decomposition

The computational details can refer to Alg. 1.

Algorithm 1: PCA based on eigenvalue decomposition

```
Input  : Dataset $X = \{x_1, \dots, x_N\}, x_t \in \mathbb{N}^{n \times 1}, \forall t$
Output: The first principal component w
1 Normalize $X$ so that the mean of $X$ is 0;
2 Calculate the covariance matrix of $X$:
      $C \leftarrow XX^T$;
3 Calculate the eigenvalues $\lambda$ and corresponding eigenvectors $V$ of $C$;
4 Select the maximal eigenvalue $\lambda_m$ and corresponding eigenvector $v_m$;
5 Calculate the first principal component:
      $w \leftarrow v_m^T X$;
6 return w;
```

Advantages:

1. This algorithm is easy to implement and understand;
2. This algorithm is unsupervised learning so sample labels are unnecessary.

Limitations:

1. This algorithm needs to find the covariance matrix first. When the number of samples is large , the calculation is very large;
2. We can only get the first principal component in one direction(row or column);

1.2 PCA based on SVD

The computational details can refer to Alg. 2.

```
Algorithm 2: PCA based on SVD
    Input  : Dataset $X = \{x_1, \ldots, x_N\}, x_t \in \mathbb{N}^{n \times 1}, \forall t$
    Output: The first principal component w
1   Normalize $X$ so that the mean of $X$ is 0;
2   Carry out SVD for $X$:
                        $X \leftarrow U\Sigma V^T;$
3   Multiply the transposition of matrix $U$ on both sides of the above formula, then the compression
      of original data is $\Sigma V^T$, denoted by $\tilde{X}$;:
                        $U^T X \leftarrow \Sigma V^T;$
4   Assign the first row of $\tilde{X}$ to w;
5   return w;
```

Advantages:

1. SVD can handle any matrix but eigenvalue decomposition can only handle square matrices;
2. Because there is no need to calculate the covariance matrix, the amount of calculation is small;
3. This algorithm can get the first principal component in both the row and column;

Limitations :

1. The data conversion is a little difficult to understand;
2. The compression results of SVD are lack of interpretability.

2 Factor Analysis (FA)

According to Bayesian formula, we have

$$\begin{aligned} p(y|x) &= \frac{p(x|y)p(y)}{p(x)} \\ &= \frac{G(x|Ay+\mu, \Sigma_e)G(y|0, \Sigma_y)}{p(x)} \\ &= \frac{G(x|Ay+\mu, \Sigma_e)G(y|0, \Sigma_y)}{p(Ay+\mu+e)} \end{aligned}$$

According to $p(y) = G(y|0, \Sigma_y)$, we have

$$E(x) = E(Ay+\mu+e) = E(y) + \mu + E(e)$$

$$\begin{aligned} Cov(x) &= Cov(Ay+\mu+e) \\ &= E((Ay+\mu+e-E(x))(Ay+\mu+e-E(x))^T) \\ &= A\Sigma_y A + \Sigma_e \end{aligned}$$

The result is

$$p(y|x) = \frac{G(x|Ay+\mu, \Sigma_e)G(y|0, \Sigma_y)}{G(x|\mu+\mu_e, A\Sigma_y A^T + \Sigma_e)}$$

3 Independent Component Analysis (ICA)

ICA comes from the classic cocktail party problem and it aims to decompose the mixed source signal into independent parts. If the source signals are non-Gaussian, the decomposition is unique, or there would be a variety of such decompositions so that we can't get the real voice signal of everyone.

Suppose the source signal s consists of two normal distributions as $N(0, I)$. Obviously, s satisfies Gaussian distribution .

We have $x = As$, where x denotes the actual signals received while A represents a mixing matrix. Then x is also Gaussian, with a mean of 0 and a covariance of

$$E[xx^T] = E[Ass^T A^T] = AA^T$$

We denote $A^{'} = AR$ where R is a normalized orthogonal matrix. If A is replaced by A', then we can get $x^{'} = A^{'}s$. We can find that $x^{'}$ conforms the same distribution as x, with a mean of 0 and a covariance of

$$E[x^{'}(x^{'})^T] = E[A^{'}ss^T(A^{'})^T] = E[ARss^T(AR)^T] = ARR^TA^T = AA^T$$

So we can't determine the mixing matrix from the received signals so that we can't the real voice signal of everyone. Therefore, maximizing non-Gaussianity should be used as a principle for ICA estimation.

4 Dimensionality Reduction by FA

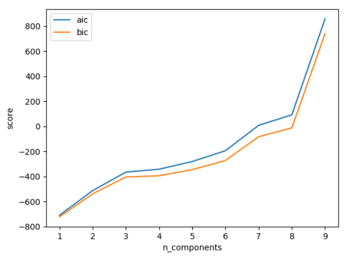
In order to compare the model selection performance of AIC and BIC, I changed some settings, such as sample size N, dimensions n and m, noise level σ2 and mean μ. I experimented and analyzed according to different settings.

4.1 Model selection performance with varied sample size N

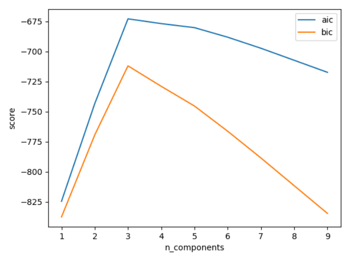
4.1.1 Setup

1. N=[10,50,100,200,300,500], n=10, m=3, $\sigma^2 = 0.1$, $\mu = 0$
2. $y_t \sim G(y|0, I)$
3. $e_t \sim G(e|0, \sigma_2 I), e_t \in$ **Rⁿ**
4. $X_t = Ay_t + \mu + e_t$

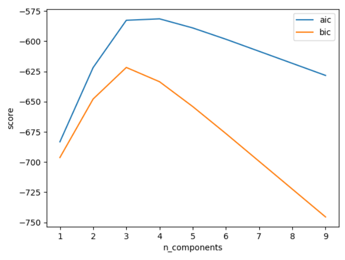
4.1.2 Results



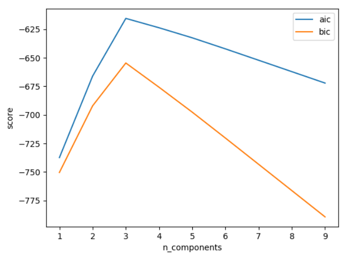
(a)



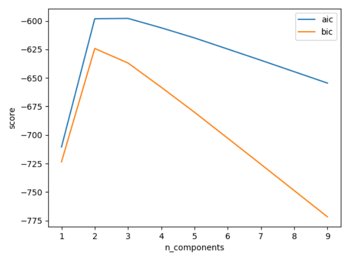
(b)



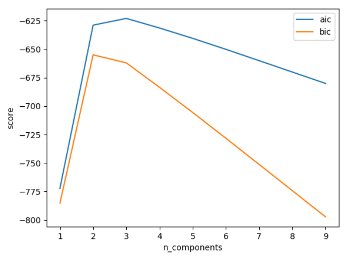
(c)



(d)



(e)



(f)

Figure 1: Sample size N in [10, 50, 100, 200, 300, 500].

In this section I vary the sample size N in {10,50,100,200,300,500} and run AIC and BIC respectively as in Figure1.

From the results we can find that their performances don't increase monotonically with the growth of sample size N and they perform poorly when N is very small to 10. As N increases, they can get the correct value or a result close to the correct value. Overall, BIC performes better and the reason may be that BIC takes N into consideration.

4.2 Model selection performance with varied dimension n

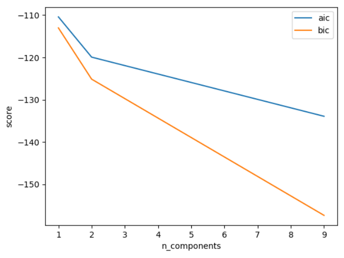
4.2.1 Setup

1. N=100, n=[2,4,6,8,10,12], m=3, $\sigma^2 = 0.1$, $\mu = 0$
2. $y_t \sim G(y|0, I)$
3. $e_t \sim G(e|0, \sigma_2 I), e_t \in \mathbf{R^n}$
4. $X_t = Ay_t + \mu + e_t$

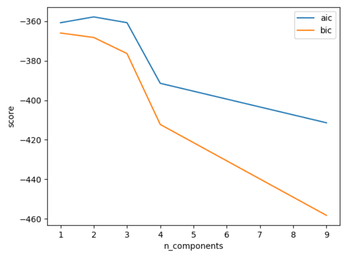
4.2.2 Results

In this section I vary the dimensiton n in {2,4,6,8,10,12} and run AIC and BIC respectively as in Figure2.

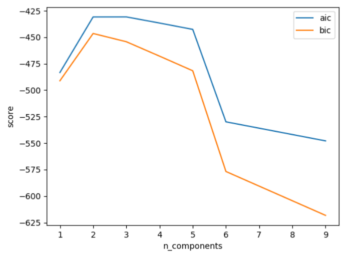
From the results we can find that both BIC and AIC perform normally when n is very small. As n grows, their selection results are getting better. Besides, their choices are always the same.



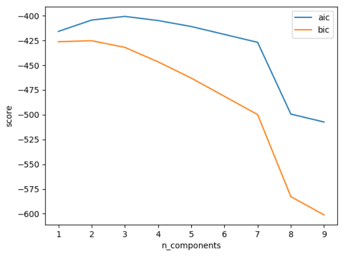
(a)



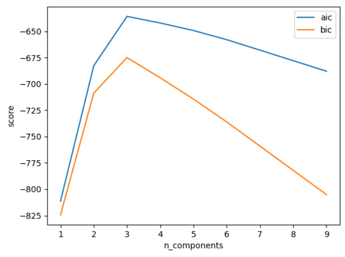
(b)



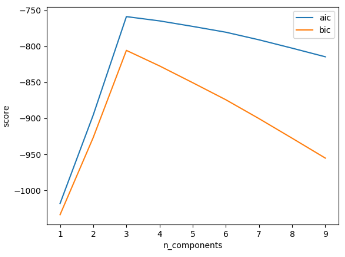
(c)



(d)



(e)



(f)

Figure 2: Dimension n in [2, 4, 6, 8, 10, 12].

4.3 Model selection performance with varied dimension m

4.3.1 Setup

1. N=100, n=10, m=[1,2,3,4,5,6], $\sigma^2 = 0.1$, $\mu = 0$
2. $y_t \sim G(y|0, I)$
3. $e_t \sim G(e|0, \sigma_2 I), e_t \in \mathbf{R^n}$
4. $X_t = Ay_t + \mu + e_t$

4.3.2 Results

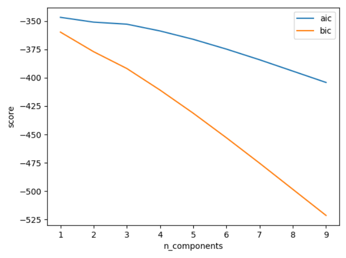
In this section I vary the dimensiton m in $\{$1,2,3,4,5,6$\}$ and run AIC and BIC respectively as in Figure3.

From the results we can find that under this parameter setting, AIC and BIC can select the correct value or the result close to the correct value as m changes. However, BIC performs better when m=6.

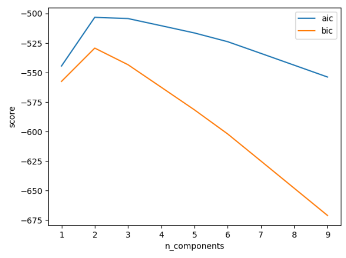
4.4 Model selection performance with varied mean μ

4.4.1 Setup

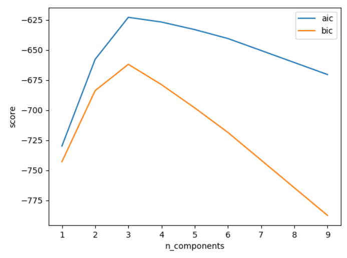
1. N=100, n=10, m=3, $\sigma^2 = 0.1$, $\mu = [-1, -0.5, 0, 0.5, 1, 2]$
2. $y_t \sim G(y|0, I)$
3. $e_t \sim G(e|0, \sigma_2 I), e_t \in \mathbf{R^n}$
4. $X_t = Ay_t + \mu + e_t$



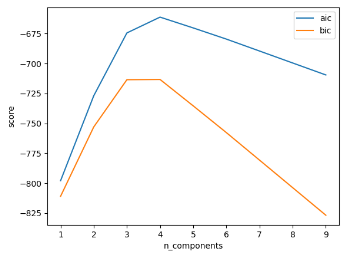
(a)



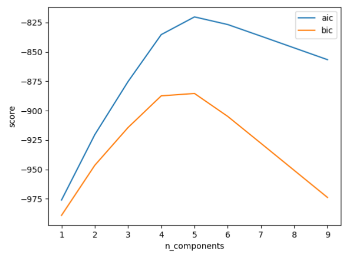
(b)



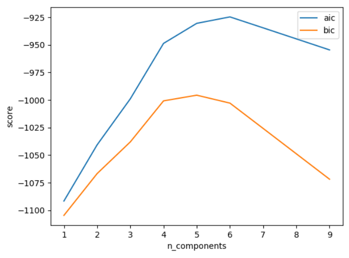
(c)



(d)

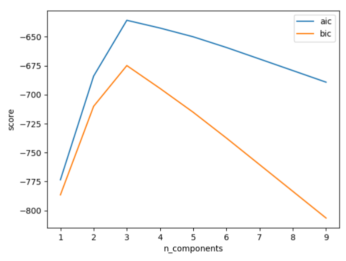


(e)

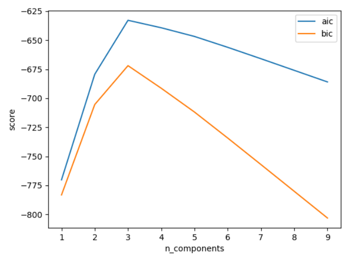


(f)

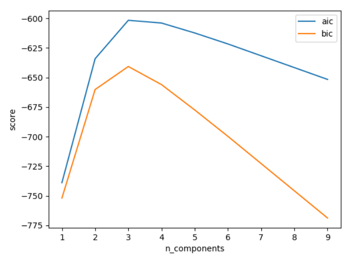
Figure 3: Dimension m in [1, 2, 3, 4, 5, 6].



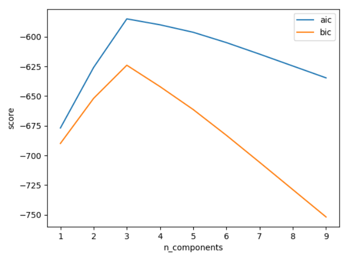
(a)



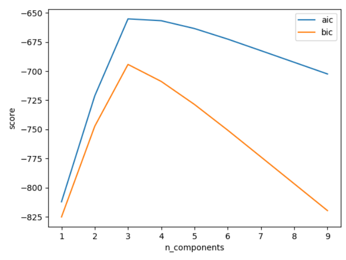
(b)



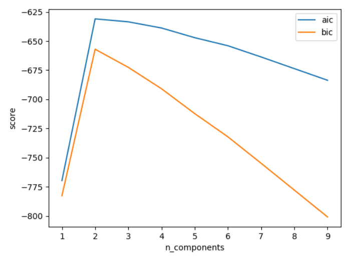
(c)



(d)



(e)



(f)

Figure 4: Mean μ in [-1, -0.5, 0, 0.5, 1, 2].

4.4.2 Results

In this section I vary the dimensiton μ in {-1, -0.5, 0, 0.5, 1, 2} and run AIC and BIC respectively as in Figure4.

From the results we can find that AIC and BIC are relatively stable as μ changes although they choose 2 instead of 3 when $\mu = 2$.

4.5 Model selection performance with varied noise level σ^2

4.5.1 Setup

1. N=100, n=10, m=3, $\sigma^2 = [0.0001, 0.001, 0.01, 0.1, 1, 10]$, $\mu = 0$
2. $y_t \sim G(y|0, I)$
3. $e_t \sim G(e|0, \sigma_2 I), e_t \in \mathbf{R^n}$
4. $X_t = Ay_t + \mu + e_t$

4.5.2 Results

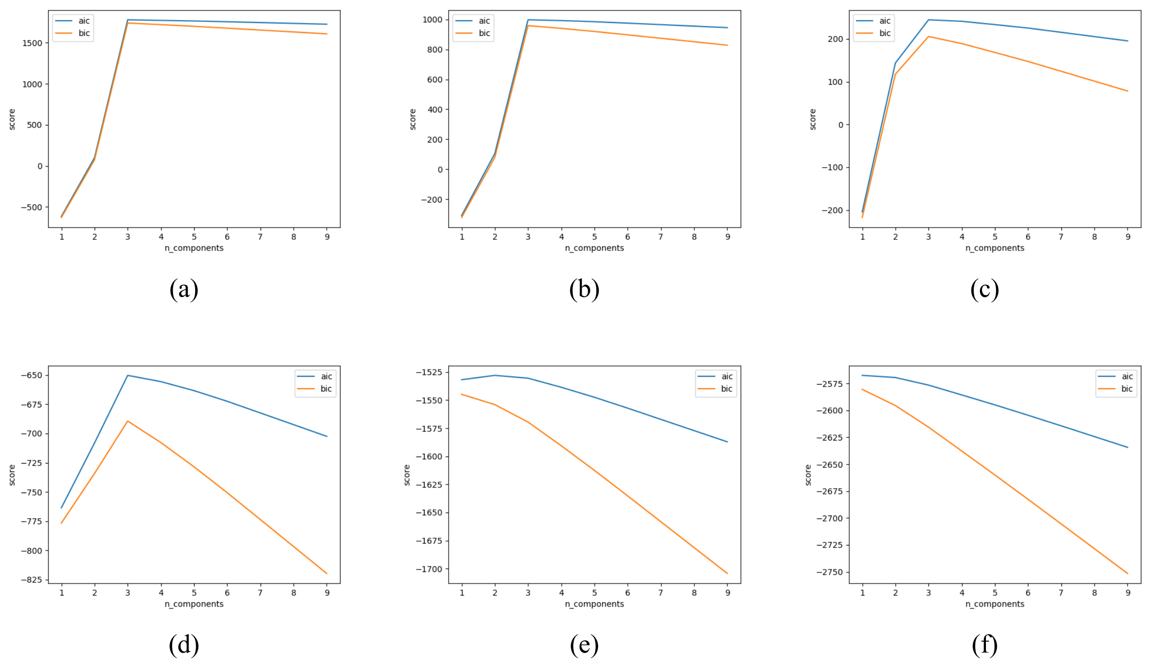


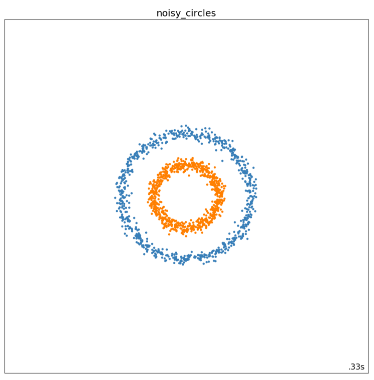
Figure 5: Noise level σ^2 in [0.0001, 0.001, 0.01, 0.1, 1, 10].

In this section I vary the dimensiton σ^2 in {0.0001, 0.001, 0.01, 0.1, 1, 10} and run AIC and BIC respectively as in Figure5.

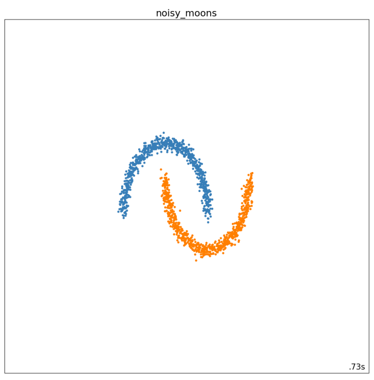
From the results we can find that AIC and BIC perform stable when σ^2 is less than or equal to 0.1, but perform poorly when σ^2 is greater than 0.1.

5 Spectral clustering

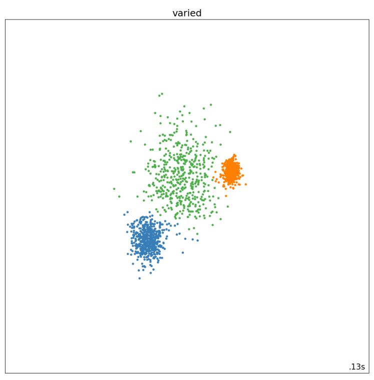
In this section, I test the spectral clustering on 6 kinds of datasets, including noisy circles, noisy moons, varied, aniso, blobs and no structure.The clustering results are in figure6



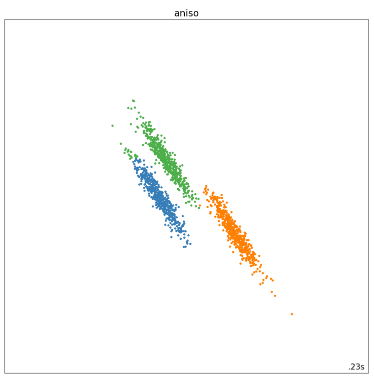
(a)



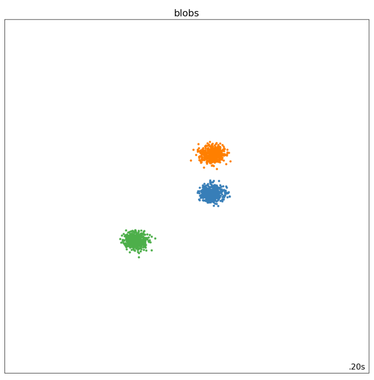
(b)



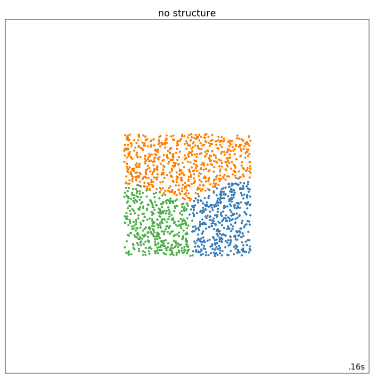
(c)



(d)



(e)



(f)

Figure 6: Spectral clustering on several datasets.

From the results, we can find that the spectral clustering works well in structured dataset such as noisy circles, noisy moons, varied, and blobs. In addition, varied and blobs have sparseness in common. However, spectral clustering works not very wel for aniso. I guess the reason is the proximity matrix in spectral clustering is usually based on distance between data points. And for unstructured dataset , spectral clustering also works not well.