516021910212
Liu Yuxi

Homework 2

CS 420
May 9, 2019

1 PCA algorithm

Give at least two algorithms that could take data set $X = \{x1, ..., xN\}, xt \in \boldsymbol{R_{n\times 1}}, \forall t$ as input, and output the rst principal component **w**. Specify the computational details of the algorithms, and discuss the advantages or limitations of the algorithms.

1.1 Eigen Decomposition

Algorithm 1: Eigen Decomposition

Input: data set **X**
Output: the rst principal component **w**

Conduct the mean vector $\overline{x} \leftarrow \sum_{j=1}^{N} x_j$

for *each* x_j **do**
  $x_j \leftarrow x_j - \overline{x}$

Find the covariance matrix of X

$$\mathbf{C} \leftarrow \frac{1}{n} X X^T$$

Using eigen decomposition to calculate the eigen values λ_i and eigen vectors α_i of C

Choose the maximal eigenvalue ϵ_m and corresponding eigen vector α_m Calculate the first priciple component

$$\mathbf{w} \leftarrow \alpha_m X$$

return **w**;

The eigen decomposition method is easy to implement and it is widely used. And it has both advantages and disadvantages.

Advantages

1. Easy to implement and understand
2. Measures information by variance is not subject to sample labeling.
3. Orthogonal between the main components, which can eliminate the interaction between the original data components

Disadvantages

1. The principal component explains its meaning often with some ambiguity, not as complete as the original sample
2. $X_T X$ needs large amount of calculation when dataset is large

1.2 SVD

Algorithm 2: Eigen Decomposition

Output: the rst principal component **w**

1 Conduct the mean vector $\overline{x} \leftarrow \sum_{j=1}^{N} x_j$

2 **for** *each* x_j **do**

3 $\quad x_j \leftarrow x_j - \overline{x}$

4 Carry out SVD for **X**

$$X \leftarrow U\Sigma V^T$$

5 Multiply the transposition of **U** on both sides of the above formula

$$U^T X \leftarrow \Sigma V^T$$

6 $U^T X$ is a $d \times n$ matrix (d is the row number of U), which means it compresses X in rows. So the vector corresponding to the max λ in Σ is the needed w. According to SVD, the first row vector α_1 usually corresponding to the maximum λ

$$\tilde{X} \leftarrow U^T X$$

7 Assign the first row of $\tilde{X}$ to **w**

8 return **w**;

Advantages

1. SVD could handle matrices that are not square matrices, while the eigen decomposition is only applicable to the square matrices
2. Reduce the amount of calculation intensively without calculating $X^T X$ comparing with eigen decomposition

Disadvantages

1. The compression results of SVD are lack of interpretability but there is no influence on its performance.
2. The complexity of calculating the singular values is $O(N^3)$, which is not suitable for large scale samples.

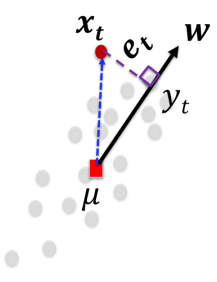

Figure 1: Error definition

1.3 Lagrangian Multiplier

The definition of calculating error is shown in Figure 1.

For Lagrangian Multiplier:

1. Advantages: has low computation complexity
2. Disadvantages: can not compute all the components

Algorithm 3: Lagrangian Multiplier

Output: the rst principal component **w**

1 Define the error

$$J(w) = \frac{1}{N}\sum_{t=1}^{N} ||x_t - (x_t^T w)w||^2$$

2

$$L(w, \lambda) = J(w) - \lambda(w^T w - 1)$$

3

$$\frac{\partial L}{\partial w} = \frac{\partial J}{\partial w} - \lambda\frac{\partial(w^T w - 1)}{\partial w} = 0$$

4

$$\frac{1}{N}\sum_{t=1}^{N} x_t x_t^T = (-\lambda)w$$

return the w_i which has the largest λ_i

2 Factor Analysis

According to $p(y) = G(y|0, I)$, we can get $E(x)$ and $Cov(x)$:

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

According to Bayes formula:

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

where μ_e denotes the mean value of $\mathbf{e}$, Σ_e denotes the covariance matrix of $\mathbf{e}$.

3 Independent Component Analysis(ICA)

The ICA believes that a signal can be broken down into a linear combination of several statistically independent components. This decomposition is unique as long as the source signal is not Gaussian. If the source signal is Gaussian, then there may be an infinite number of such decompositions.

Prove:

Suppose that only two people emit sound signals that match the multi-valued normal distribution

$$S \sim N(0, I)$$

I is a unit matrix of 2×2

$$x = As$$

So X suits normal distribution.

$$x \sim N(0, AA^T)$$

Let R be an orthogonal matrix, where $RR^T = R^T R = I$

$$A' = AR$$

Replace A with A' and we get $s' = A's$. The distribution of S has not change so

$$x' \sim N(0, AA^T)$$

Therefore, regardless of whether the mixing matrix is A or A', the distribution of x is the same, then the mixing matrix cannot be determined and the original signal cannot be determined.

4 Dimensionality Reduction by FA

In this part I test the Factor Analysis model. Firstly I generate the data subject to Gaussian distribution.

4.1 Setup

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

4.2 Results

I implemented two model selection process in my experiment, which are AIC and BIC. When the sample size is small and the vector is low dimension, it is very clear that the loglikelihood curse has a turning point at the setting m. This phenomenon can be seen in Figure 2 and Figure 3. Both curves get to their peaks exactly at the setting common factor. And the BIC curve falling faster than AIC curve for its considering the sample size.

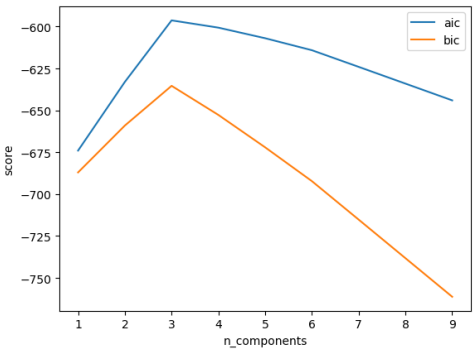

Figure 2: n_components=3

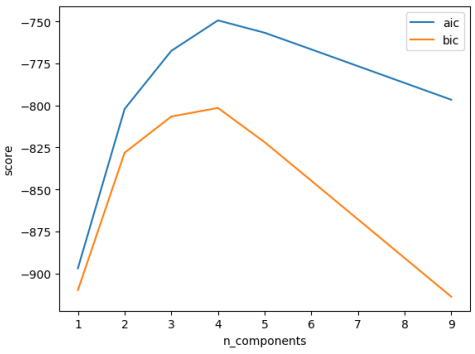

Figure 3: n_components=4

However, there are some fail examples when I turn the y_m larger. The fail example occurs in Figure 4. Although AIC choose the correct n_components, BIC seems to penish too much for high components so it choose n_components as 4. I think it is because the original sample factor has a low dimension so $m = 5$ is very large for a 10-dimensional vector. When I turn the sample vector to 20-dimension, this problem is solved, which can be seen in Figure 5. And both AIC and BIC gets a good result.

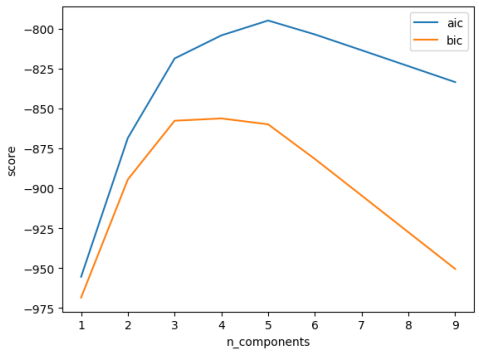

Figure 4: n_components=5 when sample vector is 10-dimension

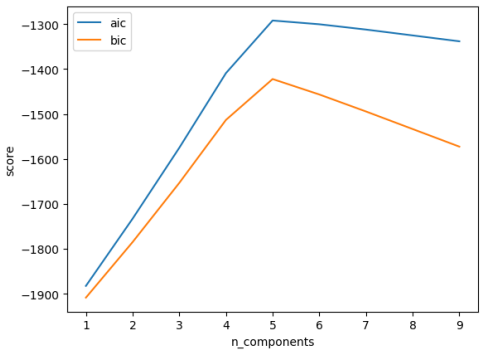

Figure 5: n_components=5 when sample vector is 20-dimension

5 Spectral Clustering

5.1 Success examples

Spectral Clustering (SC) is a graph-based clustering methoddividing a weighted undirected graph into two or more optimal subgraphs, making the subgraphs as similar as possible within the subgraph. The distance between the maps should be as far apart as possible to achieve the purpose of common clustering. The main advantages of the spectral clustering algorithm are:

1. Spectral clustering only requires a similarity matrix between data, so it is effective for clustering processing sparse data, which is difficult for traditional clustering algorithms such as K-Means.

2. Due to the use of dimensionality reduction, the complexity of processing high-dimensional data clustering is better than traditional clustering algorithms.

Spectral clustering works well when the dataset is sparse like Figure 6,7,8 and 9.

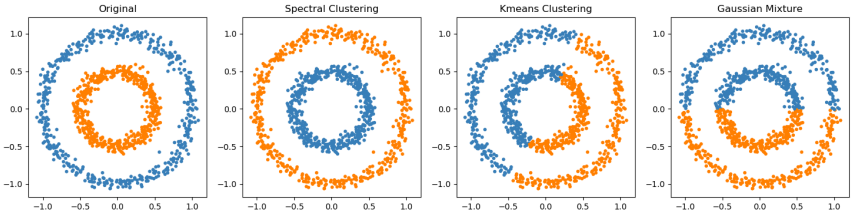

Figure 6: make_circle

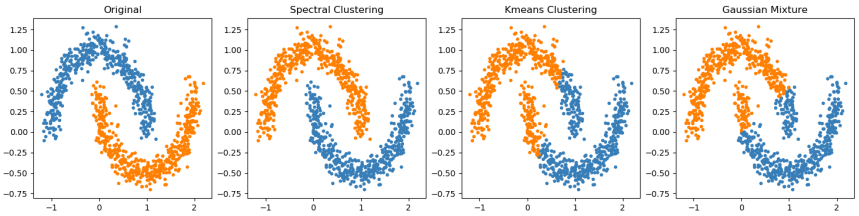

Figure 7: make_moon

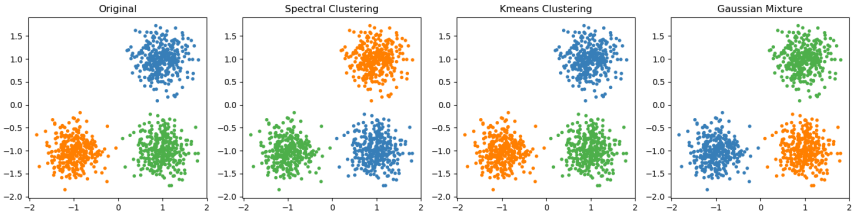

Figure 8: Separate dataset

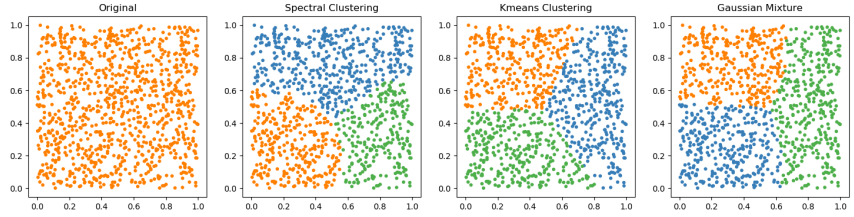

Figure 9: No structure

5.2 Fail Examples

Sometimes spectral clustering algorithms will fail:

1. If the dimension of the final cluster is very high, the running speed of the spectral clustering and the final clustering effect are not good due to the insufficient magnitude of dimensionality reduction.

2. The clustering effect depends on the similarity matrix, and the final clustering effect obtained by different similarity matrices may be different.

Spectral clustering is not effective for clustering processing dense data while traditional clustering method works pretty well, which is shown in 10.

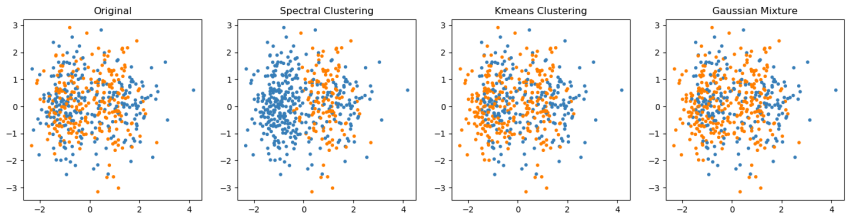

Figure 10: Fail

And spectral clustering depends on the similarity matrix. In Figure 6 and Figure 7. I endow affinity with *nearest_neighbors* and it works well. However, when I endow affinity with *rbf* or *sigmoid*, spectral clustering lost its power.

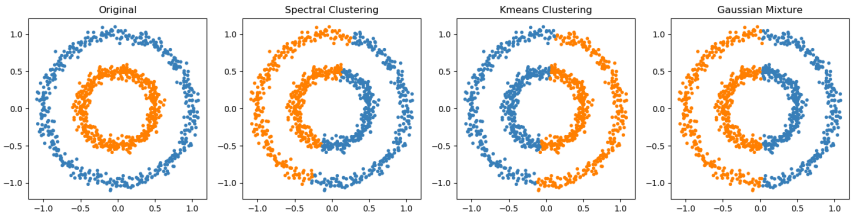

Figure 11: circle sigmoid

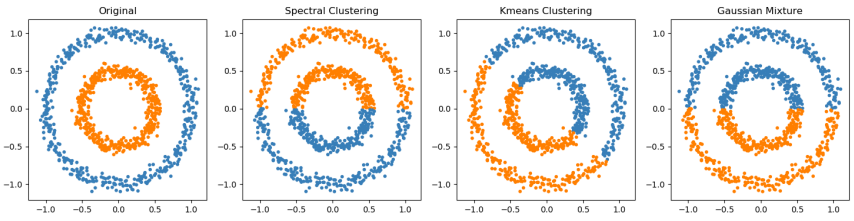

Figure 12: circle rbf

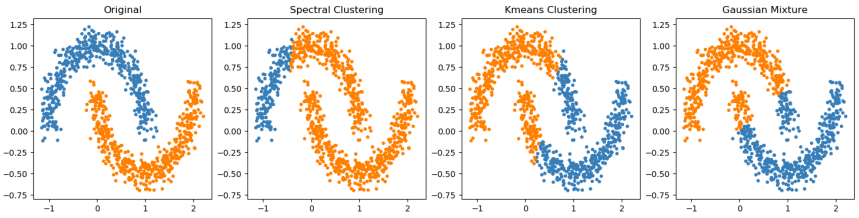

Figure 13: moon sigmoid

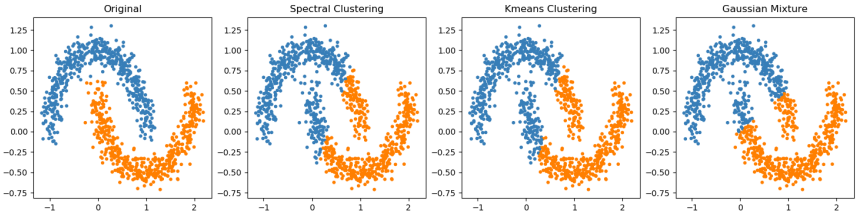

Figure 14: moon rbf

5.3 Conclusion

As a conclusion, when the spectral clustering works well, it needs:

1. Sparse subplots, if the subplots cross with each other, then spectral can not specify it.
2. Suitable similarity matrix, this need experiment to choose the best one
3. High dimensional data will perform better

And when it fails, the occasions are:

1. Dense subplots, the subplots crossing with each other makes it difficult to specify
2. Wrong similarity matrix