516021910395 Yuxiang Liu

Homework 2

Machine Learning May 10, 2019

1 PCA algorithm

1.1 PCA based on Eigen/diagonal Decomposition

1.1.1 Algorithm description

The first algorithm is based on eigen/diagonal decomposition, and the pseudo code is as Alg. 1.

Algorithm 1: PCA based on Eigen/diagonal Decomposition

```
Input: Dataset $X = \{x_1, \cdots, x_N\}, x_t \in \mathbb{N}^{n\times 1}, \forall t$.
Output: The first principal component $\mathbf{w}$.
1 Normalize $x_t, \forall t \in \{1, 2, \cdots, N\}$ such that the corresponding mean is 0;
2 Calculate the covariance matrix of $X$:
      $C \leftarrow XX^T;$
3 Calculate the eigenvalues and eigenvectors of $C$;
4 Select the maximum eigenvalue $\lambda_m$ along with its corresponding eigenvector $\mathbf{x_m}$;
5 Calculate the first principal component
      $\mathbf{w} \leftarrow \mathbf{x_m}^T X;$
6 return w;
```

1.1.2 Discussion

Advatages

1. This algorithm is easy to understand and to implement.
2. Sample labels are unnecessay since the algorithm is unsupervised learning.

Limitations

1. The computation can be large when the size of the dataset is large because the covariance matrix XX^T has to be calculated first.
2. We can only reduce the dimension by row compression.
3. The procedure to calculate the covariance matrix may introduce slight lack of precision.

1.2 PCA based on Singular Value Decomposition

1.2.1 Algorithm description

The second algorithm is based on singular value decomposition, and the pseudo code is as Alg. 2.

Algorithm 2: PCA based on Eigen/diagonal Decomposition

```
Input: Dataset $X = \{x_1, \cdots, x_N\}, x_t \in \mathbb{N}^{n \times 1}, \forall t$.
Output: The first principal component $\mathbf{w}$.
1 Normalize $x_t, \forall t \in \{1, 2, \cdots, N\}$ such that the corresponding mean is 0;
2 Conduct SCD on $X$:
      $X \leftarrow U\Sigma V^T;$
3 Multiply $U^T$ on both sides of above formula and the right side expression is the
    compression of original data:
      $U^T X \leftarrow \Sigma V^T;$
4 Get the first row of $\Sigma V^T$ and we get $\mathbf{w}$;
5 return $\mathbf{w}$;
```

1.2.2 Discussion

Advatages

1. SVD can be performed on nonsymmetric matrix.
2. The computation will not increase rapidly when the size of dataset gets large since we don't have to calculate XX^T to conduct decomposition.
3. Actually, SVD can reduce the dimension from both directions.

Limitations

1. Dataset have to be zero-mean and the sparsity of data is lost.
2. The decomposited matrix is weak in interpretability.

2 Factor Analysis (FA)

From Bayesian formula, we have

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

Since $\mathbf{x} = \mathbf{A}\mathbf{y} + \mu + \mathbf{e}$, $\mathbf{y}$ satisfies Gaussian distribution as $G(\mathbf{y}|0,\Sigma_y)$ and μ is constant, thus we get

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

where μ_e is the mean of $\mathbf{e}$ and is 0 generally.

Combine two formulas above, we can finally get

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

Since it is obvious that $p(\mathbf{y}|\mathbf{x})$ is a Gaussian distribution, we can make further calculation. The exponent of a general Gaussian distribution can be written as

$$-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu) = -\frac{1}{2}\mathbf{x}^T\Sigma^{-1}\mathbf{x} + \mathbf{x}^T\Sigma^{-1}\mu + const$$

Similarly, we can denote the exponent of $p(\mathbf{y}|\mathbf{x})$ as

$$-\frac{1}{2}\{[\mathbf{x}-(\mathbf{A}\mathbf{y}+\mu)]^T\Sigma_e^{-1}[\mathbf{x}-(\mathbf{A}\mathbf{y}+\mu)] + \mathbf{y}^T\Sigma_y^{-1}\mathbf{y} - [\mathbf{x}-(\mu+\mu_e)]^T(\mathbf{A}\Sigma_y\mathbf{A^T}+\Sigma_e)^{-1}[\mathbf{x}-(\mu+\mu_e)]\}$$

Then we can get terms that contain $\mathbf{y}$ as

$$-\frac{1}{2}\mathbf{y}^T(\mathbf{A}^T\Sigma_e^{-1}\mathbf{A}+\Sigma_y^{-1})\mathbf{y}$$

$$\frac{1}{2}[\mathbf{y}^T\mathbf{A}^T\Sigma_e^{-1}(\mathbf{x}-\mu) + (\mathbf{x}-\mu)^T\Sigma_e^{-1}\mathbf{A}\mathbf{y}] = (\mathbf{x}-\mu)^T\Sigma_e^{-1}\mathbf{A}\mathbf{y}$$

Therefore, we have

$$\begin{aligned} \Sigma_{\mathbf{y}|\mathbf{x}} &= (\mathbf{A}^T\Sigma_e^{-1}\mathbf{A}+\Sigma_y^{-1})^{-1} \\ \Sigma_{\mathbf{y}|\mathbf{x}}^{-1}\mu_{\mathbf{y}|\mathbf{x}} &= \mathbf{A}^T\Sigma_e^{-1}(\mathbf{x}-\mu) \\ \mu_{\mathbf{y}|\mathbf{x}} &= \Sigma_{\mathbf{y}|\mathbf{x}}\mathbf{A}^T\Sigma_e^{-1}(\mathbf{x}-\mu) \\ &= (\mathbf{A}^T\Sigma_e^{-1}\mathbf{A}+\Sigma_y^{-1})^{-1}\mathbf{A}^T\Sigma_e^{-1}(\mathbf{x}-\mu) \end{aligned}$$

Then we can conclude that

$$p(\mathbf{y}|\mathbf{x}) = G(\mathbf{y}|\Sigma_{\mathbf{y}|\mathbf{x}}\mathbf{A}^T\Sigma_e^{-1}(\mathbf{x}-\mu), (\mathbf{A}^T\Sigma_e^{-1}\mathbf{A}+\Sigma_y^{-1})^{-1})$$

3 Independent Component Analysis (ICA)

ICA is to decompose the mixed source signal into independent parts. If the mixed source signals are non-Gaussian, the decomposition can be unique, or there will be many kinds of decompositions.

I use an example to explain. Suppose that the source signal s consists of two normal distributions as $N(0, I)$, then s satisfies Gaussian distribution, and we have the received signal $x = As$ which is also Gaussian. The mean of x is 0 and the covariance matrix is

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

For a normalized orthogonal matrix R, we denote $A' = AR$ and $x' = A's$. Thus, we have

$$E[x'x'^T] = E[A'ss^T A'^T] = E[ACss^T (AC)^T] = ACC^T A^T = AA^T.$$

It is obvious that x and x' satisfy the same distribution so we cannot determine the mixing matrix from the received signals as well as the source signals. However, if the Gaussian property can be avoided, the above case will not exist.

Therefore, we have, maximizing non-Gaussianity could be used as a principle for ICA estimation.

4 Dimensionality Reduction by FA

To compare the model selection performance by AIC and BIC, I vary some setting values such as sample size N, dimensionality n and m, noise level σ^2, and mean μ. And my benchmark setting for a better comparison is: sample size $N = 100$, observerd variable dimension $n = 7$, latent variable dimension $m = 3$, variance $\sigma^2 = 0.1$, and mean $\mu = 0$.

4.1 Model selection performance with varied sample size N

In this section I vary the sample size N in $\{10, 20, 50, 80, 100, 200, 300, 500\}$ then run AIC and BIC respectively as in Table 1.

From the results, we can find that the performance doesn't increase with sample size monotonously but shows some fluctuation, which may due to the randomness of the dataset. However, the tendency of performance increases with the smple size for both AIC and BIC. Particularly, perdormance of BIC first reaches optimal and is better than that of AIC.

Table 1: Sample size

N	n	m	variance	mean	AIC	BIC	AIC value	BIC value
10	7	3	0.1	0	5	5	-35.7441	-36.5005
20	7	3	0.1	0	4	4	-89.381	-91.3725
50	7	3	0.1	0	4	3	-284.506	-288.032
80	7	3	0.1	0	4	4	-418.718	-423.482
100	7	3	0.1	0	4	3	-549.93	-554.327
200	7	3	0.1	0	4	3	-1153.28	-1158.41
300	7	3	0.1	0	3	3	-1646.78	-1652.34
500	7	3	0.1	0	4	3	-2923.21	-2931.03

4.2 Model selection performance with varied dimension n and m

4.2.1 Observed variable dimension n

First, I vary the observed variable dimension n in $\{2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15\}$. The results of AIC and BIC are shown in Table 2.

Table 2: Dimension n

N	n	m	variance	mean	AIC	BIC	AIC value	BIC value
100	2	3	0.1	0	1	1	-211.543	-212.846
100	3	3	0.1	0	1	1	-321.695	-322.998
100	4	3	0.1	0	2	2	-320.224	-322.829
100	5	3	0.1	0	3	3	-469.34	-473.248
100	6	3	0.1	0	3	3	-564.735	-568.643
100	7	3	0.1	0	4	3	-517.105	-522.205
100	8	3	0.1	0	5	3	-608.179	-613.266
100	9	3	0.1	0	5	5	-569.287	-575.8
100	10	3	0.1	0	5	5	-626.2	-632.713
100	12	3	0.1	0	5	5	-693.489	-700.002
100	15	3	0.1	0	5	5	-869.466	-875.979

From the experimental results, the value selected by AIC and BIC basically increases monotonously with observed variable dimension n. They perform almost the same in this setting. BIC prefers to choose smaller value than AIC and is better sometimes.

4.2.2 Latent variable dimension m

Here I vary the latent variable dimension m in $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. The results of AIC and BIC are shown in Table 3.

In my parameter setting, the performance difference between AIC and BIC is not absolute.

Table 3: Dimension m

N	n	m	variance	mean	AIC	BIC	AIC value	BIC value
100	7	1	0.1	0	1	1	-294.645	-295.948
100	7	2	0.1	0	3	2	-469.37	-472.968
100	7	3	0.1	0	3	3	-488.765	-492.673
100	7	4	0.1	0	3	3	-652.362	-656.27
100	7	5	0.1	0	4	4	-666.797	-672.007
100	7	6	0.1	0	4	4	-705.342	-710.552
100	7	7	0.1	0	5	4	-831.587	-836.808
100	7	8	0.1	0	4	4	-937.249	-942.46
100	7	9	0.1	0	5	5	-906.741	-913.254

From the results, BIC prefers to choose smaller or equal value than AIC and the value they choose basically increases with dimension m. For dimension m that is small, BIC performs better and for dimension m that is large, AIC performs better.

4.3 Model selection performance with noise level σ^2

In this part, I vary the noise level σ^2 in $\{0.0001, 0.001, 0.01, 0.1, 1, 10, 100\}$. The experimental results are in Table 4.

Table 4: Noise level (variance) σ^2

N	n	m	variance	mean	AIC	BIC	AIC value	BIC value
100	7	3	0.0001	0	4	3	829.821	825.1286
100	7	3	0.001	0	4	3	421.7515	417.686
100	7	3	0.01	0	4	3	-7.45865	-11.4836
100	7	3	0.1	0	4	3	-528.648	-532.596
100	7	3	1	0	3	3	-1126.89	-1130.79
100	7	3	10	0	3	3	-1839.86	-1843.77
100	7	3	100	0	3	2	-2578.27	-2582.02

From the results, we can see that most time BIC performs better than AIC while still sometimes AIC performs better.

4.4 Model selection performance with varied mean μ

Here I vary the mean μ in $\{-1, -0.8, -0.5, -0.2, 0, 0.2, 0.5, 0.8, 1\}$ and the results are in Table 5.

From the results, we can still see that most time BIC performs better than AIC while still sometimes AIC performs better.

Table 5: Mean μ

N	n	m	variance	mean	AIC	BIC	AIC value	BIC value
100	7	3	0.1	-1	4	3	-509	-513.827
100	7	3	0.1	-0.8	3	3	-482.648	-486.556
100	7	3	0.1	-0.5	5	4	-511.178	-516.411
100	7	3	0.1	-0.2	4	3	-495.126	-499.561
100	7	3	0.1	0	4	3	-536.343	-541.28
100	7	3	0.1	0.2	3	3	-544.559	-548.467
100	7	3	0.1	0.5	3	3	-562.756	-566.664
100	7	3	0.1	0.8	3	3	-577.5	-581.408
100	7	3	0.1	1	4	3	-597.268	-601.37

5 Spectral clustering

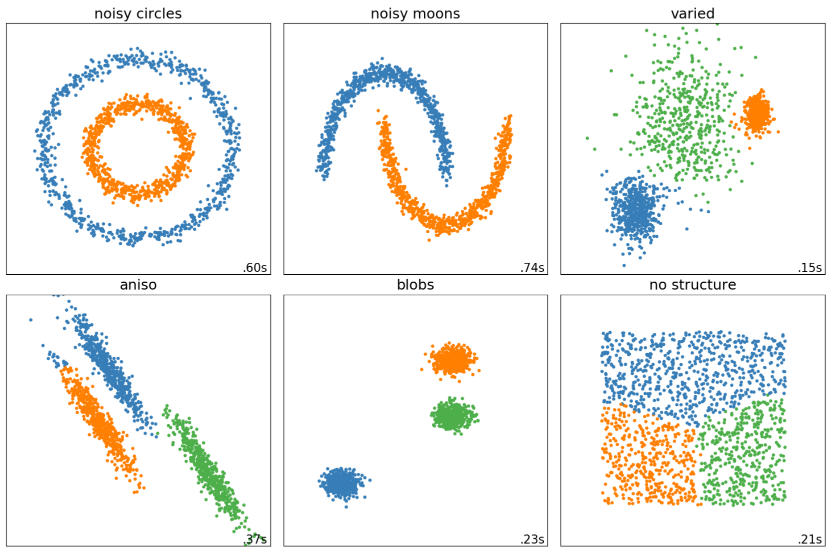

Figure 1: Spectral clustering on several datasets

In this section, I test the spectral clustering on 6 kinds of datasets: noisy circles, noisy moons, varied, aniso, blobs, and no structure. Some of them are the datasets in *sklearn*. The clustering results are in Fig. 1.

From the results, we can see that the spectral clustering works well in structured dataset where each cluster gathers in a circle or circle-like shape, such as noisy circles, noisy moons, varied, and blobs. For structured dataset as aniso which is rod-like, spectral clustering works not very well. This is beccause, the proximity matrix in spectral clustering is usually based on distance between data points. And for unstructured dataset (the last subgraph in Fig. 5), spectral clustering also works not well and fails to determine the cluster number.