Report on Homework 2

CS420, Machine Learning, Shikui Tu, Summer 2018

Zelin Ye 515030910468

1 PCA algorithm

When it comes to principal component, the most common used approach is principal component analysis (PCA) algorithm. Thus, I can use the original PCA algorithm to extract the first principal component of the dataset. The computational details can refer to Alg. 1.

Algorithm 1: Original PCA

Input : The dataset X, a $n \times N$ matrix
Output: The first principal component w
Conduct normalization for X, and make sure the mean of X is 0;
Find the covariance matrix of X, denoted by C:

$$C = XX^T;$$

Calculate the eigenvalues λ and eigenvectors V of X;
Choose the maximal eigenvalue λ_m and corresponding eigenvector v_m;
Calculate the first principal component:

$$w = v_m^T X;$$

return w;

The above algorithm (PCA) is a classical and commonly used method to solve principal components, which has the following characteristics.

Pros:

1. PCA has simple logic and is easy to implement;
2. The orthogonality among principal components chosen by PCA can eliminate the interaction between original data components;
3. PCA belongs to unsupervised learning, and it is not restricted by sample labels.

Cons:

1. PCA treats all samples, namely the collection of eigenvectors, as a whole and neglects the category attributes. Nevertheless, the projection direction it neglects might contain some important separability information;
2. PCA would be time-consuming when encountering large amount of data;
3. The actual meanings of principal components extracted by PCA are a little bit ad-hoc and hard to explain.

The original PCA might have good performance when handling linear data, but it would encounter difficulties under non-linear data. In non-linear cases, a variant of PCA called KPCA (kernel principal components analysis) shows its strength.

The innovation of KPCA is that it introduces a non-linear mapping function $\phi(x)$, mapping the data from original space to high-dimensional space. Besides, the deduction of KPCA takes advantage of the theorem that *each vector in the space could be expressed linearly by all the samples in the space*. The details of KPCA can refer to Alg. 2.

Algorithm 2: KPCA

Input : The dataset X, a $n \times N$ matrix; kernel function $\phi(x)$
Output: The first principal component w

Conduct normalization for X, and make sure the mean of X is 0;
Calculate kernel matrix K:

$$K = X^T X,$$

where X=$[\phi(x_1), ..., \phi(x_N)]$;
Find the covariance matrix of X, denoted by C:

$$C = XX^T;$$

Calculate the eigenvalues λ and eigenvectors U of K;
Choose the maximal eigenvalue λ_m and calculate corresponding eigenvector u_m;
Calculate the corresponding eigenvectors v_m of covariance matrix C by u_m:

$$\begin{aligned} v_m &= \frac{1}{\|X^T u_m\|} X^T u_m \\ &= \frac{1}{\sqrt{u_m^T X X^T u_m}} X^T u_m \\ &= \frac{1}{\sqrt{u_m^T (\lambda_m u_m)}} X^T u_m \\ &= \frac{1}{\sqrt{\lambda_m}} X^T u_m; \end{aligned}$$

Calculate the first principal component:

$$w = v_m^T X;$$

return w;

KPCA is a ingenious extension of PCA, and is also widely used (e.g. dimension reduction, clustering). The pros and cons of KPCA are as follows.

Pros:

1. Basically, KPCA owns almost all of the advantages of PCA;
2. KPCA has a stronger universality, which could find out the non-linear information contained in dataset;
3. KPCA uses simple kernel functions (e.g. polynomial kernel function) to realize complex non-linear transform;

Cons:

1. The logic and implementation of KPCA is a little bit complicated;
2. The dimension of kernel matrix K is $N \times N$ (N is the number of samples). It might take quantities of time to process K when handling large scale of samples;

3. Different choices of kernel functions and parameters would affect the result to a certain extent, which makes this algorithm more tricky.

4. Same as PCA, the actual meanings of principal components extracted by KPCA are also inexplicable.

2 Factor Analysis (FA)

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

where μ_e denotes the mean value of e, generally considered to be 0.

3 Independent Component Analysis (ICA)

ICA aims to decompose the 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.

Suppose the source signal s consists of two components, conforming to multi-valued normal distribution, namely $s \sim N(0, I)$. Obviously, the probablity density funciton of s is centered on the mean 0, and the projection plane is an ellipse.

Meanwhile, 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 $E[xx^T] = E[Ass^TA^T] = AA^T$.

Let C be a orthogonal matrix, and $A^{'} = AR$. If A is replaced by A', then we can get $x^{'} = A^{'}s$. It is easy to find that $x^{'}$ also conforms to normal distribution, with a mean of 0 and a covariance of $E[x^{'}(x^{'})^T] = E[A^{'}ss^T(A^{'})^T] = E[ACss^T(AC)^T] = ACC^TA^T = AA^T$.

Apparently, x and $x^{'}$ conform to the same distribution with different mixing matrices. Then we cannot determine the mixing matrix or the source signals from the received signals. Nevertheless, if x is non-Gaussian (e.g. Uniform Distribution), such case would be effectively avoided. Therefore, maximizing non-Gaussianity should be used as a principle for ICA estimation.

4 Causal Discovery Algorithms

4.1 Problem Description

Generally speaking, the housing price are affected by many factors, such as the number of rooms, crime rate, pupil-teacher ratio and etc. There also exist some causalities among those factors. It is beneficial for further mastering the housing market if one can figure out these causalities. However, such causalities are often difficult to determine intuitively, it is thus necessary to conduct causal discovery on these factors.

I utilize the factors and corresponding values in the Boston Housing dataset, which is available at the UCI Repository [1]. The details of the dataset can refer to Appendix A.1.

4.2 Approach

Taking the data type of features and the scale of samples into account, I apply *Kernel-based Conditional Independence Test* (KCI-test) [2] to casual discovery, which performs well when the conditioning set is large and the sample size is not very large.

Basically, conditional independence test for continuous variables is rather challenging due to the curse of dimensionality. KCI-test constructs an appropriate test statistic and deriving its asymptotic distribution under the null hypothesis of conditional independence.

pass

The main purpose of this application is to figure out the causalities among the features in Boston Housing dataset. Therefore, I use KCI-test to test the dependence and conditional independence among features. Then I construct the dag casualities based on PC algorithm [3].

4.3 Experiments

The results of unconditional independence testing on the features in Boston Housing dataset are shown in Tab. 1.

Table 1: The p-values of unconditional independence testing

p \ id / id	1	2	3	4	5	6	7	8	9	10	11	12
1	0.000	0.000	0.000	0.000	**0.003**	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
5	**0.003**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.005**	0.000	0.000
6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
7	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
10	0.000	0.000	0.000	0.000	**0.002**	0.000	0.000	0.000	0.000	0.000	0.000	0.000
11	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
12	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

The results of conditional independence testing on features can refer to Tab. 2.

Table 2: The results of conditional independence testing (Since the number of results is excessive, I only count the numbers here and not specify the results).

Condition z	The Number of Independent Features under z
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

The final result of PC algorithm can refer to Fig. 1.

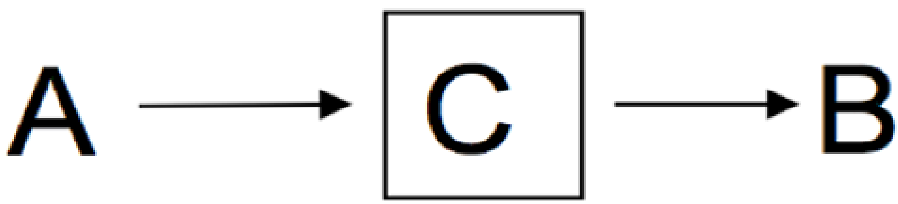

Figure 1: The final result of PC algorithm.

5 Causal Tree Reconstruction

pass

A Appendix

A.1 Details of the Boston Housing Dataset

The Boston Housing Dataset consists of 506 samples, each has 12 features, namely the factors that affect housing price. The values of all features are positive real numbers. Table 3 shows a detailed description of these features.

Table 3: Details of the features in Boston Housing dataset

Identifier	Feature Names	Description
1	CRIM	per capita crime rate by town
2	ZN	proportion of residential land zoned for lots over 25,000.000 sq.ft.
3	INDUS	proportion of non-retail business acres per town
4	NOX	nitric oxides concentration (parts per 10.0 million)
5	RM	average number of rooms per dwelling
6	AGE	proportion of owner-occupied units built prior to 1940.000
7	DIS	weighted distances to five Boston employment centres
8	TAX	full-value property-tax rate per $10,000.0
9	PTRATIO	pupil-teacher ratio by town
10.000	B	1000(Bk - 0.63)2 where Bk is the proportion of blacks by town
11	LSTAT	lower status of the population
12	MEDV	Median value of owner-occupied homes in $1000’s

References

[1] A. Asuncion and D. Newman, “Uci machine learning repository,” 2007.

[2] K. Zhang, J. Peters, D. Janzing, and B. Schölkopf, “Kernel-based conditional independence test and application in causal discovery,” *arXiv preprint arXiv:1202.3775*, 2012.

[3] P. Spirtes and C. Glymour, “An algorithm for fast recovery of sparse causal graphs,” *Social science computer review*, vol. 9, no. 1, pp. 62–72, 1991.