Spring 5-1-2018

Matrix Methods of Data Analysis

Kaiye Yu

Abstract

Nowadays, data is playing a bigger part than ever before in the history. In order to get more useful information, methods involving matrix are powerful. There are different algorithms that are able to help one to learn the information they need from data. In this study, there are two main algorithms that I will focus on. One is Pagerank algorithm, a traditional algorithm that was applied to searching engine decades ago in Google. However, Pagerank algorithm has certain limits in providing information like covariance between different factors. Thus, another method is also studied, which is principal component analysis (PCA), while there are also weak points of PCA in the study, such as missing data and this paper will provide five potential solutions to such problem. In this study, my data base is 37 students' grades on 13 different math courses and based on the information from this data pool, advisors are able to give better advice to their students.

Executive Summary

We now live in the age of big data and there is always some useful information deep in the data awaiting for digging. Algorithms like Pagerank and Principal Component Analysis (PCA) can be helpful in some cases. However, things cannot always be perfect. The data sets one gets are sometimes incomplete, and there are different strategies for dealing with missing data in PCA. In my capstone project, I want to find out the best way among those five to eliminate or alleviate the error brought by incomplete data, using Matlab. My data set is 37 math major students' performances on 13 math courses, while most of them have not finished all 13 courses. Thus, there are numerous missing data in this study. With the help of Pagerank, we can predict the courses the students will take in the future, while PCA can help us to analyze their math ability based on the data we get.

Pagerank was applied by Google to web search around 1996. In Pagerank algorithm, there are outlinks and inlinks that relate each factor, which are courses in this study, and these links between factors can transport the "rank". The rank of a factor represents its importance level. Thus, in one-directional cases, the more inlinks a factor gets, the more important this factor is. However, most situations are recursive, which means the transportation does not happen in one direction. In such case, the rank is transported forth and back between factors. When this happens, it is much more complicated to decide which factor is more important. To solve such problem, we can set up a matrix to represent mapping of rank transportation. Based on the initial condition of different factors, we can find each factor's final rank by the mapping matrix. We can imagine rank as water, link as pipes and factors as water tank. The water flows between different water tanks based on the size and number of different pipes coming in. After a long time, we can find out that the water level in each tank tends to be constant, and the tank with more water is more important.

Principal Component Analysis (PCA) is a useful dimension-reduction method to help one understands the variances between different factors. It can reduce a large set of data groups to a smaller one while the new data groups still contain the information people need. The following diagram can illustrate this.

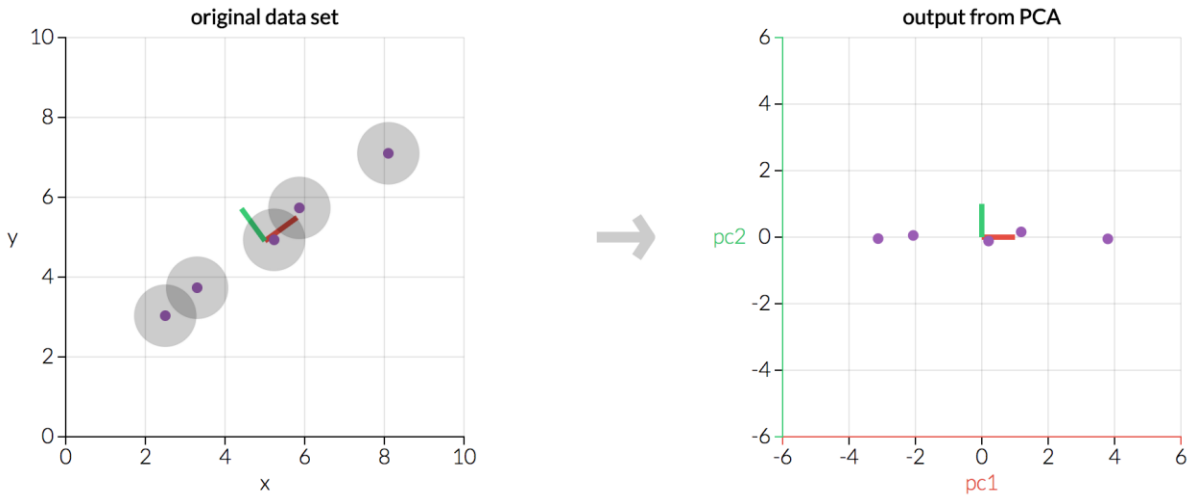


Diagram from http://setosa.io/ev/principal-component-analysis/

As we can see, the data points in the original data set are widely distributed, while the PCA's outputs' variance on y-axis is much smaller. In other words, PCA is a way to set up a new coordinate system that minimizes the variance to achieve reduced dimension of data set. The result we get from PCA are coefficients of all factors. A larger coefficient of a constant means a more important factors. When we apply PCA, we can decide how many degrees of components we want. For example, if the data pool are (weight, height) of people, the first component will be the "sizes" of different people and the second component means people's "body shapes". In our study, the first order component is students' general math ability and the second component can represent their tendency toward pure or applied math, but meanings behind components since the third order start to be vague.

It is certain that there will exist error if some of the data is missing, but the methods we apply can influence the result greatly. The five methods we apply are: disregarding missing data, filling in average, applying alternating least squares without filling in, imputation and probabilistic principal component analysis (PPCA). All of these five methods have their own strengths and weaknesses.

The first method is easy to understand. We ignore the missing data by replacing them as Nan on Matlab like: [1 Nan 1] and then apply PCA on the matrix. The second one is filling-in the mean values. The third one is alternating least square algorithm (ALS), which can predict every missing value based on the existed values. Next, imputation means instead of using original matrix of data, we keep finding a new coordinate system for the data and set up a new estimated matrix and repeating this process until the difference between the old and new matrix becomes small. PPCA is a new method found in recent years, and Alexander Ilin and Tapani Raiko recommend it when the number of missing data is large.

Our results show that ignoring missing data is the worst since the results we get is just an empty matrix. Filling in with mean value is the most efficient one and we can find higher order components by this method. ALS is not useful unfortunately. Even though we can use ALS to find high order components, we find out that it suffers from overfitting. For PPCA, we can only use this method to find first order component because it does not converge for two components.

In conclusion, the imputation algorithm produces the best 1st and 2nd components; the filling in method can be used if it's necessary to have more than 2 components. The ALS and PPCA methods perform poorly, though in different ways: one produces too many components, one too few.

Table of Contents

1. Introduction

The main purpose of this project is to use methods from linear algebra to find the most useful and relevant information. In the age of big data, the information is growing exponentially, people can feel overwhelmed by mountains of information. Thus, it is necessary to classify the information people need the most. The data pool for this study is 37 students' grades on 13 different math courses in Syracuse University, including MAT295 Calculus 1, MAT296 Calculus 2, MAT 331 First Course in Linear Algebra, MAT375 Introduction to Abstract Mathematics, MAT397 Calculus 3, MAT412 Introduction to Real Analysis 1, MAT414 Introduction to Ordinary Differential Equations, MAT517 Partial Differential Equations and Fourier Series, MAT521 Introduction to Probability, MAT525 Mathematical Statistics, MAT531 Second Course in Linear Algebra, MAT532 Applied Linear Algebra, MAT581 Numerical Methods with Programming. In this analysis, the first algorithm applied is Pagerank, according to which, the rank of a certain page i is weighted by the outlinks to the page i.

In my data example, there exist numerous outlinks between these 13 MAT courses. For instance, some of these courses are the prerequisite courses for some others in the curriculum. In PageRank, the prerequisite courses transport their rank into the course of the next level. Based on the relations (prerequisite) between different elements (courses), one can express these relations by a matrix, named A. The initial conditions of students, which means their current status of different courses, are also necessary: they are presented by vector v. If one multiplies A with v, then the rank of initial element will start transporting the rank between elements via links. Suppose the transport keep going forever, the ranks of different elements would tend to converge to an equilibrium. This equilibrium vector can offer the prediction to future action of different elements.

However, solely elements correlation is not enough for data study. Differentiating based on individuals is also necessary, since all students have different performances and talents on different area. This is also why the consideration of covariance is so crucial in this study. In order to study the covariance of elements, principal component analysis(PCA) is a good method. PCA was first introduced by Pearson back to 1901, who used this algorithm to visualize observed data and study the structure of the data set. It is able to represent the data by diagrams in a more efficient and more understandable way. PCA is part of orthogonal linear transformation. It can set up a new coordinate system that minimizes the variance of the data. I chose Matlab as the study's platform because PCA can be achieved easily on it.

The focus of this paper is on applying PCA algorithm when people encounter incomplete data set. In this study's data base, all of the thirty-nine students have some courses that they have not taken yet, so there are numerous missing data.

This paper proposes five potential ways to solve this problem. The first method is disregarding the missing data. It means that we can replace the missing data with NaN in Matlab. The second way is replacing the missing data with maximum likelihood estimation. The third way is applying alternating least square algorithm (ALS) to PCA on Matlab. ALS can complete the matrix without filling in the data by ourselves. Instead, ALS can predict every missing data. In the fourth method, we also need to use imputation. We predict the missing by ourselves and set up a new estimated PCA mapping matrix based on the estimated data set. Only choosing the first few dimensions we need, we can predict the data set backward. This algorithm repeats this process until the difference between the old and new matrix converges to 0. The last method is completing matrix with probabilistic model for PCA. This method can also be identified as looking for maximum likelihood estimator for parameters in expectation-maximization (EM) algorithm.

2. Pagerank Algorithm

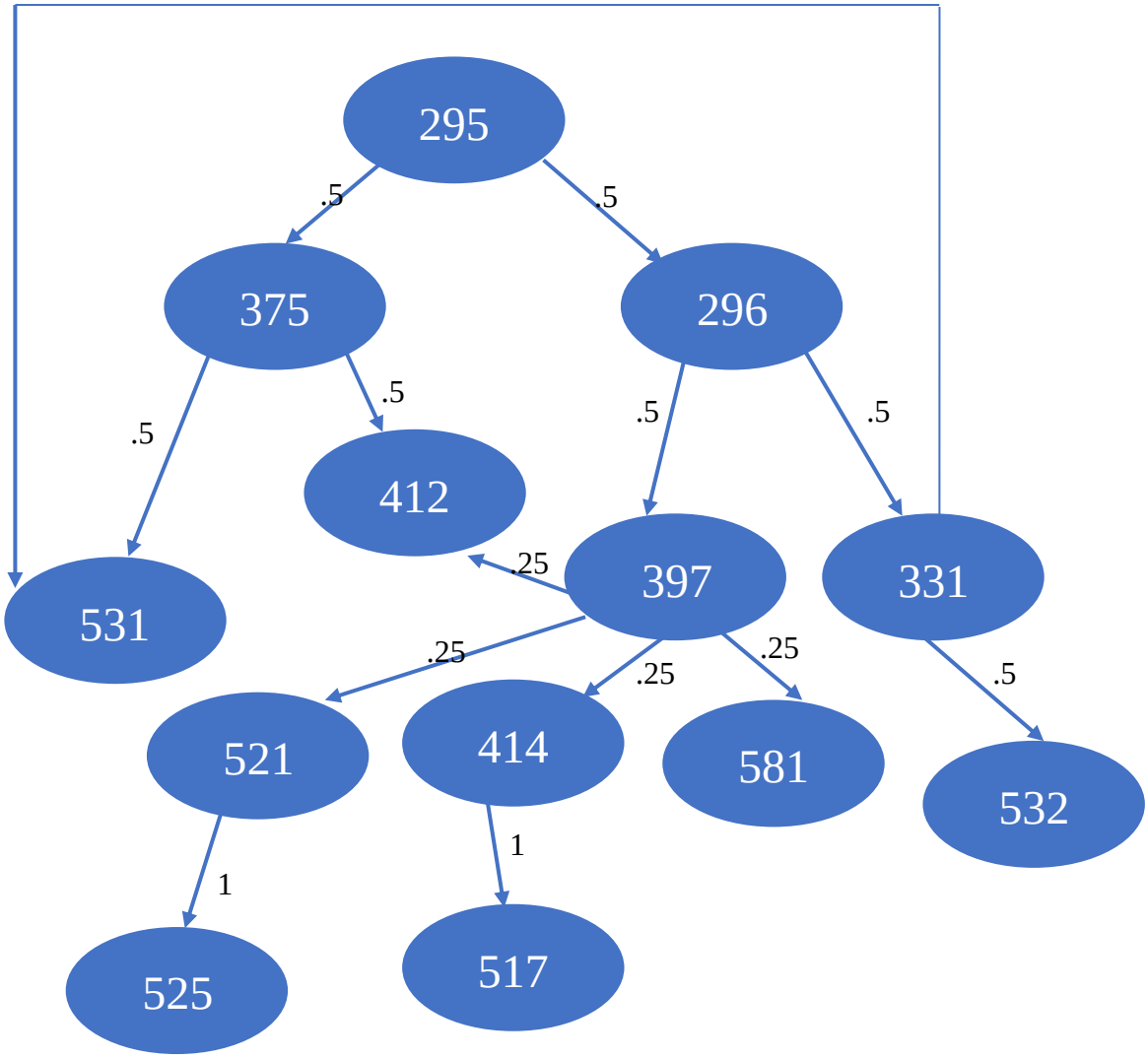
Pagerank Algorithm was originally applied on searching websites by Google. Each page has its outlinks and inlinks. For page i, we denote O_i as its outlinks to page i and I_i as its links from page i to other pages. The links between pages transport the rank between different pages. For example, if a highly ranked page has its outlinks to the pages of next level, it also transports rank into the next level. In other words, a rank of certain page can be calculated by summing up rank from the last level pages. If we denote number of outlinks to page i as N_i, we can convey the rank of page i as following:

$$r_i = \sum r_j / N_j \quad (1)$$

However, this method can be only applied to simple linear models. If the model of the data is recursive, then directly applying formula (1) is no longer valid. Instead, we need to setup a mapping matrix for the rank transport.

Based on the description of different courts, we can also convey the relationships into a diagram and a matrix as follows:

.5


295
.5
.5
375
296
.5
.5
.5
.5
412
531
.25
397
331
.25
.25
.25
.5
521
414
581
532
1
1
525
517

	295	296	331	375	397	412	414	517	521	525	531	532	581
A=295	[0	.5	0	.5	.5	0	0	0	0	0	0	0	0
296	0	0	.5	0	.5	0	0	0	0	0	0	0	0
331	0	0	0	0	0	0	0	0	0	0	.5	.5	0
375	0	0	0	0	0	.5	0	0	0	0	.5	0	0
397	0	0	0	0	0	.25	.25	0	.25	0	0	0	.25
412	0	0	0	0	0	0	0	0	0	0	0	0	0
414	0	0	0	0	0	0	0	1	0	0	0	0	0
517	0	0	0	0	0	0	0	0	0	0	0	0	0
521	0	0	0	0	0	0	0	0	0	1	0	0	0
525	0	0	0	0	0	0	0	0	0	0	0	0	0
531	0	0	0	0	0	0	0	0	0	0	0	0	0
532	0	0	0	0	0	0	0	0	0	0	0	0	0
581	0	0	0	0	0	0	0	0	0	0	0	0	0]

On the other hand, we still need information about initial condition of each student, which also means which courses that the student have taken or not. We can denote the initial condition of each person as v_i , and if we take each person's initial condition and multiply with the matrix, we can transport the rank of the initial condition to different level. Since there are too many students in our data sample, we are only going to use the first student as one example. The initial value of this student can be written as follows if we assume these initial importance is uniformly distributed.

```
v= [1/11 1/11 1/11 1/11 1/11 1/11 1/11 1/11 1/11 0 1/11 0 1/11]';
```

After transporting the rank for numerous of times, the result vector we get is going to converge. In other words, $\lim_{n\to\infty} Av^n$ converges and the final vectors we get can represent the rank of each courses. In Matlab, we can translate this method into following code:

```
v2=zeros(13,1);
for i = 1:10000
    v2=A*v;
    if norm(v2-v, 'fro') < 1e-10
        break
    end
    v=v2;
end
display(v)
```

Unfortunately, the result we get is 13*1 matrix with all entries equal to zero. Clearly, this result is not good. We believe that this phenomenon is because of the limited number of links between the courses, which causes the mapping matrix to contain too many zeros. Thus, Pagerank is not available in this study.

3. Principal Component Analysis

Principal Component Analysis is a powerful method to provide useful information of variance between different factors. PCA has two main steps. The first step is dimension reduction by subtracting the mean of each measurement type. Then we apply singular value decomposition (SVD) to the new matrix to find principal component coefficients. Based on the coefficients(C) and the new data matrix(A), we can also find the score(S) of the principal components with the formula S=A*C. With the help of principal component coefficients, one can learn which component is the most important. However, the main problem we are facing is that every student has not finished all these 13 courses. This means that almost half of the data is missing. Thus, the main goal here is to achieve PCA algorithm with those missing data. To deal with these problem, there are five potential methods that can solve it.

3.1 Disregard the Missing Data

The first method is the easiest one among those five by just disregarding those missing data and represent them by NaN on Matlab. In this way, we can represent our data as follows:

295	*296*	*331*	*375*	*397*	*412*	*414*	*517*	*521*	*525*	*531*	*532*	*581*
4	4	4	4	3.333	4	3.667	2.333	4	NaN	3.667	NaN	4
NaN	NaN	4	4	4	4	4	NaN	4	NaN	3.667	NaN	NaN
3.667	4	3.667	4	4	NaN	3.667	4	3.667	NaN	NaN	3.333	3.667
NaN	NaN	4	4	3	3	NaN	NaN	3.333	NaN	2.333	NaN	NaN
NaN	NaN	4	4	4	1	3.667	NaN	3.333	NaN	2.667	NaN	4
NaN	2.333	4	3.667	1.667	NaN	1.667	NaN	2	3	NaN	3	NaN
4	3.333	3.333	4	3.667	3	3	3	3.667	NaN	NaN	4	3
NaN	2	2.333	4	2.667	3	3.667	2.333	3.667	NaN	3.667	NaN	NaN
NaN	3.333	4	2	2.333	NaN	NaN	NaN	2.333	NaN	0	NaN	NaN
NaN	3.667	3.667	3.667	3.667	NaN	3.333	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	4	4	3.333	3.667	4	4	NaN	NaN	3.667	4
NaN	4	4	2.667	4	3	4	NaN	4	NaN	4	NaN	NaN
3	3.333	4	4	4	0	1.667	NaN	0	NaN	3.333	NaN	NaN
3.333	NaN	3	2.667	2.667	2.333	2	NaN	3.333	1	1	NaN	NaN
NaN	NaN	3	3.667	NaN	2	NaN	1	4	3	2	NaN	2.333
NaN	NaN	3.333	4	3.333	NaN	4	1	3.333	3.333	NaN	2.333	3.667
NaN	4	3.333	3.667	3.667	3	4	3.333	3.667	NaN	2.333	3.333	3
NaN	NaN	4	4	4	NaN	4	4	4	NaN	NaN	NaN	NaN
3	2.333	3	4	2.667	1	1.667	1	2	1	3.333	NaN	NaN
NaN	2.667	2	3.667	4	2.333	3.333	2.333	3.333	NaN	NaN	NaN	NaN
3.667	2.667	2.667	4	3.333	3.333	3.333	NaN	3	NaN	NaN	NaN	NaN
3	2	2.667	2.333	2	2	1.667	1	2.667	1	3.667	NaN	NaN
4	4	4	4	4	3.667	4	2.333	4	4	3.667	NaN	NaN
2.333	NaN	NaN	2.667	NaN	1.667	2.667	NaN	2	NaN	2	1	NaN
3.333	3	2.333	3.667	3.667	3	3	3.667	2.333	3.333	NaN	3.333	3
NaN	3	3.667	2.667	4	3	4	2.667	3	NaN	3.667	NaN	NaN
NaN	3	2.667	2	3	1	1	NaN	2.333	NaN	1	NaN	NaN
NaN	2.333	3.333	4	3.667	2.333	1.667	2	3.667	3	NaN	2.333	NaN
NaN	NaN	2.667	2	NaN	1	NaN	NaN	NaN	NaN	3	NaN	NaN
NaN	NaN	3.333	4	3.333	NaN	4	1	3.333	3.333	NaN	2.333	3.667
4	4	2	3.333	4	2	2	2	3	1	NaN	2.667	2
4	3	3.667	4	3.333	3	3	NaN	3.333	NaN	2.333	NaN	NaN
4	4	4	3.333	4	1.667	NaN	NaN	4	3.333	NaN	2.333	3.667
4	2.333	2	2.667	2	NaN	2.333	NaN	NaN	NaN	2.667	NaN	NaN
NaN	NaN	3.333	4	4	3	4	3.667	3	2.667	3.667	3.667	4
4	2	2.333	4	3.333	2	4	NaN	3.333	3	3.667	NaN	NaN
NaN	4	3.667	4	2.333	1.667	4	NaN	2.667	NaN	NaN	NaN	NaN

However, when we input PCA(A') on Matlab, the result is following:

```
C =13×0 empty double matrix

S =37×0 empty double matrix

L =[]
```

It is not hard to find that all the results we get are some empty matrices. It is because Matlab can only do PCA in a matrix with complete rows. While we replace all the missing data with NaN, Matlab deletes all the rows that have NaN. Unluckily, all the data samples in this study have missing data, and Matlab deletes all the rows, just leaving an empty matrix A.

However, ignoring missing data pairwise is another choice. In this method, one can choose two rows of data, one deletes all the columns with missing values in those rows. Based on the existed values, one can apply PCA in Matlab with imbedded code:

```
[C,S,L]= pca(A,'rows','pairwise')
```

The result from this method shows that this one is not actually a good choice. The results shows error as: "The estimated covariance matrix is not positive semi-definite". The estimated covariance matrix from pairwise deleting is not positive semi-definite while the true covariance matrix is always positive semi-definite.

3.2 Fill in Average

In the third method, we can fill in the missing data with each element's most likelihood estimation. In this study, mean value is the most reasonable and efficient estimators. It can be represented on Matlab in the following way:

```
M=nanmean(A);

Ac = A-ones(size(A,1),1)*nanmean(A);

M2=nanmean(Ac);

Ac(isnan(Ac))=0;

[C,S,L]= pca(Ac);

labels=[295
296
331
375
397
412
414
517
521
525
531
532
581]';

disp('Filled in by means')
for k = 1:3
  [W, I]= sort(C(:,k), 'descend');
  fprintf('%d\t', labels(I))
  fprintf('\n')
  fprintf('%.2f\t', W')
  fprintf('\n\n')
end
```

The result we get are:

Filled in by means

414	412	521	397	531	375	517	525	296	331	295	532	581
0.52	0.44	0.40	0.26	0.24	0.24	0.24	0.22	0.17	0.16	0.12	0.10	0.07
531	375	412	521	295	414	581	532	525	397	331	296	517

0.60	0.20	0.16	0.13	0.02	-0.01	-0.03	-0.04	-0.10	-0.17	-0.32	-0.40	-0.51
412	521	295	532	517	414	581	296	525	331	375	397	531
0.49	0.44	0.12	0.07	-0.01	-0.06	-0.10	-0.10	-0.15	-0.27	-0.32	-0.32	-0.47

In the result, we can get three degree of components and we rank variables based on the coefficients. If the coefficient is larger, then it means that this certain variable is more important. For instance, the first row of results represents the constant of computing student's general math ability. As it is shown, MAT414 has the greatest coefficient while MAT581has the smallest, which indicates that when people are looking for student's overall math ability, they should look for the student's MAT414 grade first. However, MAT581 is not that important in weighing student's math ability, compared to other courses. The second row can mean something different than the first row. While the first row represents the overall ability, the second row's interpretation is not determined. One needs to look into these courses feature and decide what these constants decide. After our observation, I hold the opinion that the second row can reflect student's tendency to pure or applied math study. We find that if the coefficient of the course is larger, then it means that the course is tending to be a pure math course, such as MAT531, but MAT517 tends more to be applied because it has a smallest coefficient.

Unfortunately, the meaning of the third rows is hard to define, and in this study, it is also hard to interpret the meaning behind the third degree components.

3.3 Applying Alternating Least Square

In the third method, we are no longer plugging in numbers by ourselves. Appling ALS algorithm can help us to fix the unknown data and try to minimize the mean square error(MSE). ALS can be realized easily in Matlab as following:

```
A=A';

M=nanmean(A);

Ac = A-ones(size(A,1),1)*nanmean(A);

M2=nanmean(Ac);

Ac(isnan(Ac))=0;

[C,S,L]= pca(A,'algorithm','als');

labels=[295
296
331
375
397
412
414
517
521
525
531
532
581]';

[C,S,L]= pca(A,'algorithm','als');

disp('ALS algorithm, without filling in')

for k = 1:3
  [W, I]= sort(C(:,k), 'descend');
  fprintf('%d\t', labels(I))
  fprintf('\n')
  fprintf('%.2f\t', W')
  fprintf('\n\n')
end
```

The result are following:

ALS algorithm, without filling in

525	532	517	375	414	531	412	296	521	397	331	295	581
0.94	0.19	0.12	0.10	0.08	0.08	0.04	0.03	0.03	0.02	0.00	-0.03	-0.22

517	412	581	414	532	397	531	521	296	375	331	295	525
0.50	0.38	0.37	0.36	0.29	0.28	0.27	0.19	0.17	0.13	0.07	-0.03	-0.14

531	295	517	412	521	532	397	296	414	525	375	331	581
0.60	0.24	0.17	0.15	0.06	-0.02	-0.10	-0.11	-0.13	-0.15	-0.16	-0.46	-0.4

As we can see, the result we get by ALS algorithm is different with the one by filling in by average. In this result, MAT525 seems to be the most important factor to measure students' overall math

abilities. Concerning the second degree of components, we can find that MAT295 has the greatest coefficient and 525 has the smallest one.

However, we can find this ALS algorithm does not make sense in our project, since there are negative coefficients in the first degree of components. For instance, if students do well in MAT295, it will lowers his or her over all math ability.

3.4 Imputation

In the fourth method, we first replace all the missing data with their mean, and then we do PCA to find its new coordinates system, C and S. This method can build up a new data set matrix, only applying first few rows of the new coordinate system. In this way, we can avoid overfitting. Then, we input the new matrix to the beginning and repeat the PCA process until the difference between new matrix and former one becomes small enough.

The complete process is as following:

```
A=A';
G=A;

missing = isnan(G);
M = ones(size(G, 1), 1) * nanmean(G, 1);
G(missing) = 0;
G = missing.*M + (~missing).*G;
k = 2;  % or 2 or 3 or 4

for i = 1:10000   % Some large number. This is how many steps we allow, to prevent infinite loop.
    M = ones(size(G, 1), 1) * mean(G, 1);
    [C, S] = pca(G);
    S = S(:, 1:k);
    C = C(:, 1:k);
    G2 = missing.*(S*C' + M) + (~missing).*G;
    if norm(G2-G, 'fro') < 1e-6    % or some other small number. If difference is small, end. The
difference is measured by Frobenius norm, because it is easy to compute.
        break
    end
    G = G2;
end

if i >= 10000
    disp('Conjugate: Failed to converge');
else
    fprintf('Imputation algorithm with k = %d\n', k)
    for j = 1:k
        [W, I]= sort(C(:,j), 'descend');
        fprintf('%d\t', labels(I))
        fprintf('\n')
        fprintf('%.2f\t', W')
```

```
        fprintf('\n\n')
    end
end
```

The second degree of component is the highest one we can get by this method and the result are:

Imputation algorithm with k = 2

581	525	414	412	532	521	517	397	375	531	296	295	331
0.47	0.45	0.34	0.32	0.31	0.27	0.23	0.18	0.17	0.16	0.14	0.14	0.10

531	525	412	375	414	581	295	521	397	532	331	296	517
0.63	0.16	0.13	0.10	0.07	0.05	0.01	-0.10	-0.11	-0.11	-0.16	-0.40	-0.57

In this case, we can only take k as two, since it fails to converge when we take k=3.

3.5 Probabilistic PCA

Probabilistic PCA is another available method by Alexander Ilin and Tapani Raiko. Concerning Gaussian noise in this model, they recommend it when the number of missing data is large, which suits our situation perfectly. In their paper " Principal Component Analysis with Missing Values", they suggest that EM algorithm can be replaced by a new set of formulas, and we translate it into Matlab as following:

```
Y=A;
c = 1;
[d, n] = size(Y);
v = 1;
x = zeros(c, n);
m = zeros(d, 1);
w = eye(13, c);

while 1

    oldw=w;

    % E step: formulas (17)

    sigma=zeros(c);
    for j=1:n
        sigma(:,:,j)=v*eye(c);
        for i = 1:d
            if ~isnan(Y(i,j))
                sigma(:,:,j)=sigma(:,:,j)+w(i,:)'*w(i,:);
            end
```

```
        end
        sigma(:,:,j)=v*inv(sigma(:,:,j));
    end

    for j=1:n
        x(:,j)=0;
        for i = 1:d
            if ~isnan(Y(i,j))
                x(:,j)=x(:,j)+w(i,:)'*(Y(i,j)-m(i));
            end
        end
        x(:,j)= sigma(:,:,j)*x(:,j)/v;
    end

    for i = 1:d
        m(i)=0;
        count=0;
        for j = 1:n
            if ~isnan(Y(i,j))
                m(i)=m(i)+Y(i,j)-w(i,:)*x(:,j);
                count=count+1;
            end
        end
        m(i)=m(i)/count;
    end

    % M step: formulas (18)

    a=0;
    b=0;
    for i = 1:d
        for j = 1:n
            if ~isnan(Y(i,j))
                a=a+x(:,j)*x(:,j)'+sigma(:,:,j);
                b=b+x(:,j)*(Y(i,j)-m(i));
            end
        end
        w(i, :)= (inv(a)*b)';
    end

    v=0;
    count=0;
    for i = 1:d
        for j=1:n
            if ~isnan(Y(i,j))
                v= (Y(i,j)-w(i,:)*x(:,j)-m(i))^2+w(i,:)*sigma(:,:,j)*w(i,:)';
                count=count+1;
            end
        end
        v=v/count;
    end


    if norm(w-oldw)<.001
        break
    end
end

disp('PPCA')
[W, I]= sort(w(:,c), 'descend');
fprintf('%d\t', labels(I))
fprintf('\n')
fprintf('%.4f\t', W'/norm(w))
fprintf('\n\n')
```

And the result of PPCA are following:

PPCA

When c=1:

581 532 531 525 521 517 414 412 397 375 296 295 331

0.3319 0.3296 0.3268 0.3259 0.3154 0.3109 0.2940 0.2610 0.2256 0.2118 0.2111 0.1984 0.1943

However, when we set c=2 or higher values, this method fails to converge, so unfortunately, we can only get the first component based on PPCA.

4. Discussion

We conclude all the results above in the following two forms:

First PCA component

Method / Courses	Filling in mean value	Imputation	PPCA	ALS	Filling in NaN
295	0.12	0.13	0.1984	-0.03	
296	0.17	0.09	0.2111	0.03	
331	0.16	0.08	0.1943	0.00	
375	0.24	0.16	0.2118	0.10	
397	0.26	0.16	0.2256	0.02	
412	0.44	0.32	0.2610	0.04	
414	0.52	0.31	0.2940	0.08	
517	0.24	0.43	0.3109	0.12	
521	0.4	0.22	0.3154	0.03	
525	0.22	0.48	0.3259	0.94	
531	0.24	0.22	0.3268	0.08	
532	0.1	0.33	0.3296	0.19	
581	0.07	0.31	0.3319	-0.22	

Second PCA component

Methods / Courses	Filling in mean value	Imputation	ALS
295	0.02	0.01	-0.03
296	-0.40	-0.40	0.17
331	-0.32	-0.16	0.07
375	0.2	0.10	0.13
397	-0.17	-0.11	0.28
412	0.16	0.13	0.38
414	-0.01	0.07	0.36
517	-0.51	-0.57	0.50
521	0.13	-0.10	0.19
525	-0.10	0.16	-0.14
531	0.6	0.63	0.27
532	-0.04	-0.11	0.29
581	-0.03	0.05	0.37

Based on the results we get from those five different potential methods, we find that every method has its strength and weakness. However, it is clear that ignoring the missing data by replacing them as Nan is not a good choice on the platform of Matlab, since Matlab will delete every row which contains Nan in the matrix. On the other hand, Filling in the mean value is an easy method to interpret and it can produce three degrees of meaningful components, while the components from the third degree are hard to interpret. Applying ALS without filling in is not a

wise choice even though it is a method that produces the highest degree of components since applying ALS suffers from overfitting. In our study, there are 13 courses. This means that the matrix C mapping the components to data has 13^2 = 169 entries, but we only have 158 observed data in our data set. When ALS try to weigh every value of different component, the result can not be informative. This is also the reason we get negative number in first degree of component. Probabilistic principal component analysis is efficient to produce the first degree of component but it is unable to produce higher degrees. Imputation is a method that can produce two degree of components. On the other hand, if we compare the order of components we get, we can find out that imputation share similar first order component coefficient order with PPCA, and its second degree coefficient order is similar to the ones of filling in with mean values.

5. Conclusion

In conclusion, Pagerank algorithm cannot be applied to the data set due to limited links between all the courses. However, we can say that imputation is our first choice dealing with missing data, and the filling in method can be used if it's necessary to have more than 2 components. The ALS method provides useless information due to overfitting, though it provides highest possible degrees of component. For PPCA, it is also not good enough because it can only provide first order degree of component analysis. Last but not least, ignoring missing data cannot be applied to analyze this data set.

According to the result from imputation, we can represent students' general capability in the formula as:

C=0.47*581grade+0.45*525grade+0.34*414grade+0.32*412grade+0.31*532grade+0.27*521grade+0.23*517grade+0.18*397grade+0.17*375grade+0.16*531grade+0.14*296grade+0.14*295grade+0.10*331grade

and based on the following formula we can analyze each student's tendency toward pure or applied math:

T= 0.63*531grade+0.16*525 grade+0.13*412 grade+0.10*375 grade+0.07*414 grade+0.05*581 grade+0.01*295 grade-0.10*521 grade-0.11*397 grade-0.11*532 grade-0.16*331 grade-0.40*296 grade-0.57*517 grade

A larger T means greater tendency toward pure math and smaller one represents the tendency to applied math.

Reference

Lars Eldén. (2007).“Matrix Methods in Data Mining and Pattern Recognition”. Society for Industrial and Applied Mathematics. Philadelphia, PA

Morris H. DeGroot & Mark J. Schervish (2011), “Probability and Statistics (fourth edition)” Pearson. New York City, NY

Mark A. Pinsky & Sameul Karlin (2011), “An Introduction to Stochastic Modeling (fourth edition)” Elsevier. 30 Corporate Drive, Suite 400, Burlington, MA 01803, USA

Henning Risvik (2007), “ Principal Component Analysis (PCA) & NIPALS Algorithm”

Stéphane Dray, Julie Josse (2014), “Principal Component Analysis with missing values: a comparative survey of methods”

Alexander Ilin, Tapani Raiko (2010), “Practical Approaches to Principal Component Analysis in Presence of Missing Values”

http://www.math.cornell.edu/~mec/Winter2009/RalucaRemus/Lecture3/lecture3.html