Principle Component Analysis

Qiuping Xu

Department of Math, FSU

1 Introduction

Principle component analysis is a well defined and popular baseline technique in dimensional reduction, factor analysis and many other applications. The linear PCA has already been extended into multilinear PCA, weighted PCA and kernel PCA. Here I will only describe the traditional linear PCA.

1.1 Background

Mathematically speaking, PCA is the procedure of finding a new coordinate systems, and the axis is ordered by the variance of the projection of the original data onto that axis. Suppose we generate a dataset with 200 samples, each has dimension 2;

```
theta=1;
X=[rand(200,1),0.1*rand(200,1)]*[cos(theta),...
-sin(theta); sin(theta), cos(theta)];
figure;
scatter(X(:,1),X(:,2),20,'ro','filled')
axis equal
```

And the data looks like the picture in Fig.1.

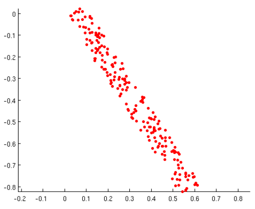

Fig. 1. Generated 200 samples

If you are a matlab user, then the matlab has built in function for PCA PCA-Matlab. The command looks like: [COEFF,SCORE,latent] = princomp(X), Here X is the input matrix in $n * k$, where the n is the size of datasets (200 in our case), and k is the dimension for each observation (2 in our case). The first output COEFF, each **column** defines the one principle direction(PC), and the columns are ordered by the variance of the projection on that direction. Since this define a new coordinate system, each column has norm 1, and arbitrary two columns are orthogonal to each other. The second output SCORE, each **column** shows the projection of the original data to the corresponding principle direction. The third output latent is the variance of each column of SCORE.

After transform the axes, now the data looks like Fig.2. The first axis ($[0.5480, -0.8365$
capture the most variance and then the second.

```
figure;
scatter(SCORE(:,1),SCORE(:,2),20,'ro','filled')
axis equal
```

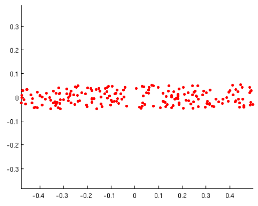

Fig. 2. PC Projections

Then we look at how much the variance is captured by each PC, in my case is 0.0826 for the first PC and 0.0008 for the second. In many research paper, you will see them in percentages, so we can divide each of them by the sum of latent. Then 99.03% and 0.97% of the variance are explained by the two PCs respectively.

```
latent./sum(latent);
```

If we have the assumption that in our dataset 1% of variance is cost by noise, we can use the the first column of the SCORE as lower dimensional representation of our data.

There are many area we need to visualize the variance along certain PC in the original data. Suppose we want to see what is look like if we change along first PC by 1 standard deviation. See Fig.3

```
k=1;% the number of PC
test=mean(X)+ 1*sqrt(latent(k))*COEFF(:,k)';
figure;
scatter(X(:,1),X(:,2),20,'ro','filled')
hold on
scatter(test(1),test(2),60,'ko','filled')
axis equal
```

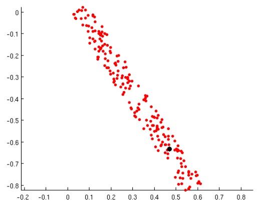

Fig. 3. Original data with change along 1st PC(black dot)

However, there is no determinate way to define the positive direction of each PC, that is the reason why people always plot ± certain standard deviation at the same time.

If we are doing certain classification problem and want to map new samples to the existing PC space. We need to center the new sample by subtracting the mean of the original data and then map to the PCs Fig.4.

```
new_sample=[0.5*rand(10,1),0.1*rand(10,1)]*[cos(theta), ...
-sin(theta); sin(theta), cos(theta)];
new_sample_centered=new_sample-repmat(mean(X),size(new_sample,1),1);
new_sample_mapped=new_sample_centered*COEFF;
figure;
scatter(SCORE(:,1),SCORE(:,2),20,'ro','filled')
hold on
scatter(new_sample_mapped(:,1),new_sample_mapped(:,2),60,'bo','filled')
axis equal
```

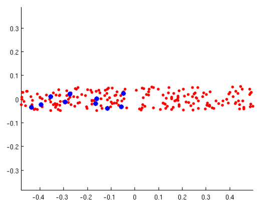

Fig. 4. Original scores and new mapped data(blue dot)

2 Alternative way of Computation

Traditional we compute $X^T X$ and do singular value decomposition, for small sample size N with large feature K (this happens a lot in shape/image and bioinformatics analysis), then this will be a large matrix, the dimension is the number of feature K. An alternative way is to compute XX^T, because the matrices XX^T and X^TX share the same nonzero eigenvalues, and if u_i is the eigenvector of X^TX, v_i is the eigenvector of XX^T, and they both correspondent to the eigenvalue σ_i^2. One can check these eigenvectors are related in identity $u_i = X^T v_i$.

For the example below, with 20 samples and 10000 features. The built in function 'Elapsed time is 56.624514 seconds', while the alternative method 'Elapsed time is 0.083536 seconds'.

```
X=[rand(20,10000),0.1*rand(20,10000)];
tic
[COEFF,SCORE,latent] = princomp(X);
toc
tic
X_center=X-repmat(mean(X), size(X,1),1);
[S V D]=svd(X_center*X_center');
COEFF1=X_center'*S./repmat(sqrt(diag(V))',size(X,2),1);
SCORE1=X_center*COEFF1;
latent1=diag(cov(SCORE1));
toc
```