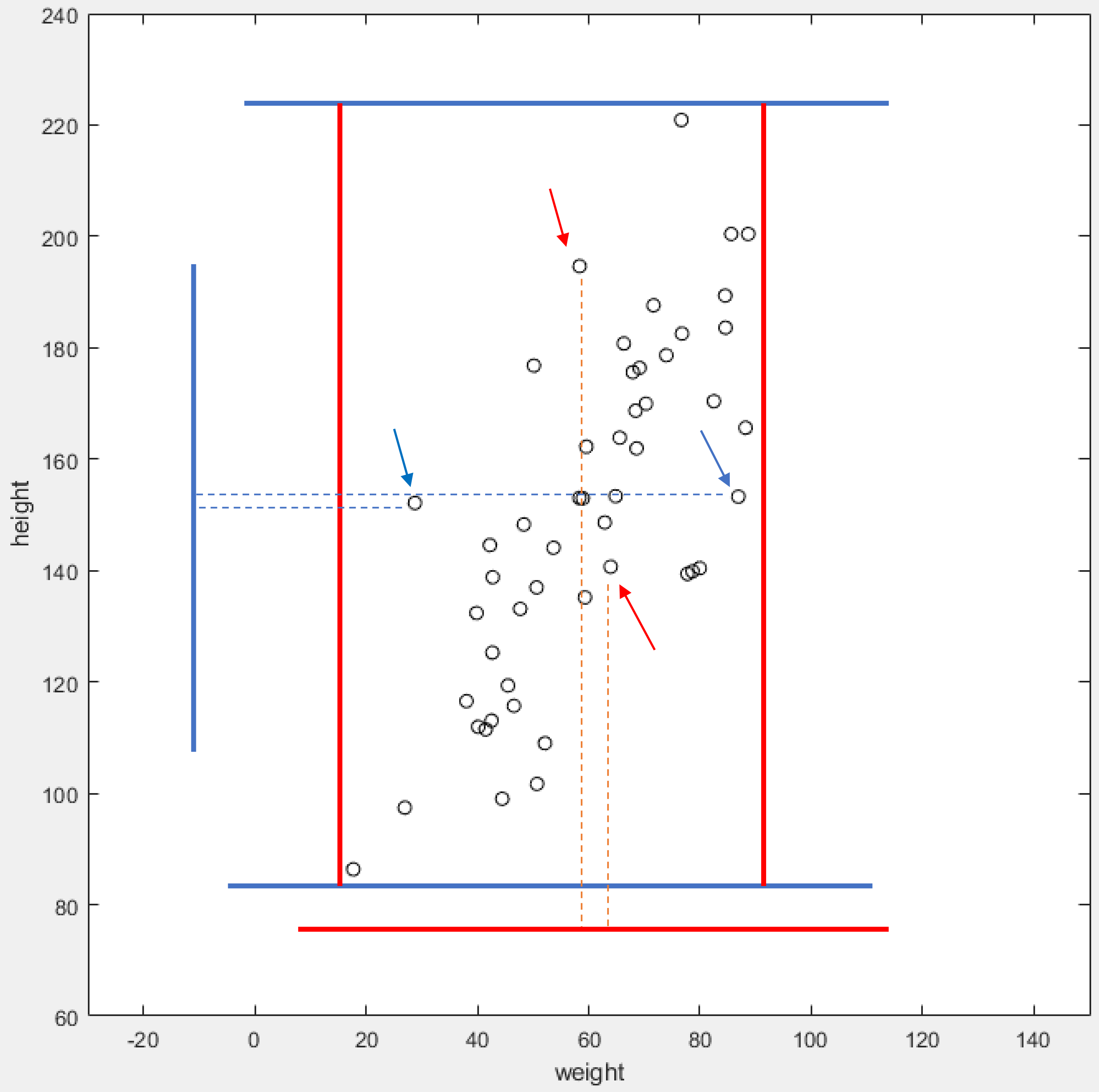
Principal Component Analysis

Halil Çağrı BİLGİ - 2231470

Muhammed Yusuf AKSEL - 2231140

- Let us say we have 50 person and 2 features of them,
- Weight and Height.
- We want to reduce this two-dimensional dataset into a single variable(dimension).
- If we choose x(weight) or y(height) directly, we lose a considerable amount of variance information.
- These fundamental choices of axis are not the best.
- So, we should find/choose the one axis that will maximize the variance information conserved.
- This is where PCA plays its role!



- We start by preprocessing the data, compute the mean and subtract it from all data points so that the data set is centered around origin.

- `% Zero mean`
- `mu = mean(X);`
- `X_zeromean = X - mu;`

- Now, we need to calculate the principal components of this data set. For this, a method called Singular Value Decomposition is used.

- `% SVD`
- `Cov_mtx = (1/m) .* X_zeromean' * X_zeromean;`
- `[U, S, V] = svd(Cov_mtx); % Principal components`

- Notice that the covariance matrix is Hermitian Symmetric.
- This means when we apply Singular Value Decomposition to covariance matrix what we actually find is the Spectral Decomposition. The singular values (s) correspond to eigenvalues and the left singular vectors correspond to the eigenvectors of the covariance matrix.

$$\mathbf{A} = \mathbf{U}\Lambda\mathbf{U}^H = \begin{bmatrix}\mathbf{u}_1\mathbf{u}_2\ldots\mathbf{u}_n\end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \ldots & \lambda_n \end{bmatrix} \begin{bmatrix} \mathbf{u}_1^H \\ \mathbf{u}_2^H \\ \vdots \\ \mathbf{u}_n^H \end{bmatrix}$$

What do we want?

- Higher variance along the direction
- Lower reproduction error, so as close as possible projected points to the original ones.

And these two occur at the same time?? (on the violet line)

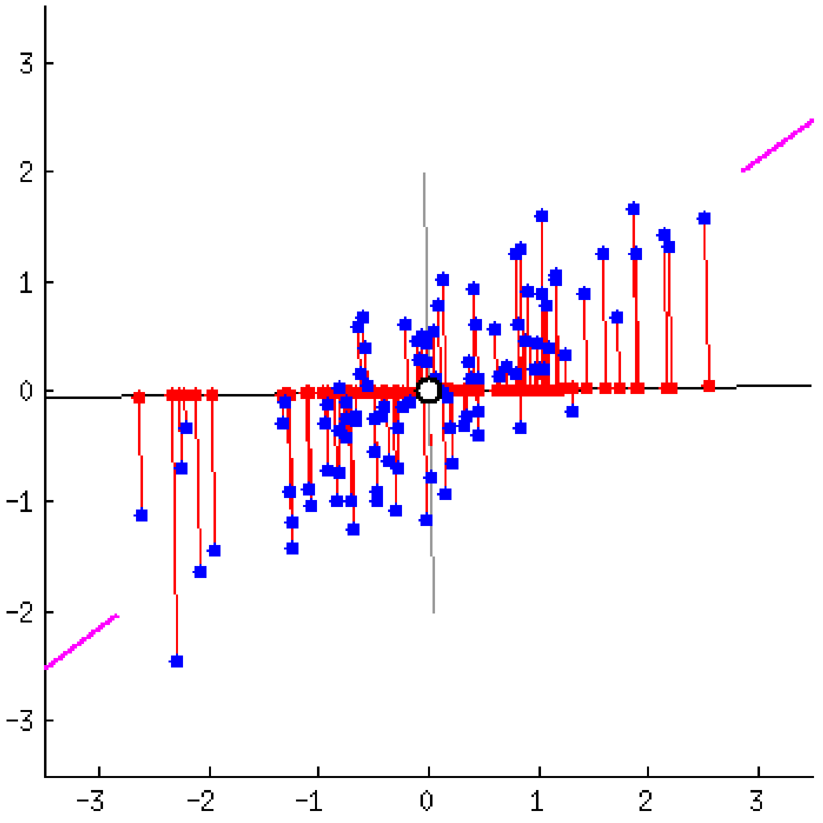
Notice that the red lines (proportional to the error in reproduction) are always perpenicular to the black line because these red lines are the closest distance of each point to the 1D black line.

We want to minimize the square of the closest distance of each point to the 1D black line. And also we don't want to lose the variance information so that at the same time we are maximizing the red dot to center distance.

Notice when the red dots(the projections) gets most seperated from each other, it is just the same moment of time when red lines are the smallest.

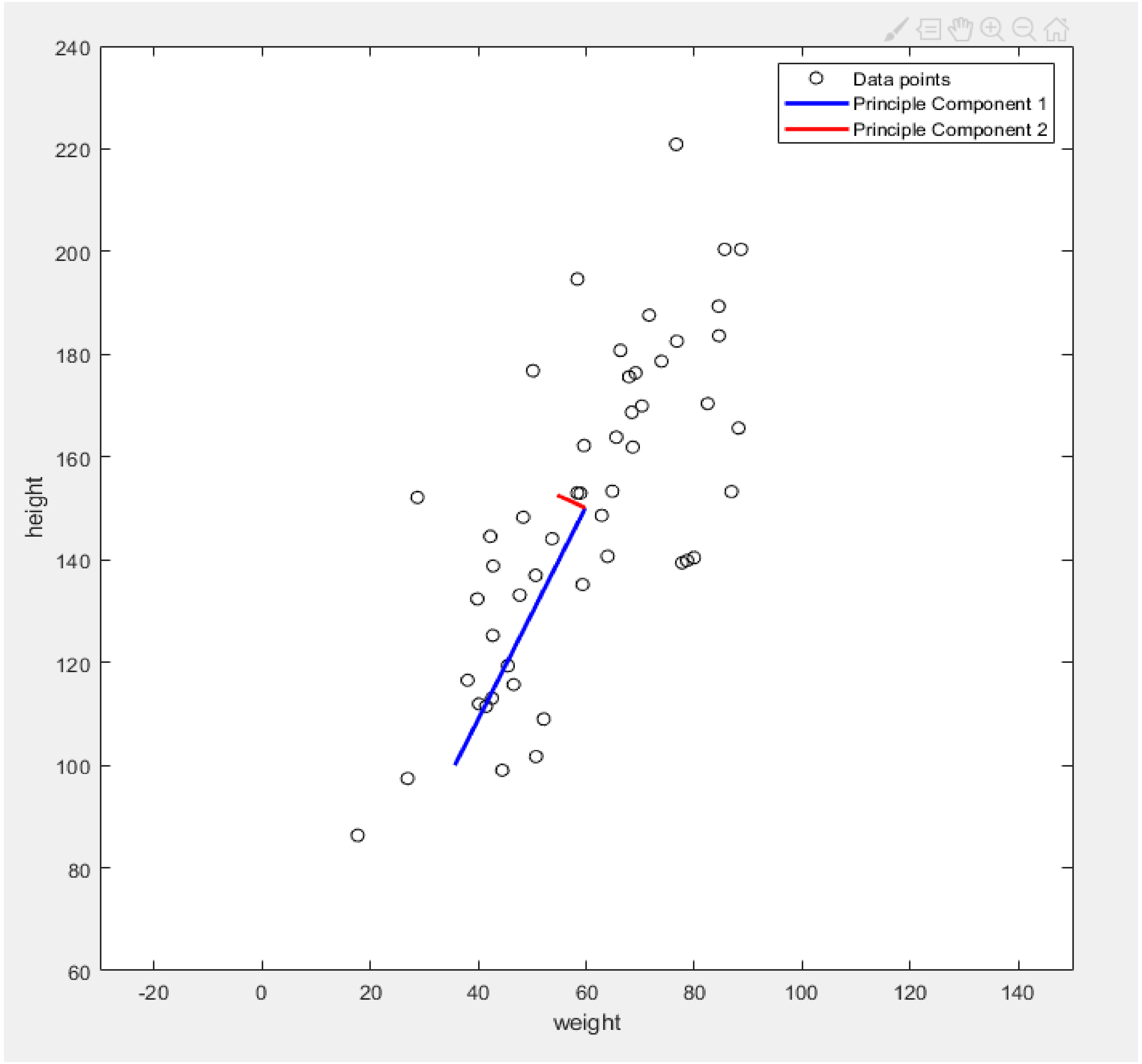
So singular value decomposition method finds this direction(the purple line) for the first principle component.

Reference : https://stats.stackexchange.com/a/140579



- So, each column vector of the U matrix is the principal component direction (eigenvectors), and S is a diagonal matrix and its diagonal entries are the eigenvalues. We will now draw the principal components found on top of the data set.

```
%  Draw the eigenvectors centered at mean of data. These lines show the
%  directions of maximum variations in the dataset.
hold on;
drawLine(mu, mu +  .05*S(1,1) * U(:,1)', '-b', 'LineWidth', 2);
drawLine(mu, mu +  .05*S(2,2) * U(:,2)', '-r', 'LineWidth', 2);
legend('Data points', 'Principal Component 1', 'Principal Component 2');
hold off;
```

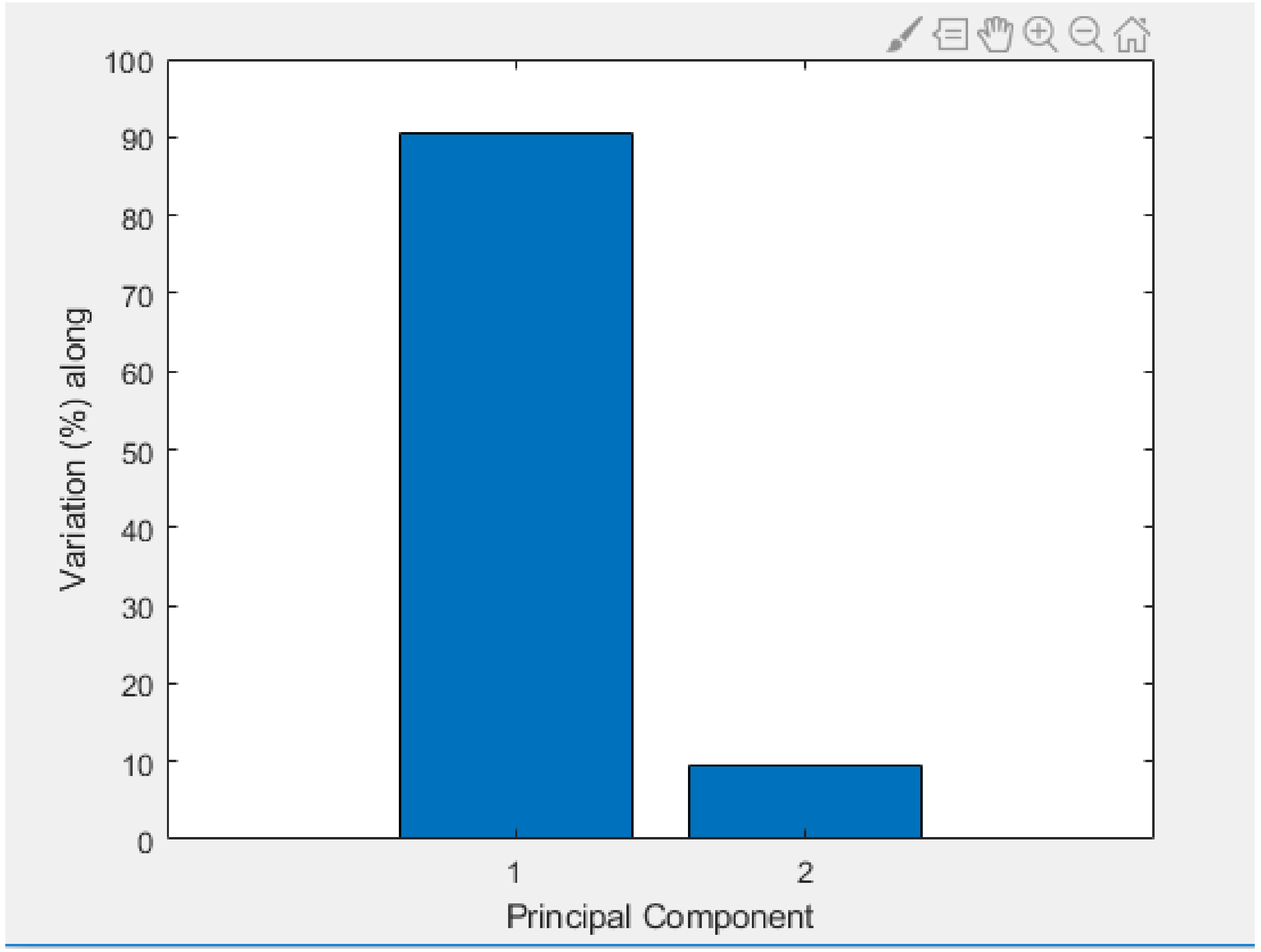


Data points
Principle Component 1
Principle Component 2
height
weight
240
220
200
180
160
140
120
100
80
60
-20
0
20
40
60
80
100
120
140

Let us see how much of the portions of variation captured by each principal component

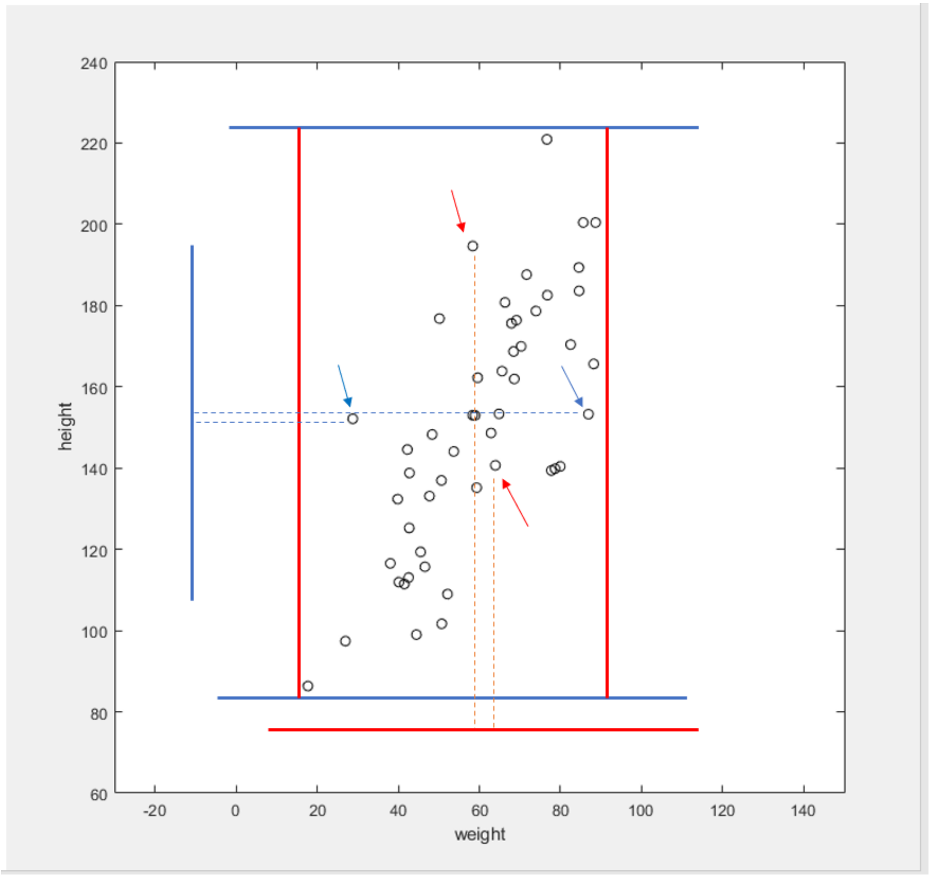
- % How much of the variance is captured by these principal components?
- % S has the variance associated with that eigenvector in its diagonal
- % entries.
-
- total_var = trace(S); % get the total variance associated with the data set
- percent_var = 100 * [S(1, 1) ./ total_var, S(2, 2) ./ total_var];
- figure;
- bar(percent_var);
- xlabel('Principal Component');
- ylabel('Variation (%) along');

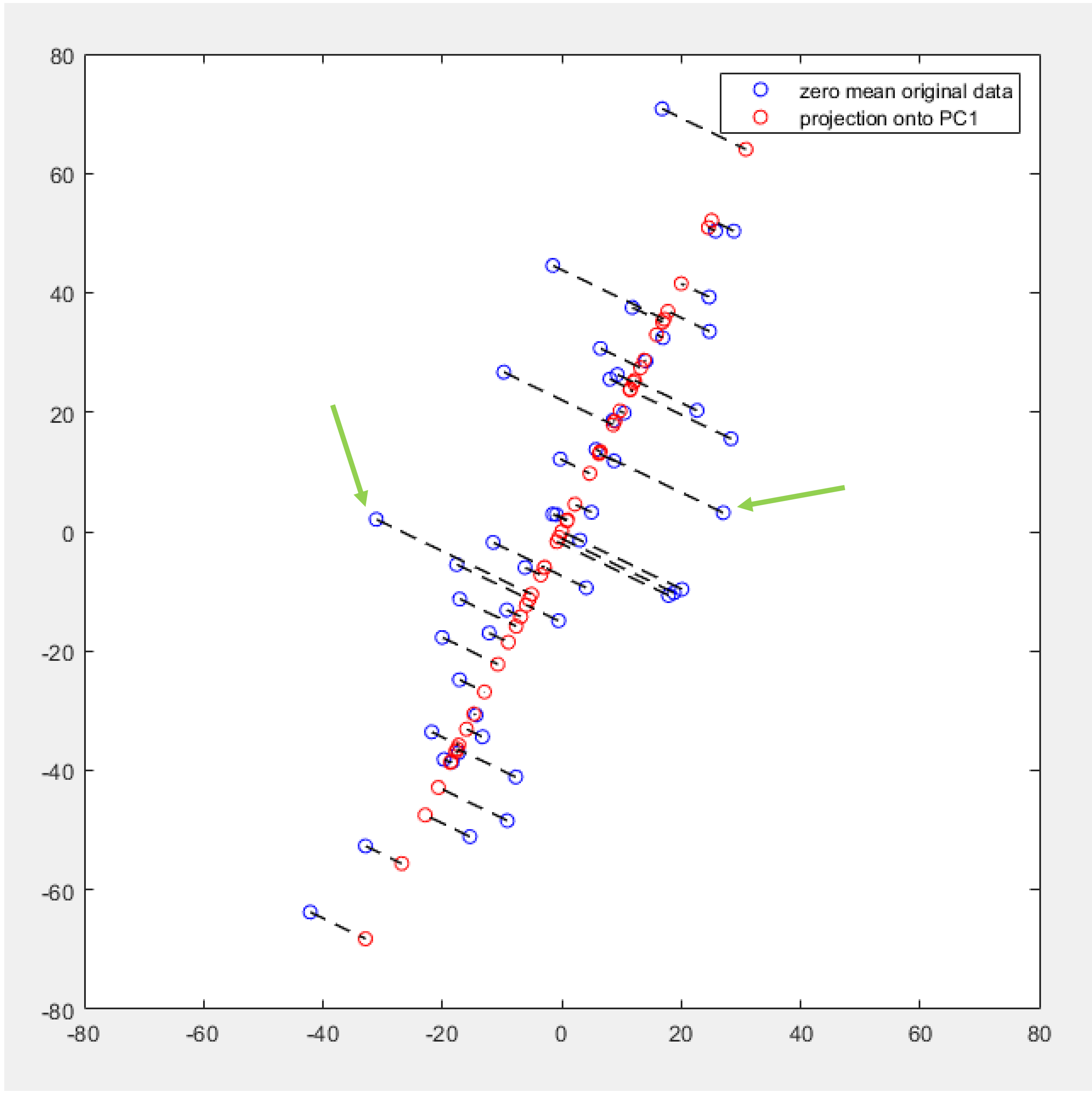
- So, we can see that almost 90 % of the variance is associated with the first principal component found. This is a good indicator that we can safely omit the second principal component and make a reduction from 2D space into 1D space by conserving a considerable amount of variation information.



```
• % draw the zero mean version of the original data first
• figure;
• plot(X_zeromean(:, 1), X_zeromean(:, 2), 'bo');
• axis([-80 80 -80 80]); axis square; hold on;

• % Project the data onto first K P.C.
• K = 1;         % Will reduce into K dimensional space
• U_reduced = U(:, 1:K);
• Z = X_zeromean * U_reduced;

• % Recover the approximation data back.
• % Which means we are projecting the data points on the line
• % which is essentially the first principal component.
• X_rec = Z * U_reduced'; % span Z onto the space spanned by first K P.C.


• plot(X_rec(:, 1), X_rec(:, 2), 'ro');
• for i = 1:size(X_zeromean, 1)
•     drawLine(X_zeromean(i,:), X_rec(i,:), '--k', 'LineWidth', 1);
• end

• hold off
```

- As seen, we projected all data points onto principal component 1.
- We have the highest variation along one direction that we could get.
- Remembered those two points pointed by green arrows?

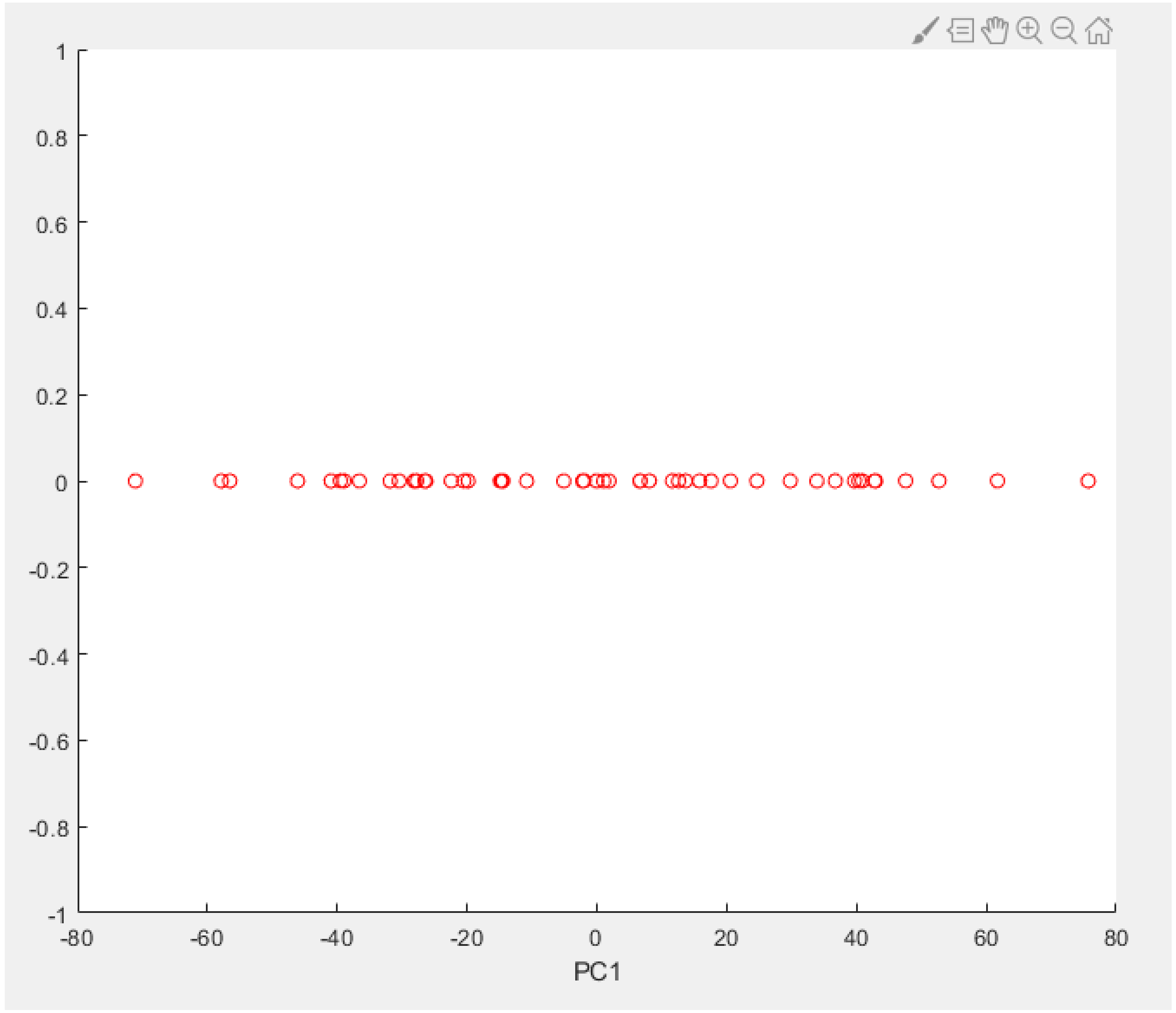




- Finally, plot this result on a more 1D looking view.

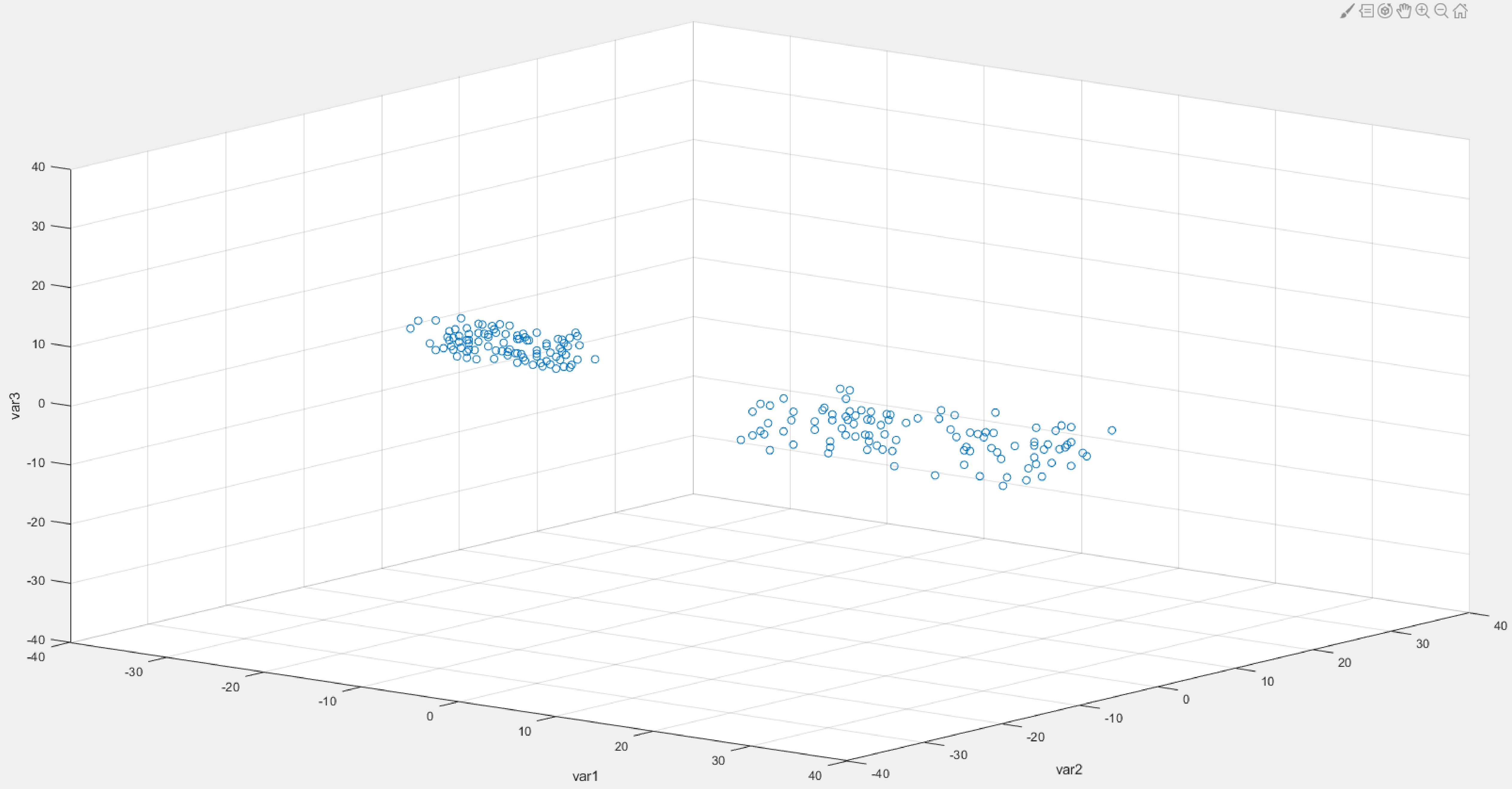
- `% plot the 1D reduced points on PC1`

- `figure;`
- `scatter(Z, zeros(1, length(Z)), 'ro');`



An example of reduction from 3D space to 2D space

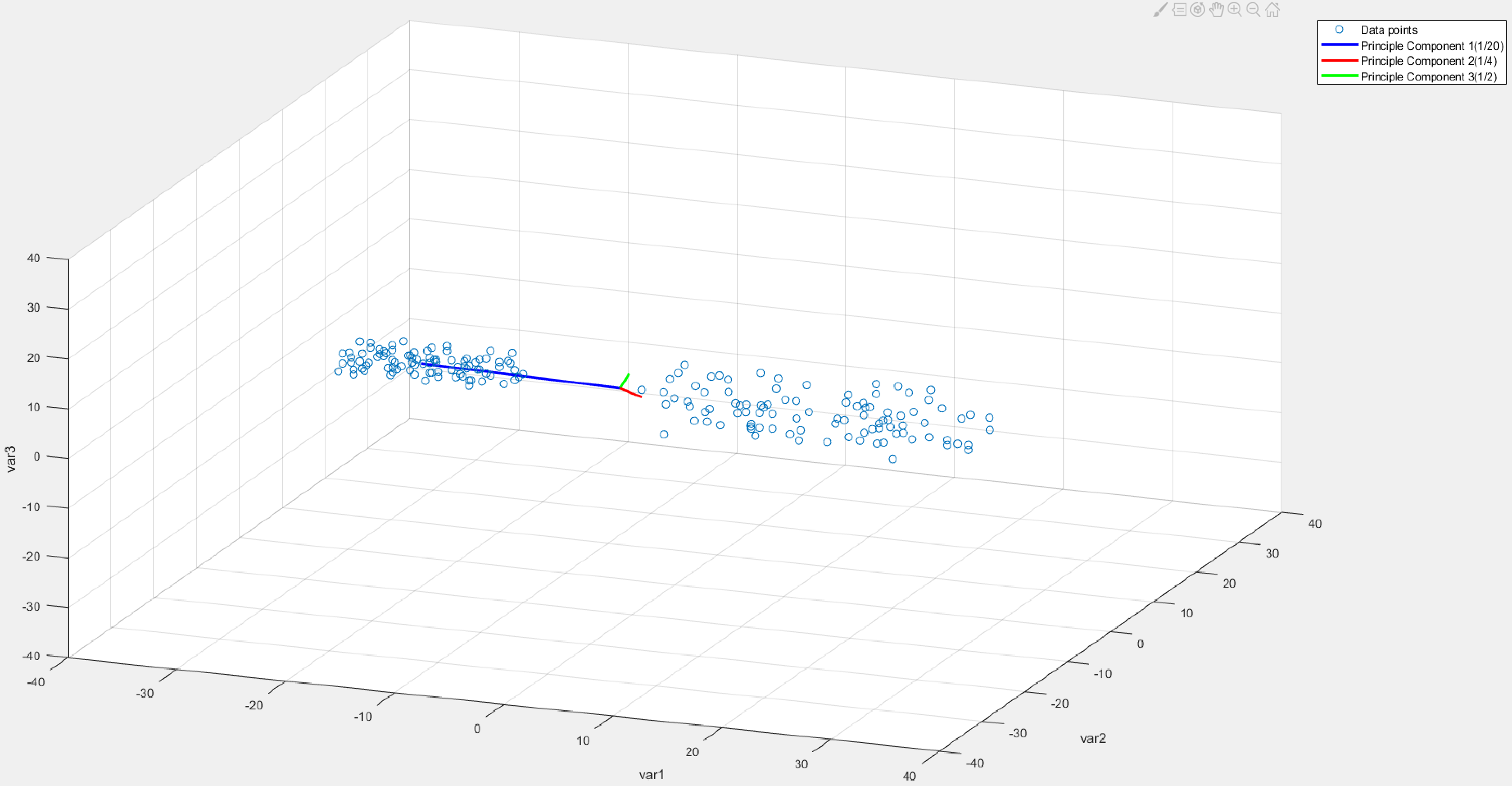
- We have a data set with 3 features(variables), namely as var1, var2, and var3
- We have 200 randomly distributed data points on two regions(clusters).



var3
40
30
20
10
0
-10
-20
-30
-40
var1
-40
-30
-20
-10
0
10
20
30
40
var2
-40
-30
-20
-10
0
10
20
30
40

- Preprocess the data and convert it to zeromean form
- `% Zero mean`
- `mu = mean(X);`
- `X_zeromean = X - mu;`
- Compute principal components of this data (apply Singular Value Decompositon)
- `% SVD`
- `Cov_mtx = (1./m) .* X_zeromean' * X_zeromean;`
- `[U, S, V] = svd(Cov_mtx); % Principal components`

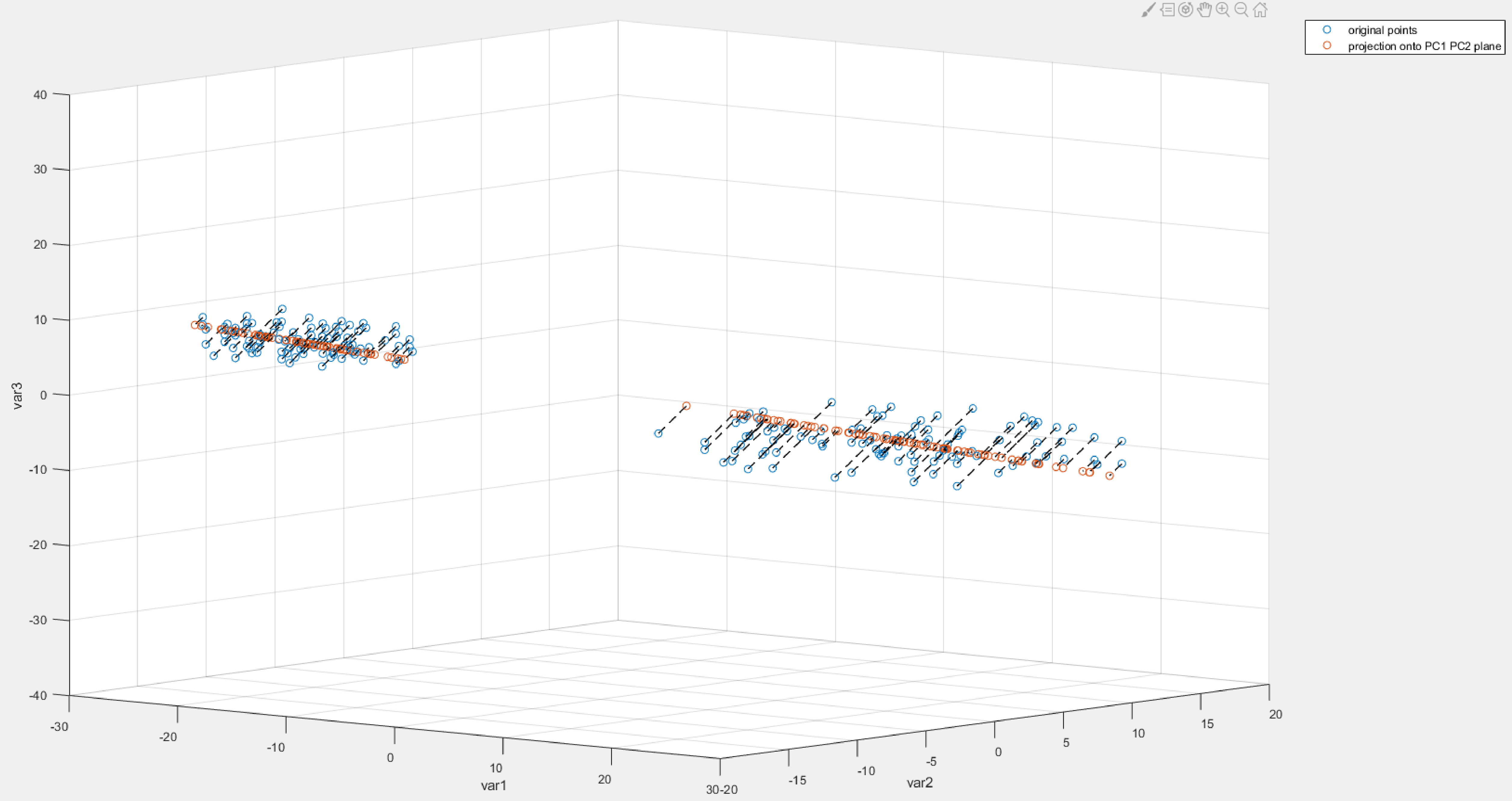
- In other words, find the directions along which we can capture maximum amount of variance.



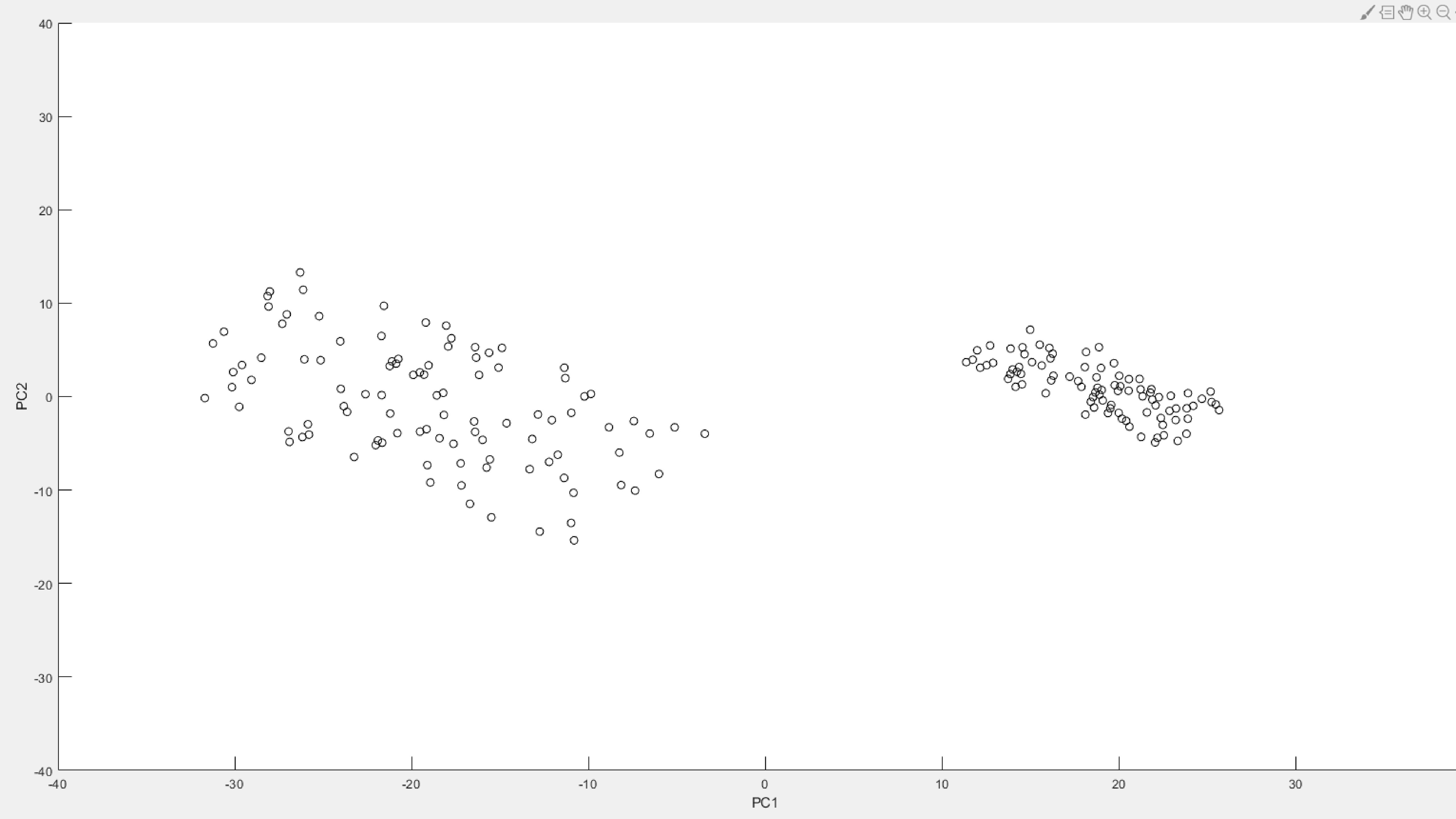
Data points
Principle Component 1(1/20)
Principle Component 2(1/4)
Principle Component 3(1/2)
var1
var2
var3
40
30
20
10
0
-10
-20
-30
-40

- Now we have the new basis for our new 2D system, PC1 and PC2.
- Project each data point onto this new plane spanned by PC1 and PC2.

```
% Project the data on PC1 PC2 plane
K = 2;
% get only first 2 principal components
U_reduced = U(:, 1:K);

% y is the results in the new 2-dimension space(PC1 PC2 plane)
y = U_reduced' * (X_zeromean)';
y = y';

% Recover the approximation data back with error. Which means we are
% visualizing data on the plane which spanned by principal components
U_reduced = U(:,1:K);
X_rec = y * U_reduced';
X_rec = X_rec + mu; % add the mean back
```


original points
projection onto PC1 PC2 plane
var1
var2
var3
40
30
20
10
0
-10
-20
-30
-40
-30
-20
-10
0
10
20
30
-20
-15
-10
-5
0
5
10
15
20

- Finally, we can plot the data reduced into PC1xPC2 plane.



PC2
PC1
40
30
20
10
0
-10
-20
-30
-40
-40
-30
-20
-10
0
10
20
30

- If we see the 3D plot with the principal components and this 2D reduced plot at the same time, we can notice the relation between them.

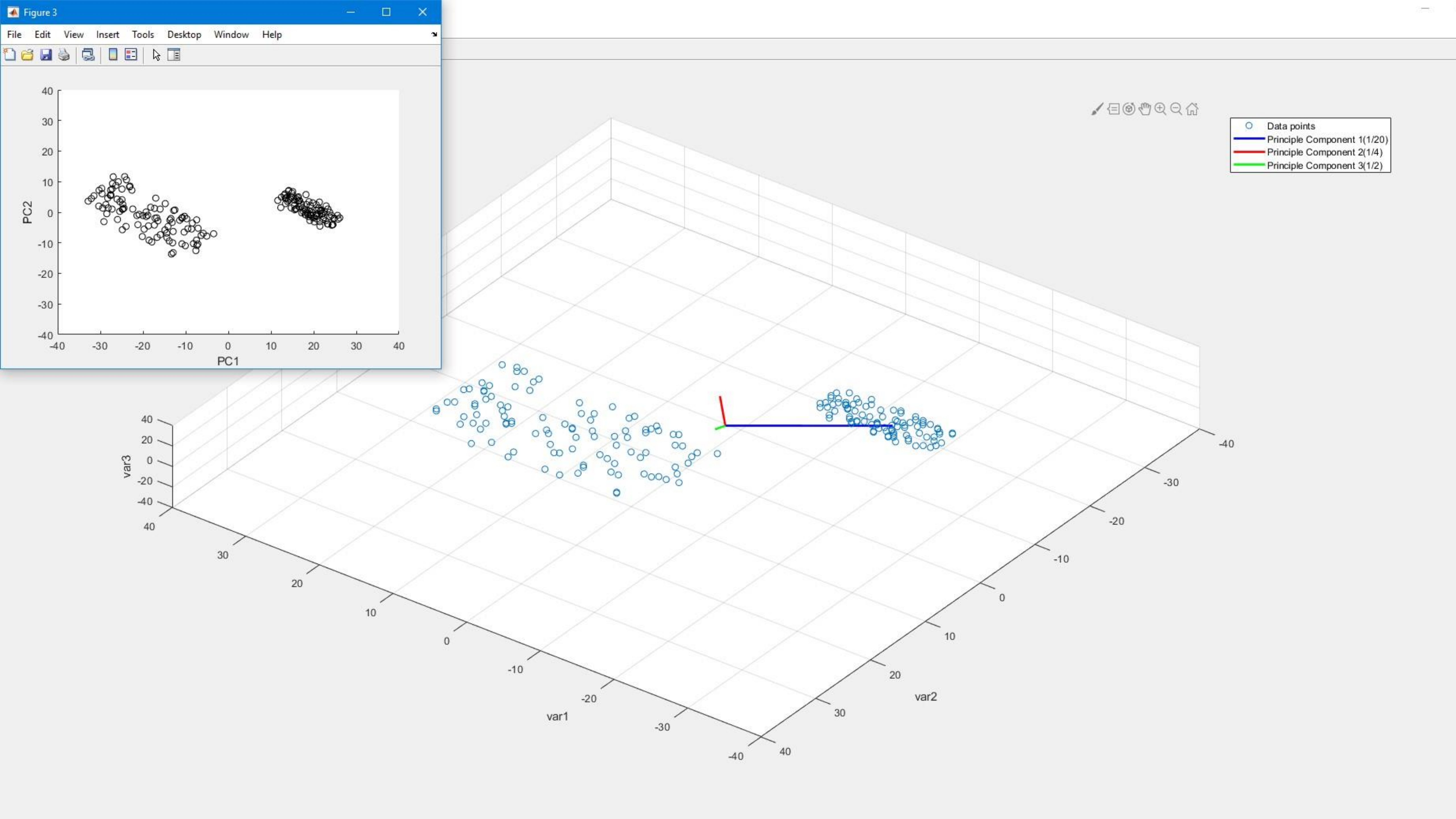
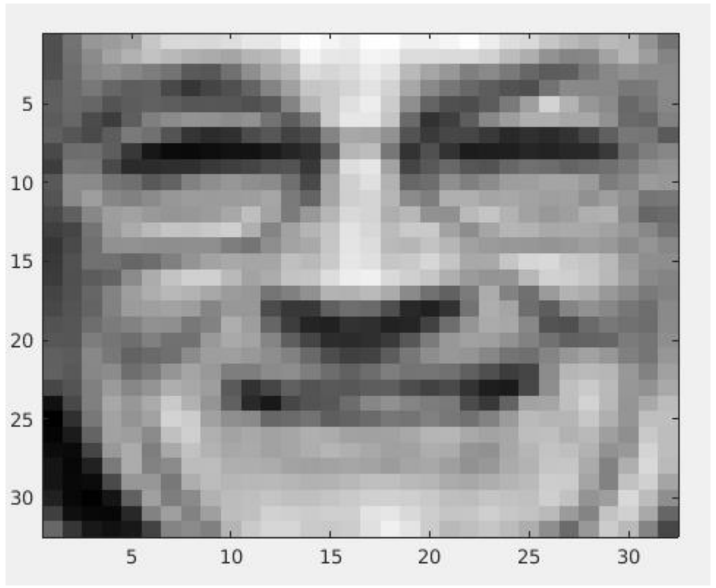


Figure 3
File Edit View Insert Tools Desktop Window Help
PC1
PC2
40
30
20
10
0
-10
-20
-30
-40
Data points
Principle Component 1(1/20)
Principle Component 2(1/4)
Principle Component 3(1/2)
var1
var2
var3

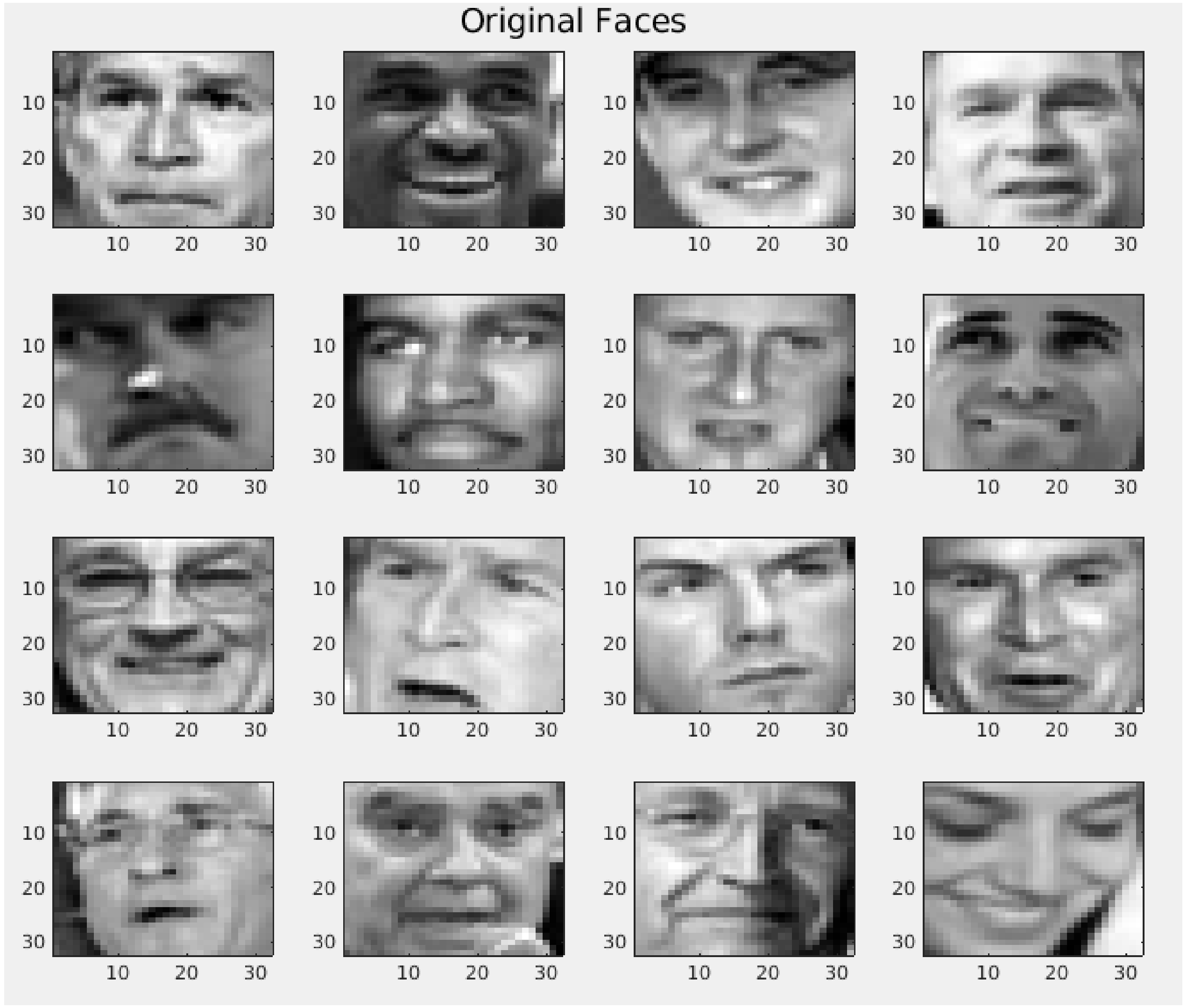
Practical Utilization of PCA

- Can we use PCA to reduces the number of features significantly?
- Can we use 10 times reduced dimensions?
- How much information we lose?

- To see the effect of PCA, we run a small demonstration.
- We collected 5000 grayscale face images from an existing dataset
- Each face picture is 32 x 32 which makes 1024 pixels for each image.
- 1024 pixels are the features, and we have total 5000 examples.
- Our data matrix is 5000 x 1024 where each image is in lexicographic ordering.



- Lets see 16 of them.

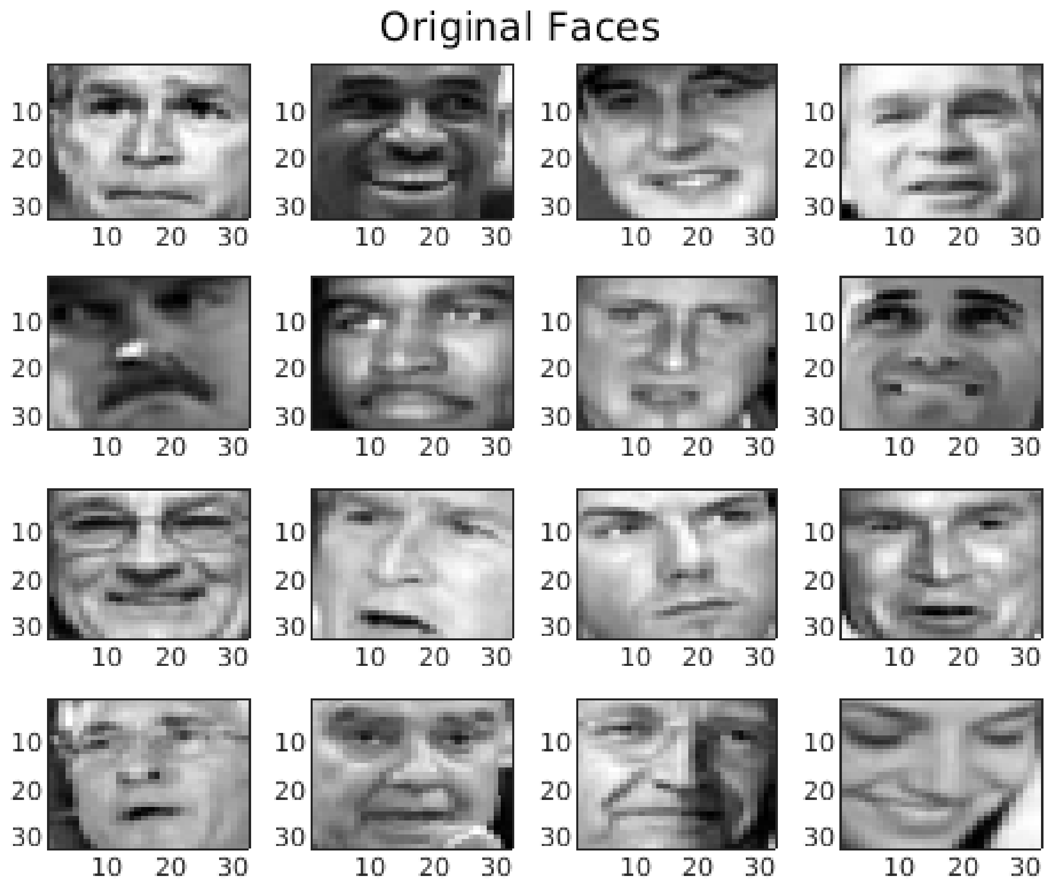


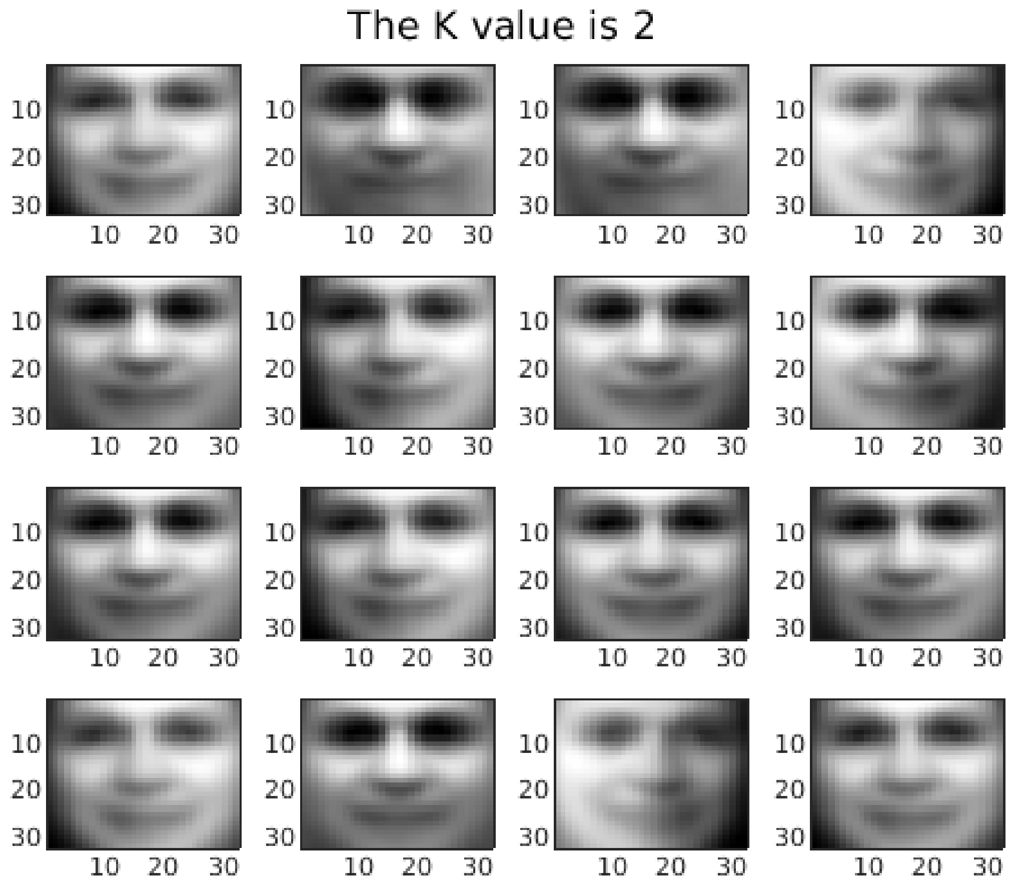
- In this practical application in addition to zero mean, we also normalize the variance of each pixel values. We obtain a X_norm which is zero mean and unit variance data set.

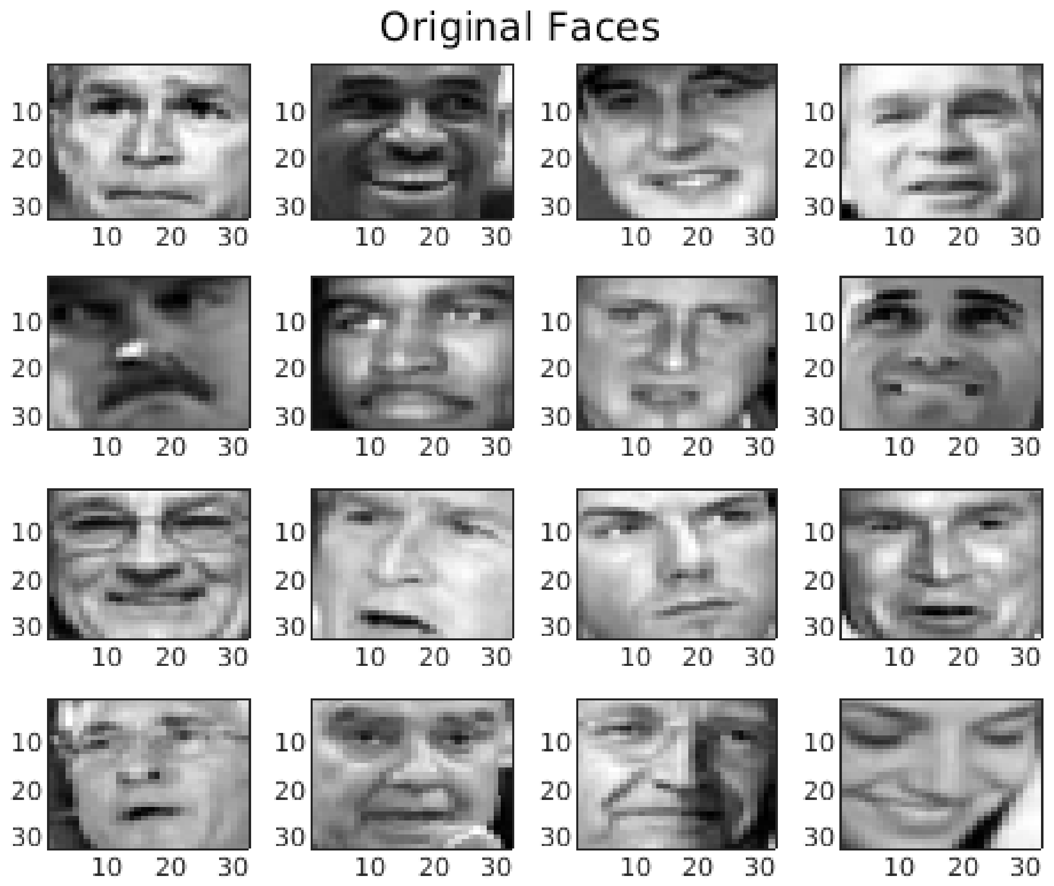
```
% How much variation each PC captures from the data.
var_perc = [];

[m, n] = size(X);  % m # of examples, n # of features (variables)

% Zero mean and normalization
mu = mean(X);
X_zeromean = X - mu;
sigma = std(X_zeromean);
X_norm = X_zeromean./sigma;

% SVD
Cov_mtx = (1/m).*X_norm'*X_norm;
[U,S,V] = svd(Cov_mtx); % Principal components
```

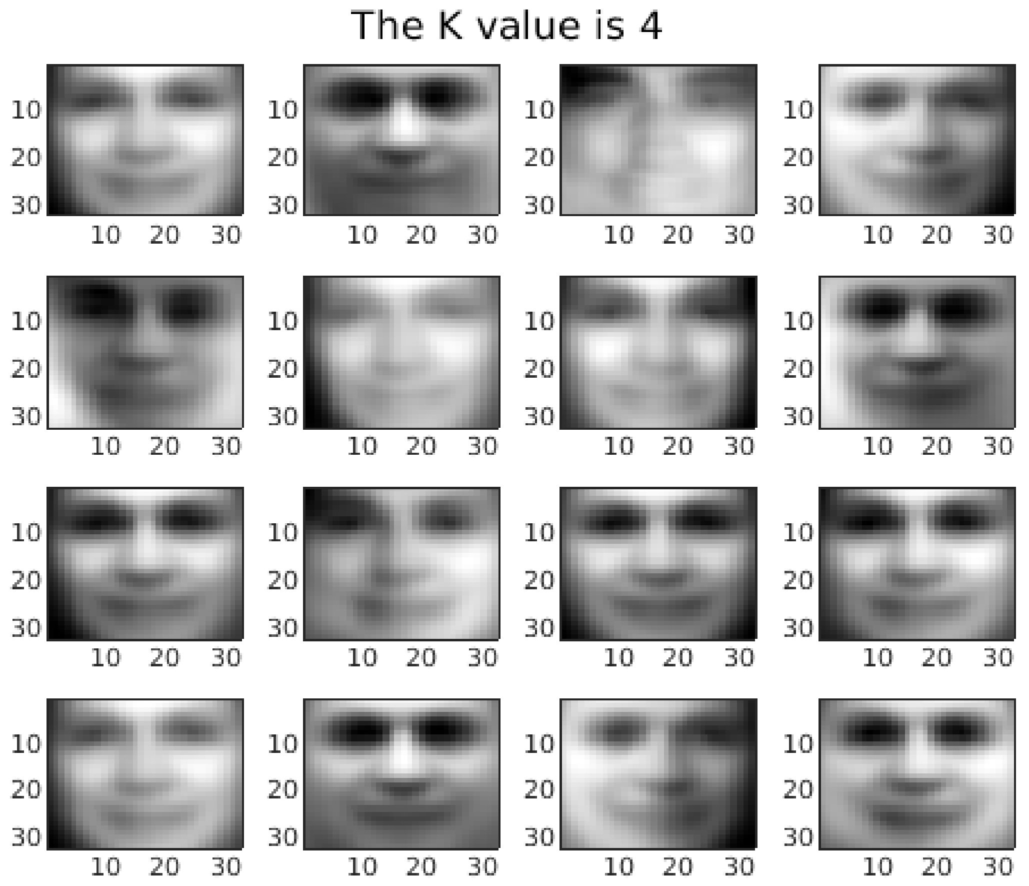
- After finding the principal components the steps are identical with the first and second examples

```
K_set = [512 256 128 64 32 16 8 4 2 1];

for k = 1:length(K_set)

    % Project data
    K = round(1024/K_set(k));
    U_reduced = U(:,1:K);
    Z = X_norm * U_reduced;

    % Recover the approximation data back with error. Which means we are
    % visualizing data on the line which spanned by principal components
    U_reduced = U(:,1:K);
    X_rec = Z*U_reduced';

    % Revert the preprocessing step
    X_rec = X_rec.*sigma;
    X_rec = X_rec + mu;

end
```

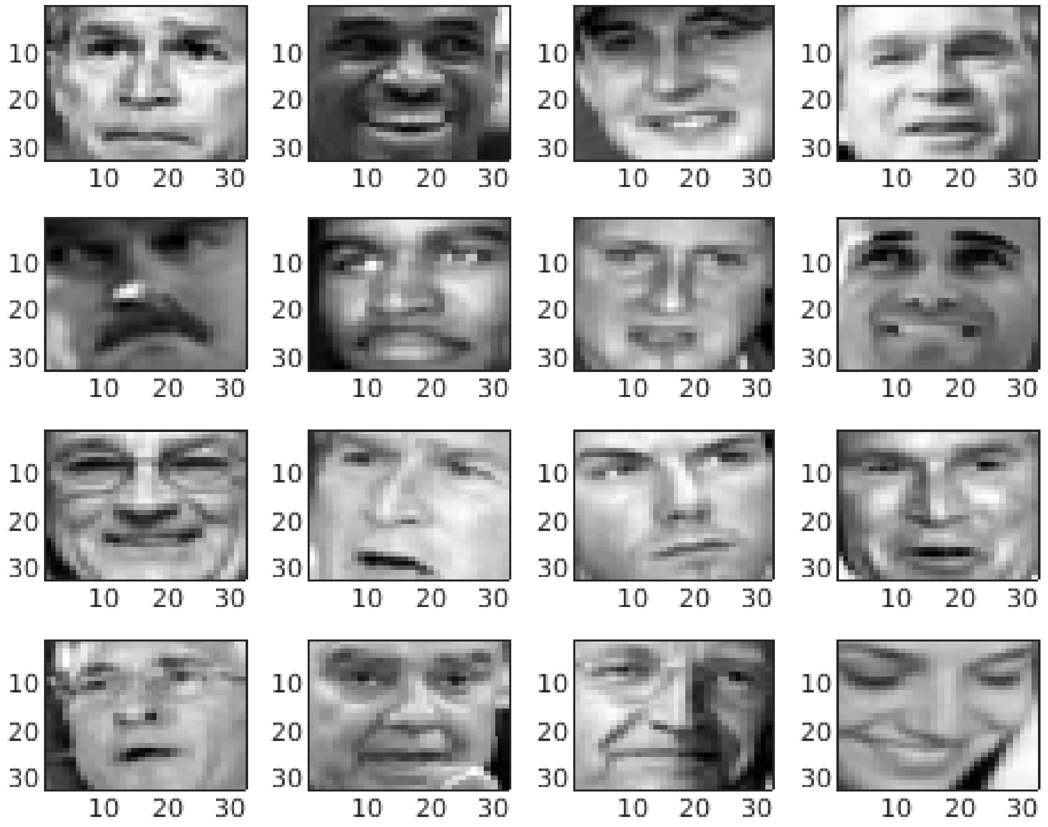

Original Faces
10
20
30


The K value is 2
10
20
30

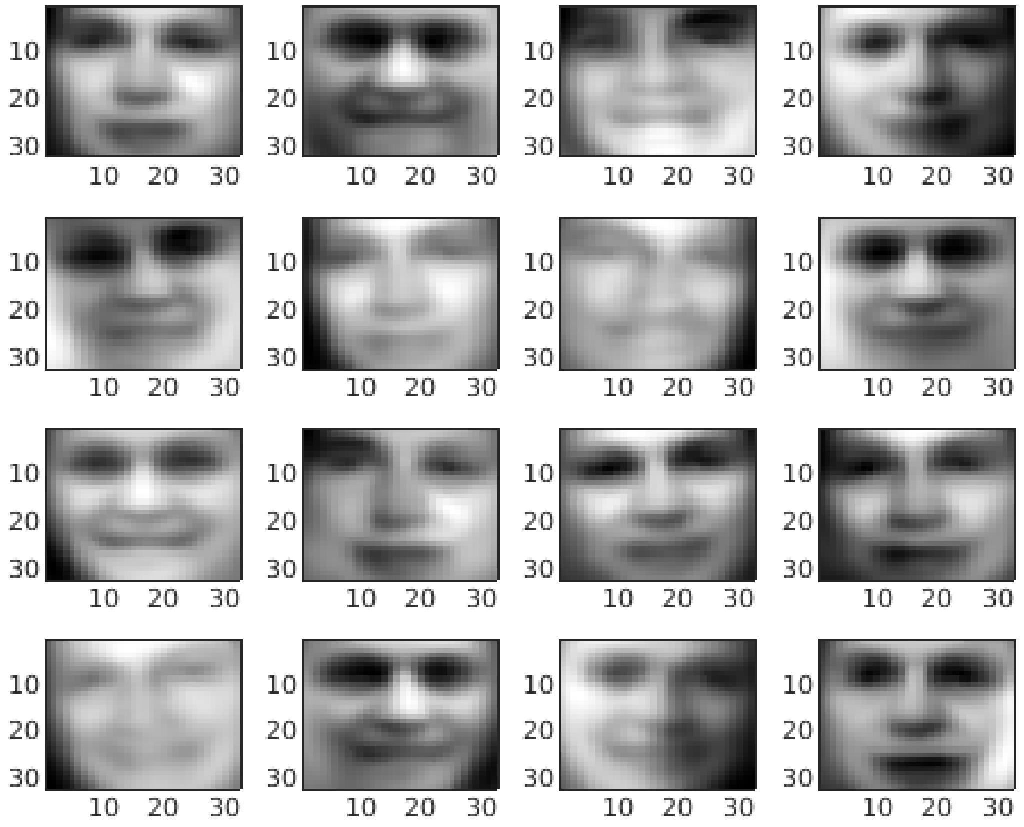

Original Faces


The K value is 4

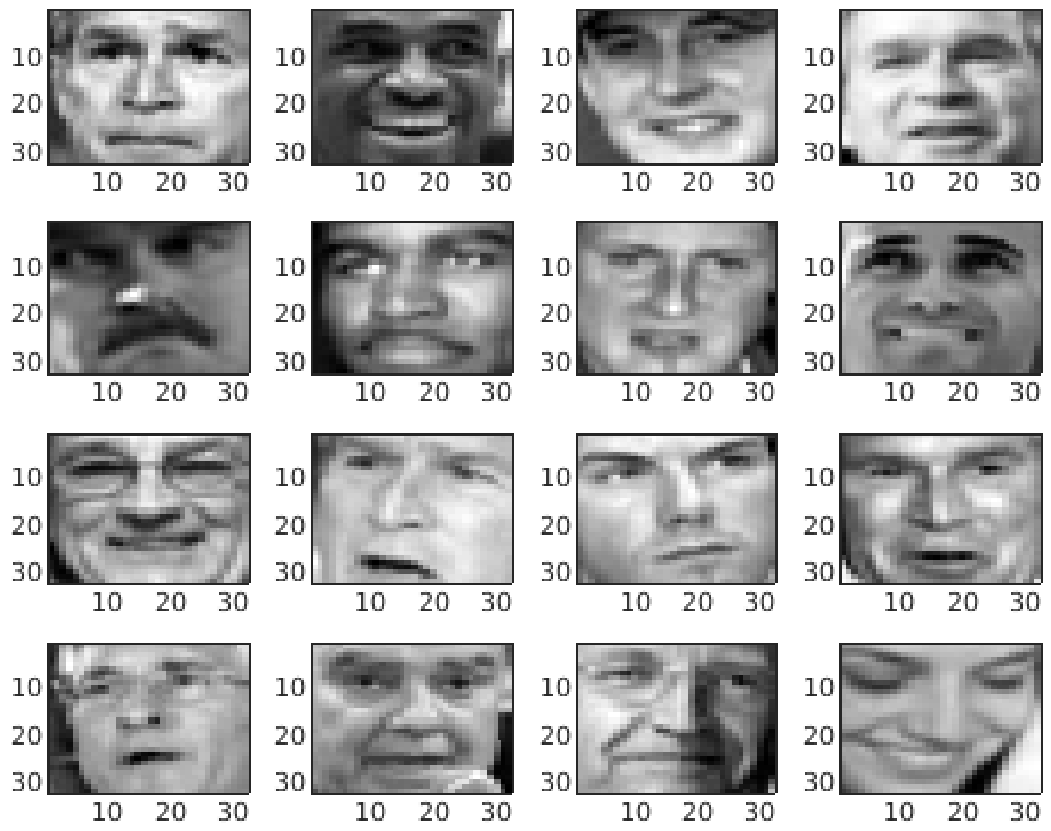
Original Faces



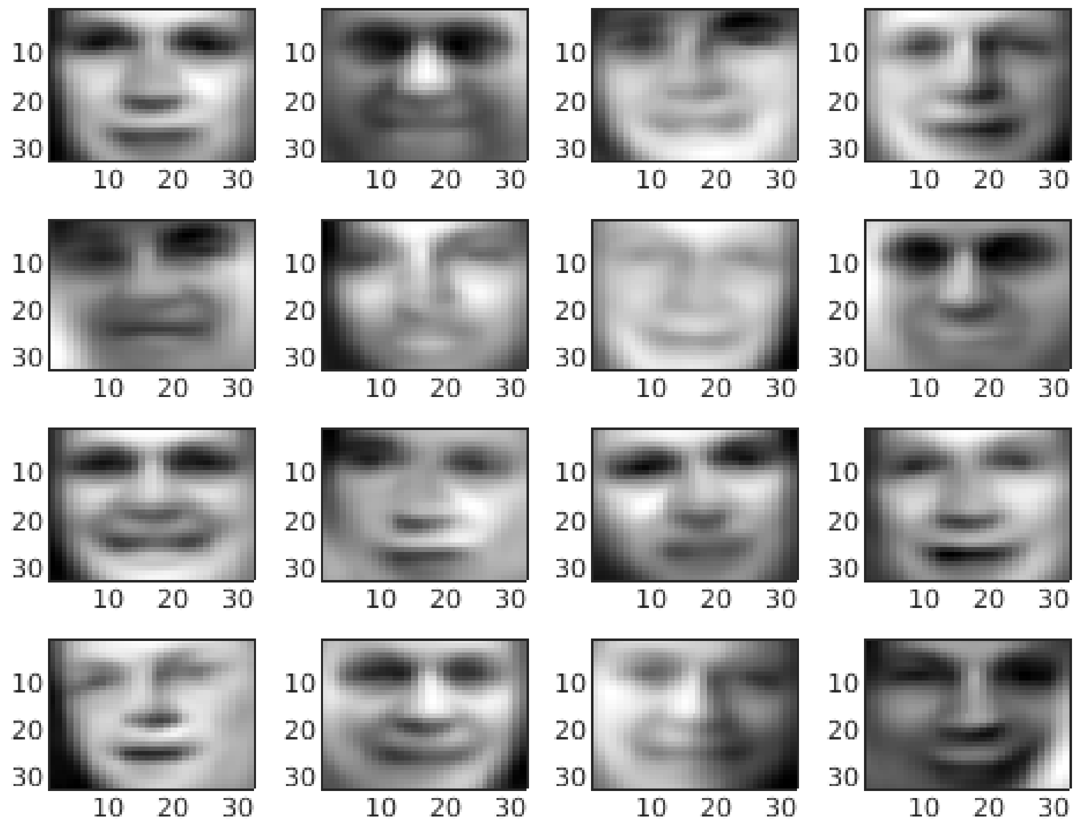
The K value is 8

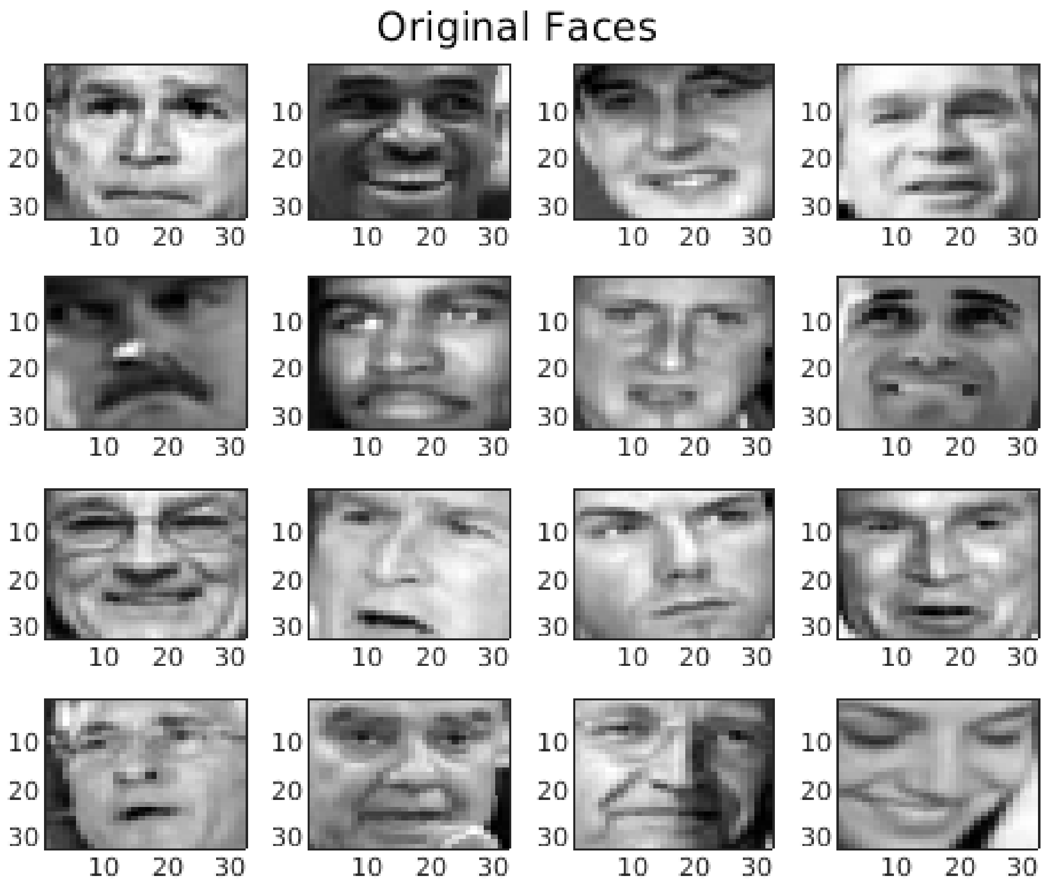


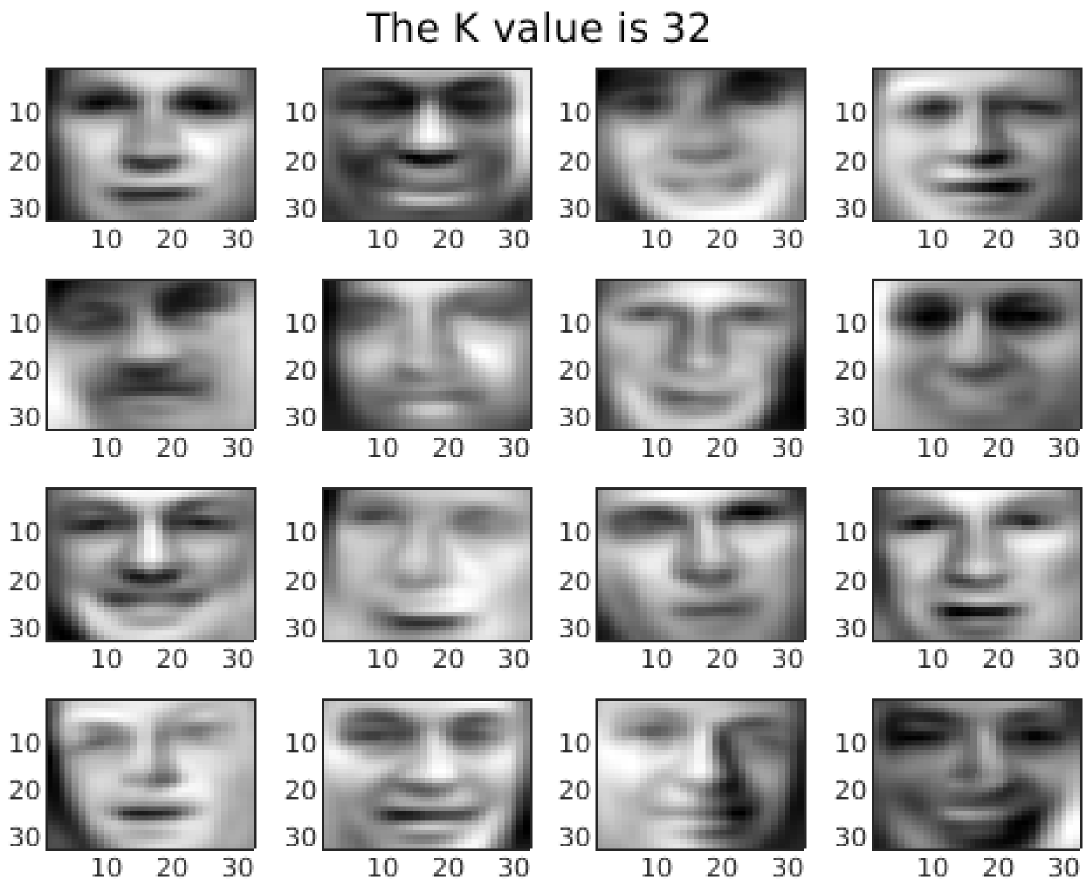
Original Faces

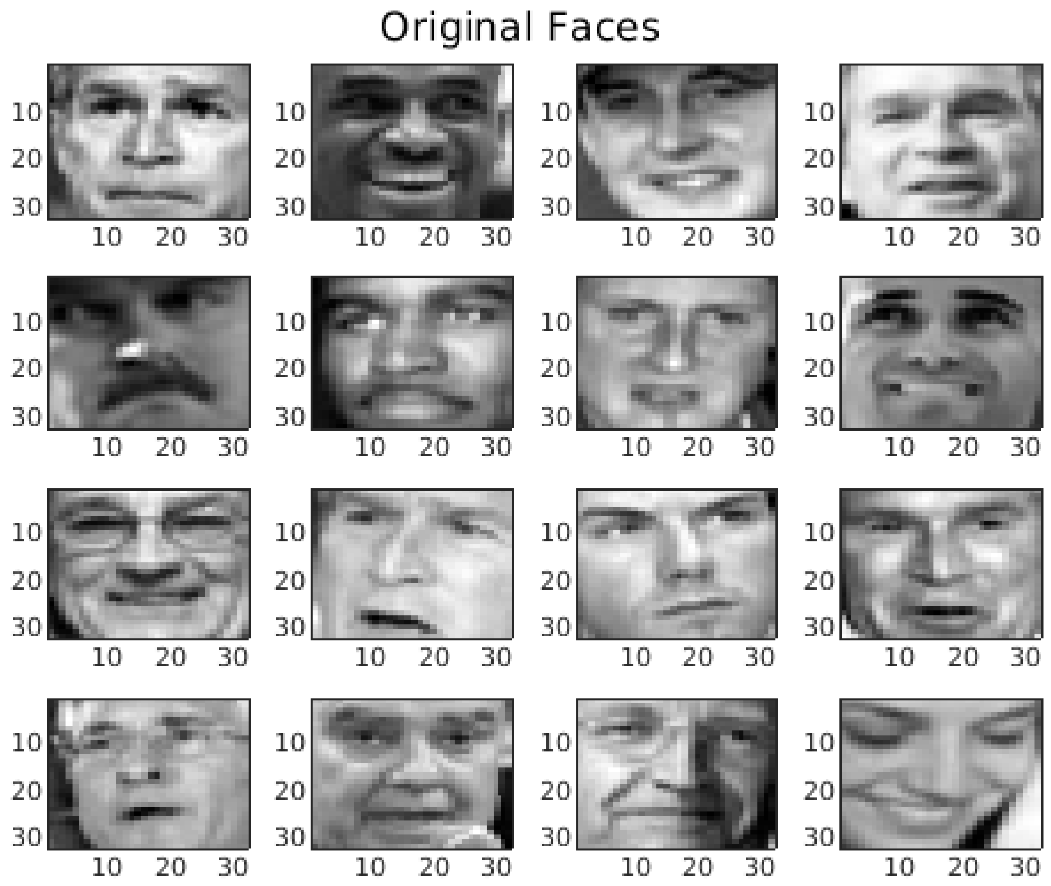


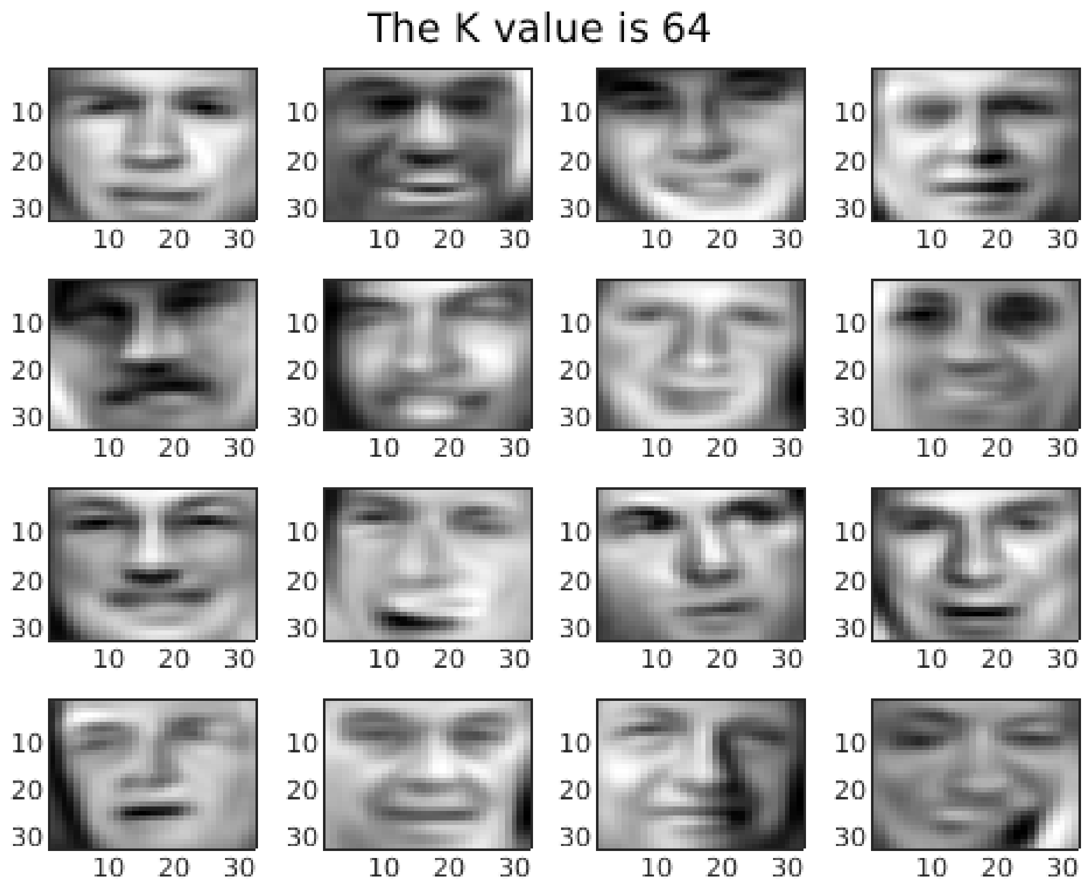
The K value is 16

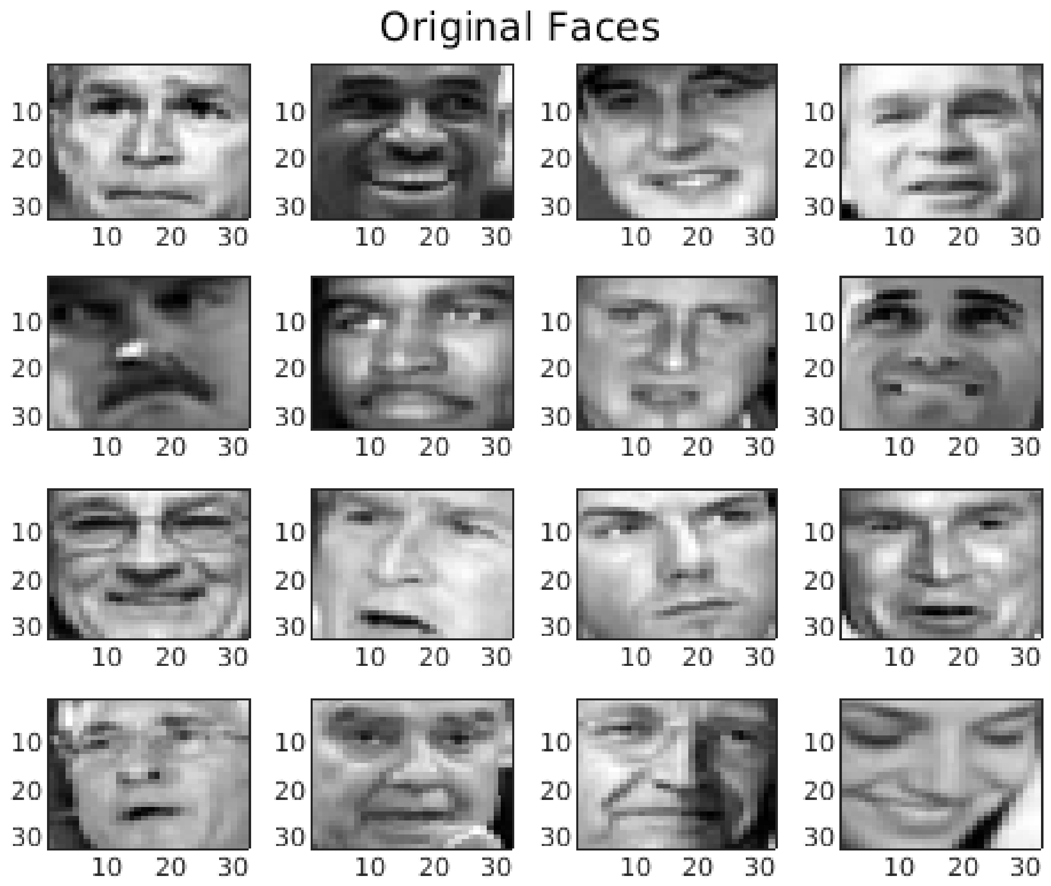


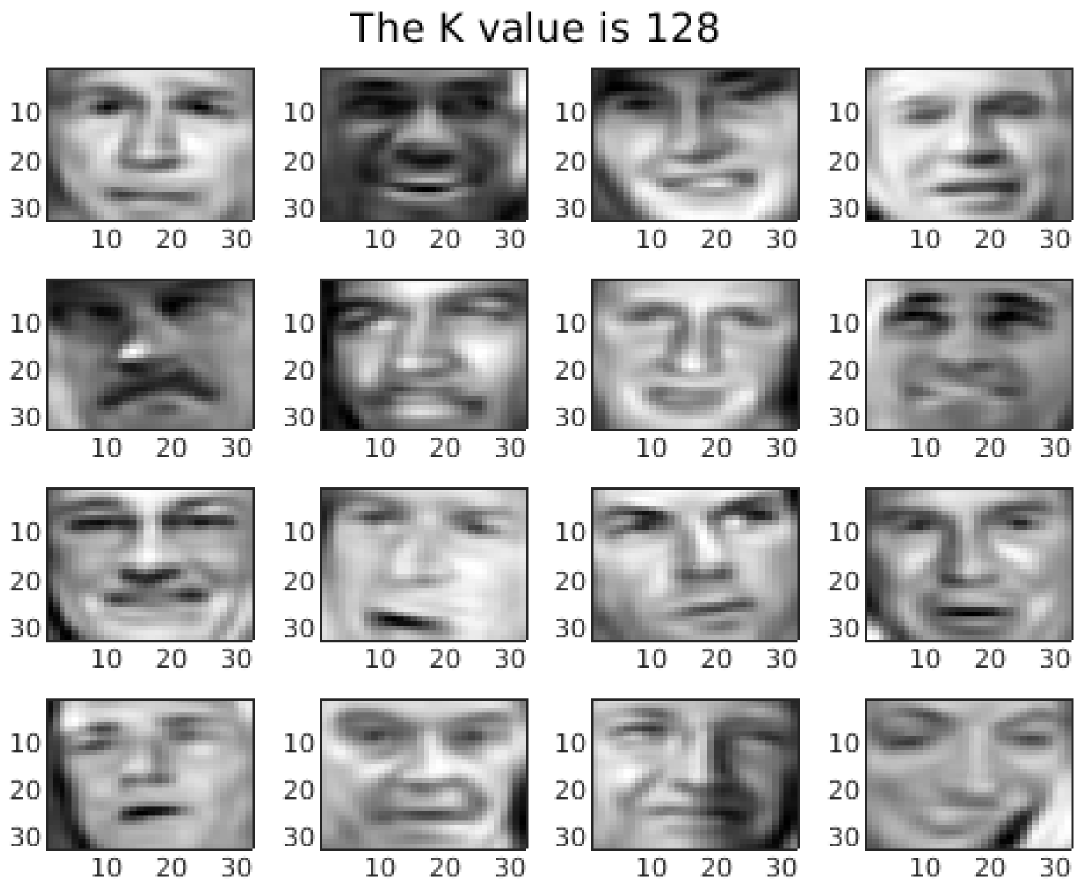

Original Faces
10
20
30

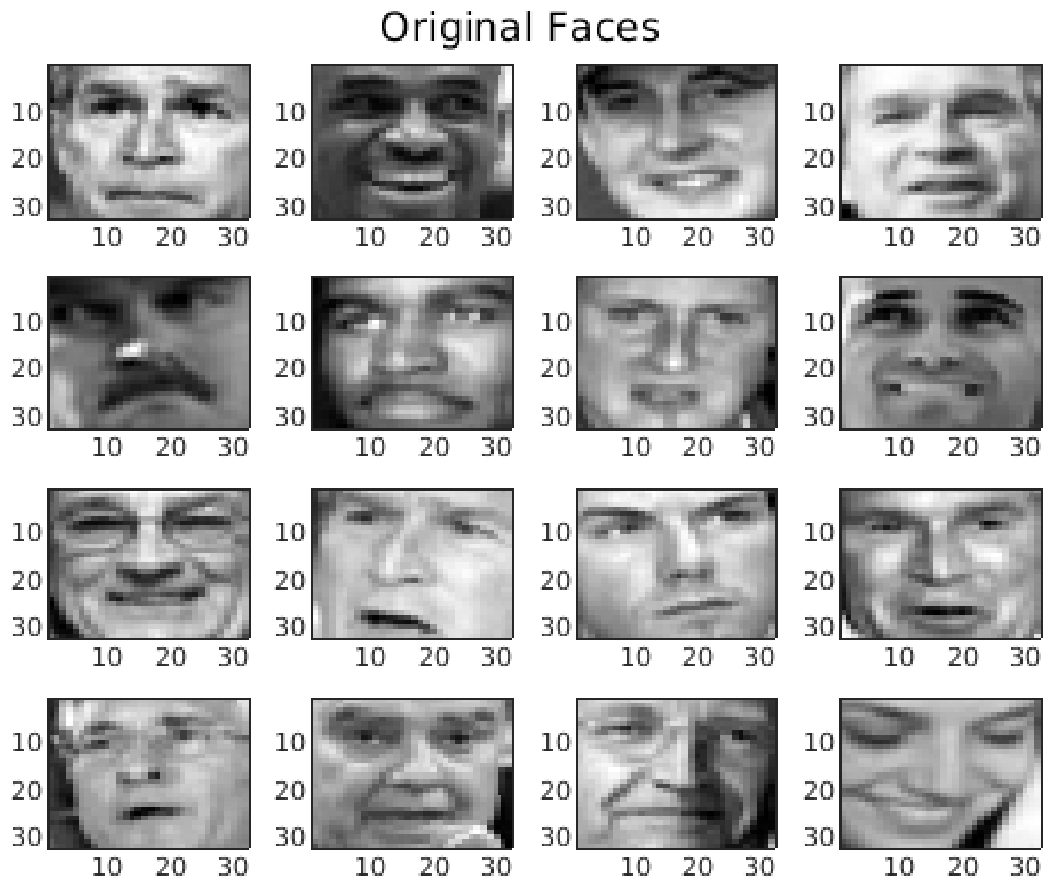

The K value is 32
10
20
30

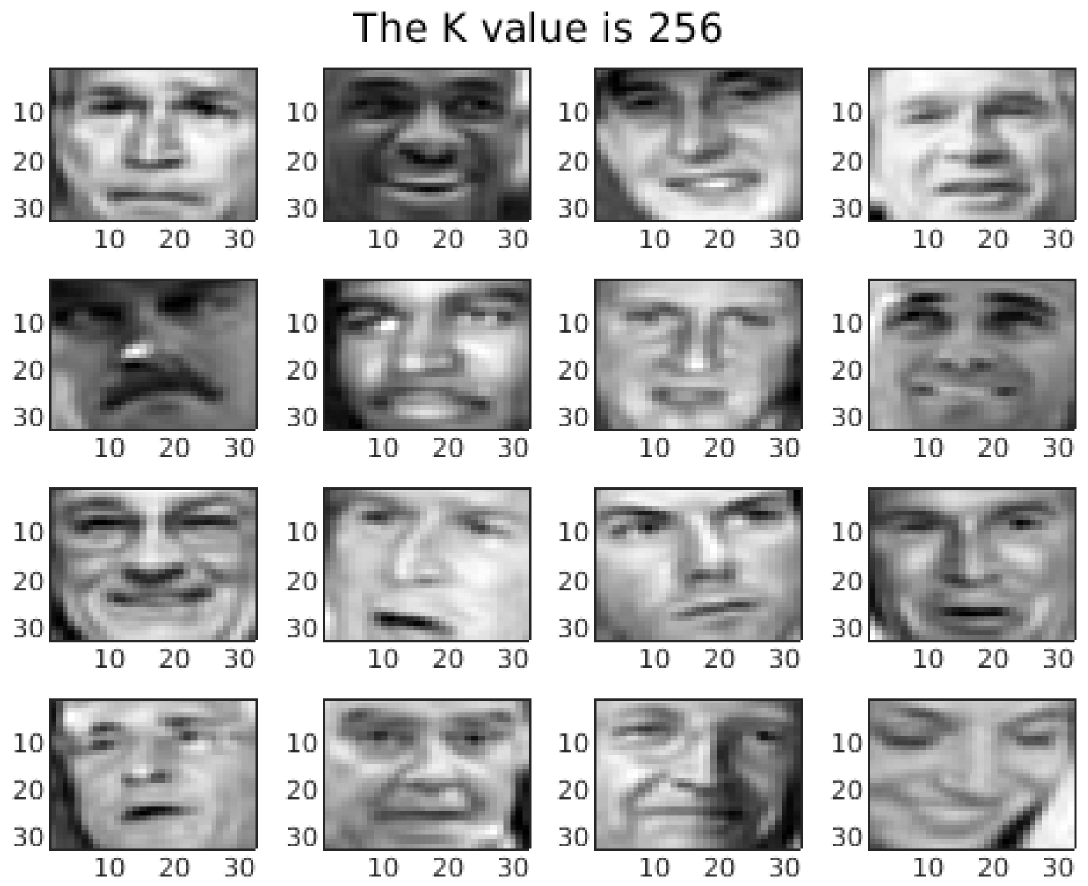

Original Faces
10
20
30

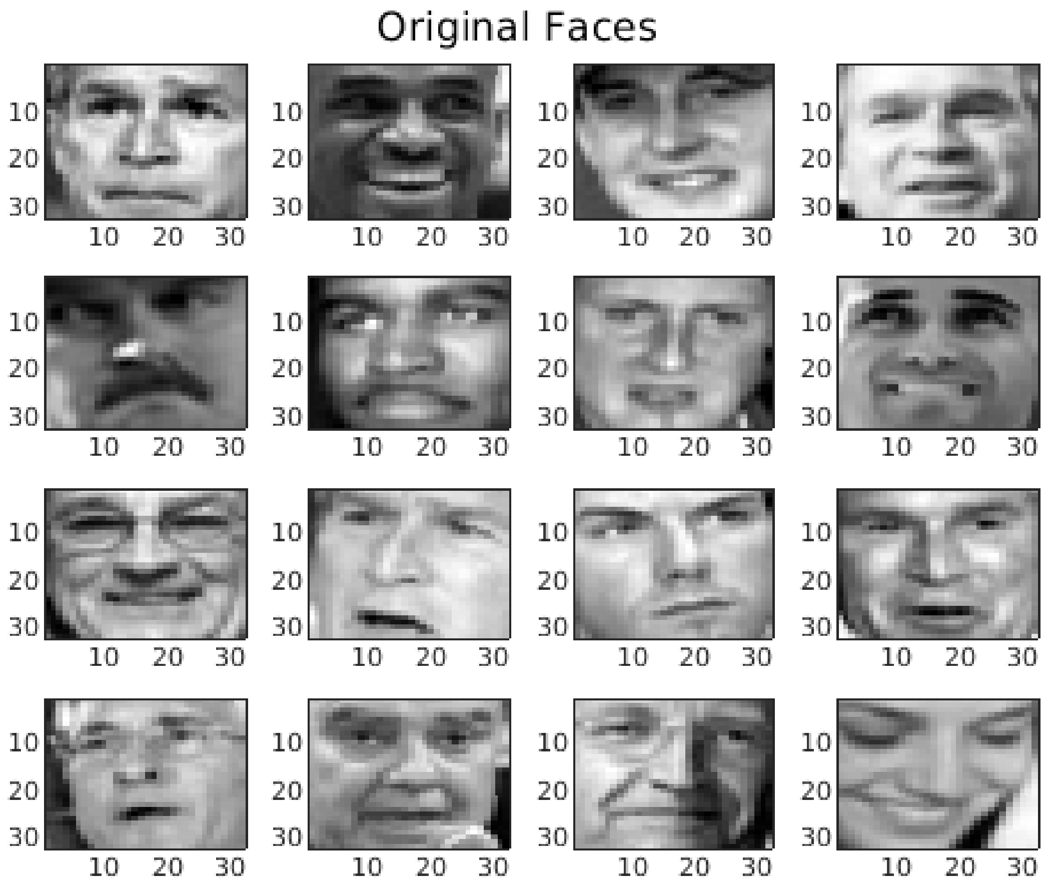

The K value is 64
10
20
30

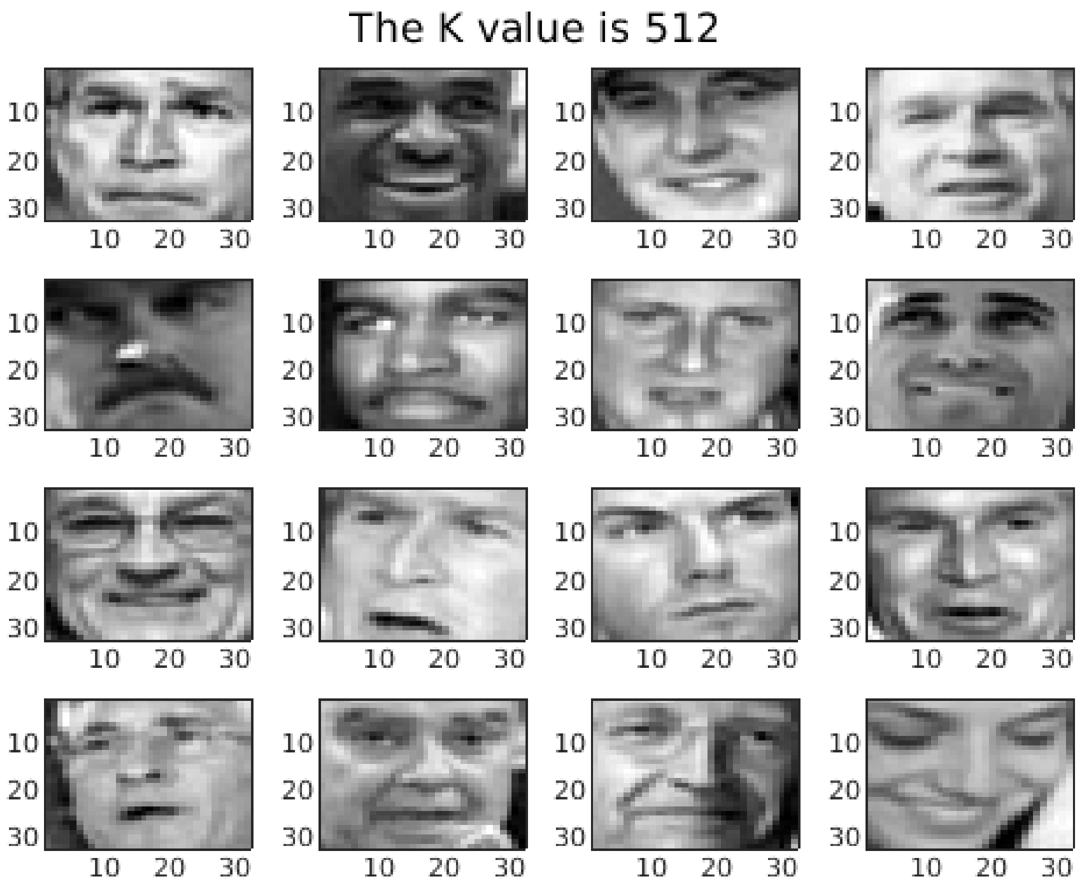

Original Faces


The K value is 128

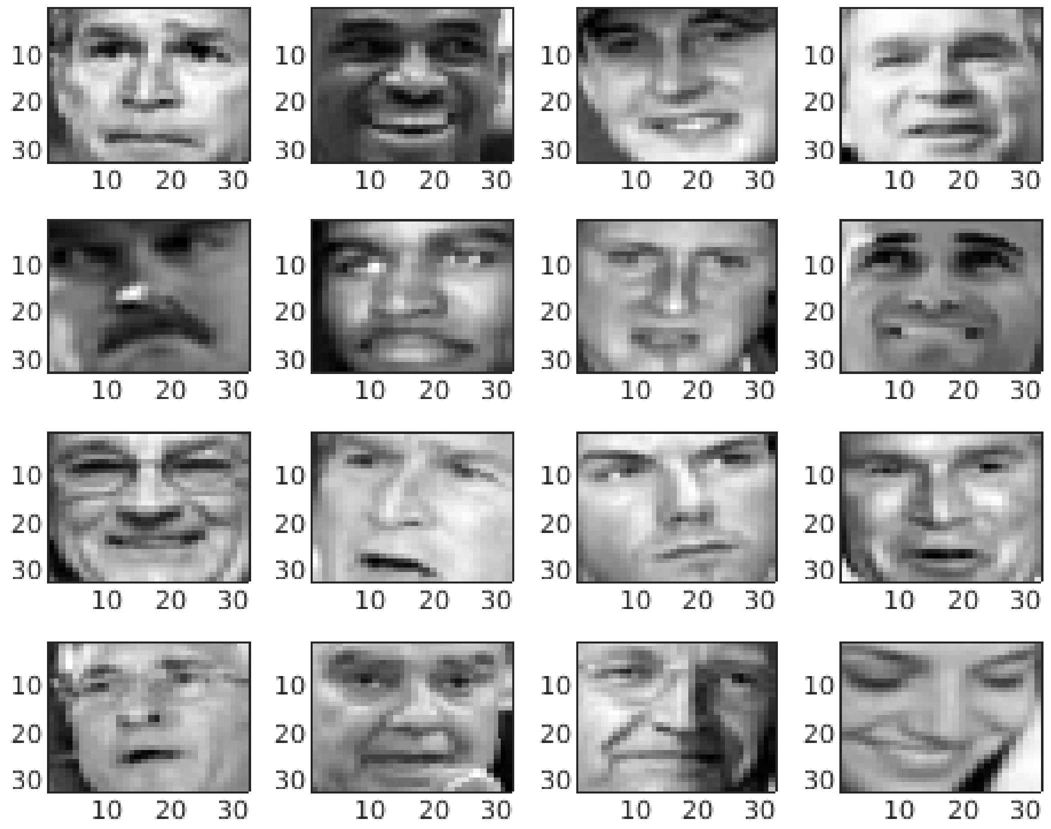

Original Faces
10
20
30


The K value is 256
10
20
30

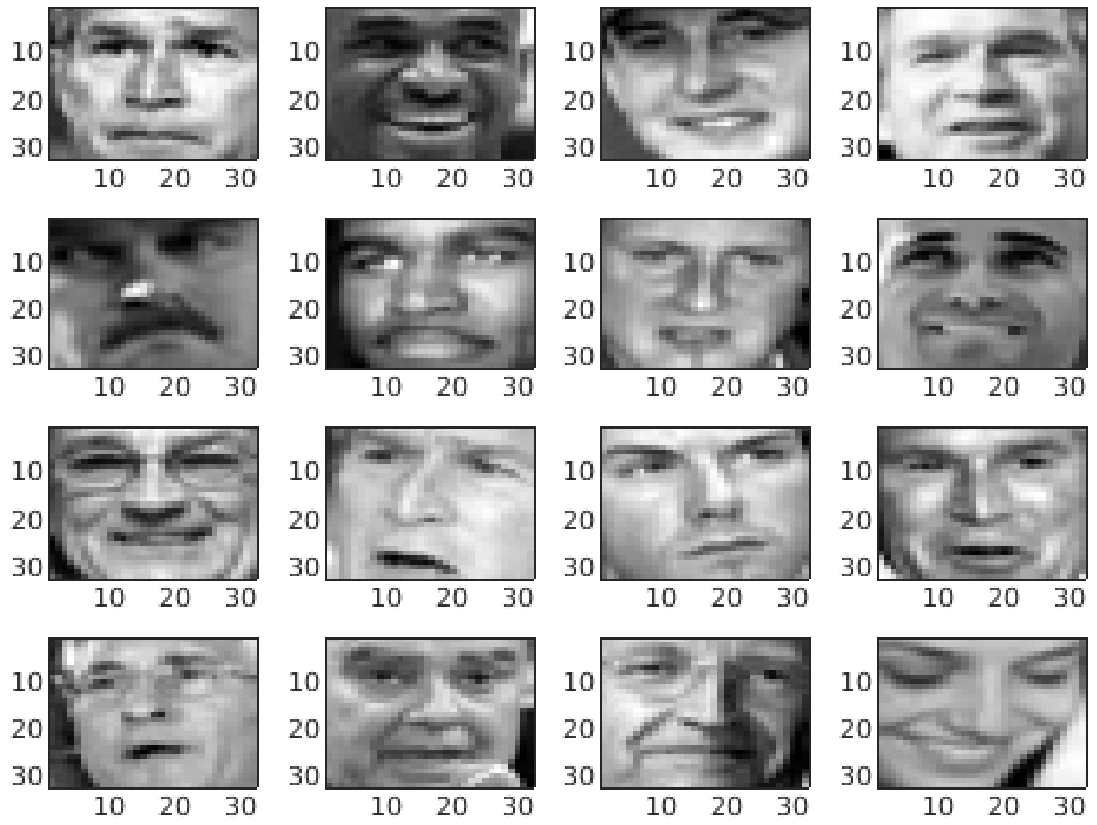

Original Faces


The K value is 512

Original Faces



The K value is 1024

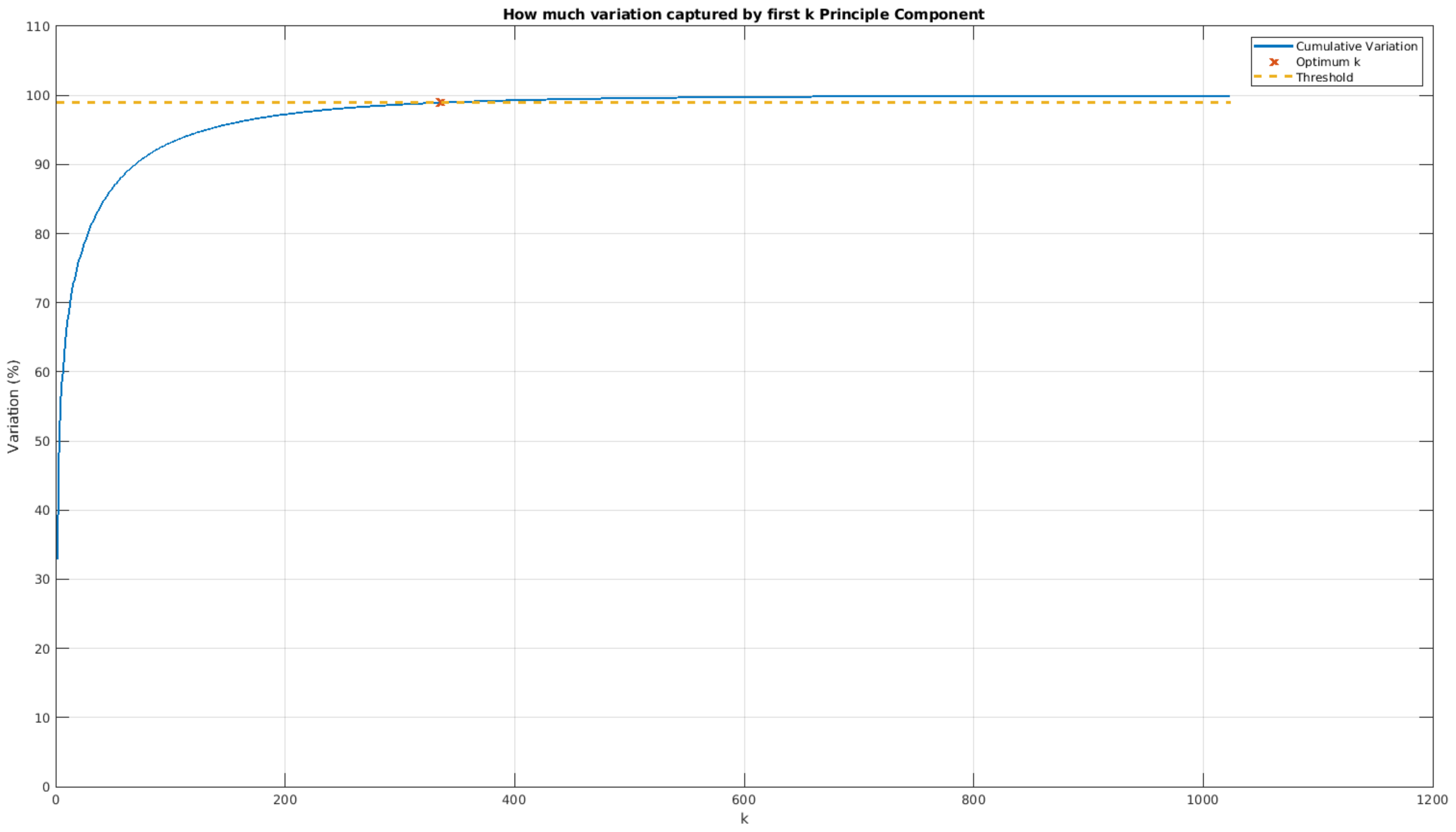


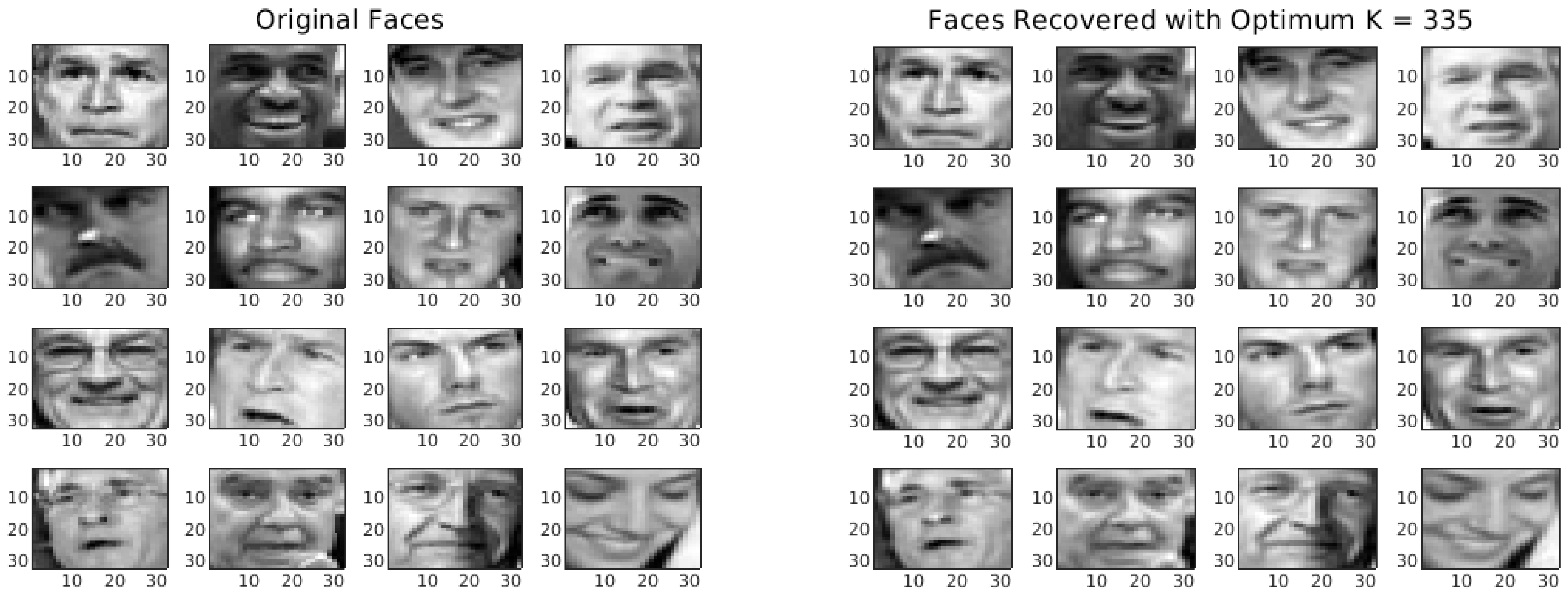
Is there a better way to find the optimum K value?

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)} - x_{rec}^{(i)}\right\|^2}{\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}\right\|} \leq 0.01$$

$$\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \geq 0.99$$

```
total_var = trace(S);

opt_k = 1;
thresh_vec = [];
THRESHOLD = 99;
for j = 1:n

    thresh = 100*trace(S(1:j, 1:j))./total_var; % in percentage
    thresh_vec(end+1) = thresh;

    if thresh > THRESHOLD
        continue;
    else
        opt_k = opt_k +1;
    end

end
```


How much variation captured by first k Principle Component
Cumulative Variation
Optimum k
Threshold
Variation (%)
k
0
10
20
30
40
50
60
70
80
90
100
110
0
200
400
600
800
1000
1200


Original Faces
Faces Recovered with Optimum K = 335
10
20
30