Project Report

GuiCheng Wu
Xinyi Xie

March 12, 2016

1 Standard Principle Component Analysis

PCA assumes the basis matrix P is an orthonormal matrix and the directions with the largest variances are the most 'important', in other words, the most principal. Because our goal is to reduce the data set into a three dimensional space, all we need is to find out the largest three variance as our projection vectors.
Here is our goal for PCA. Let dataset $X = \{x_1, ..., x_n\}$ be a d by n matrix, where d is the number of measurement types (dimensions) and n is the number of data trials, and x_i is a d by 1 vector. The goal is to project to M-dimensional subspace. The step is as follows:
1. Ensure every dimension of X has zero mean.

$$X = X - \text{mean}(X) \tag{1}$$

2. Compute the covariance matrix, we denoted it as Σ.

$$\Sigma = \frac{1}{N-1} X * X^T \tag{2}$$

3. Find the eigenvectors and eigenvalues of the covariance matrix and sort the eigenvalues to find out the largest three eigenvalues.

$$[V, D] = \text{eig}(\Sigma) \tag{3}$$

where D is a diagonal matrix of eigenvalues and V is the corresponding eigenvectors. The corresponding eigenvectors are our projection vectors, denoted by $\{v_k\}$. In our case, we only require 3 eigenvectors which have the largest eigenvalues. Then we get our projected data as follow.

$$\text{result} = v_k^T * X \tag{4}$$

Also we use a SVD version of PCA. Here is the subroutine:
The first thing is the same as above, to ensure every feature of X has zero mean.

$$X = X - mean(X) \tag{5}$$

Then construct the matrix

$$Y = X'/\sqrt{n-1} \tag{6}$$

And all the other things leave to SVD,

$$[u, s, v] = svd(Y) \tag{7}$$

where s is a diagonal matrix with nonnegative diagonal elements in decreasing order. And then calculates the variance and project the original data,

$$v = s. * s, \text{result} = v_k' * X \tag{8}$$

1.1 PCA Code

The PCA codes are as follows.

```
function [projectedData, projectionVectors]
                                = PCA(data, dimensionSize)
[M,N] = size(data);
mn = mean(data,2);

% subtract off the mean for each dimension
data = double(data) - repmat(mn,1,N);

% calculate the covariance matrix
covariance = 1 / (N-1) * (data * data');

% find the eigenvectors and eigenvalues
[eigVectors, eigValues] = eig(covariance);

eigValues = diag(eigValues);
[junk, rIndices] = sort(eigValues, 'descend');
eigValues = eigValues(rIndices);
eigValues = eigValues(1:dimensionSize);
eigVectors = eigVectors(:,rIndices);
projectionVectors = eigVectors(:, 1:dimensionSize);
projectedData = projectionVectors' * data;
```

1.2 Valid the correctness of PCA

In order to valid the correctness of our PCA function, we use the classic test case "swiss roll'. The input data to the PCA is 2000 points which make a three-dimensional s-shaped manifold. Then we use our PCA to project these data points into a two-dimension space. As the figure 1 shows, the PCA succeeds project the three-dimensional swiss roll into two dimensions. This proves the correctness of our PCA function.

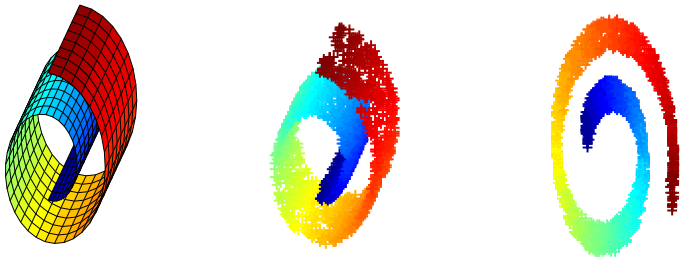


Figure 1: PCA swiss roll test case

1.3 Projection vectors and the scatter matrix

Our dataset is all the 624 images provided in the assignment 3. The three PCA projection vectors of standard PCA are as Figure 2 shown.

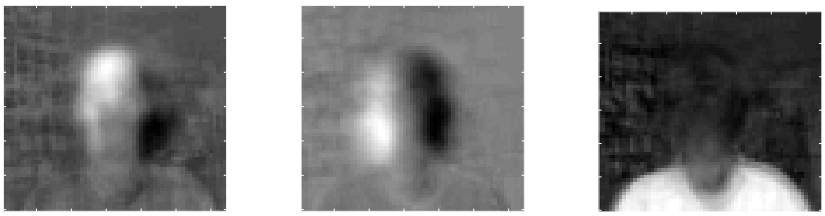


Figure 2: three projection vectors of PCA

The overall scatter matrix is as Figure 3 shown.

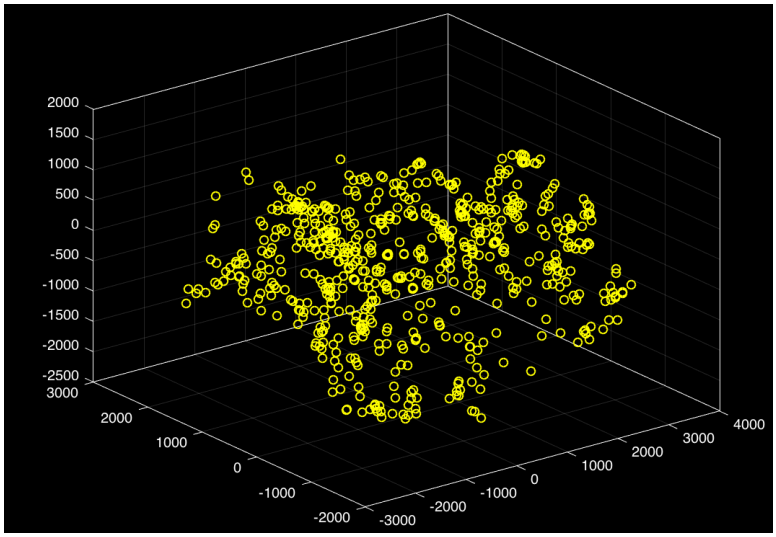
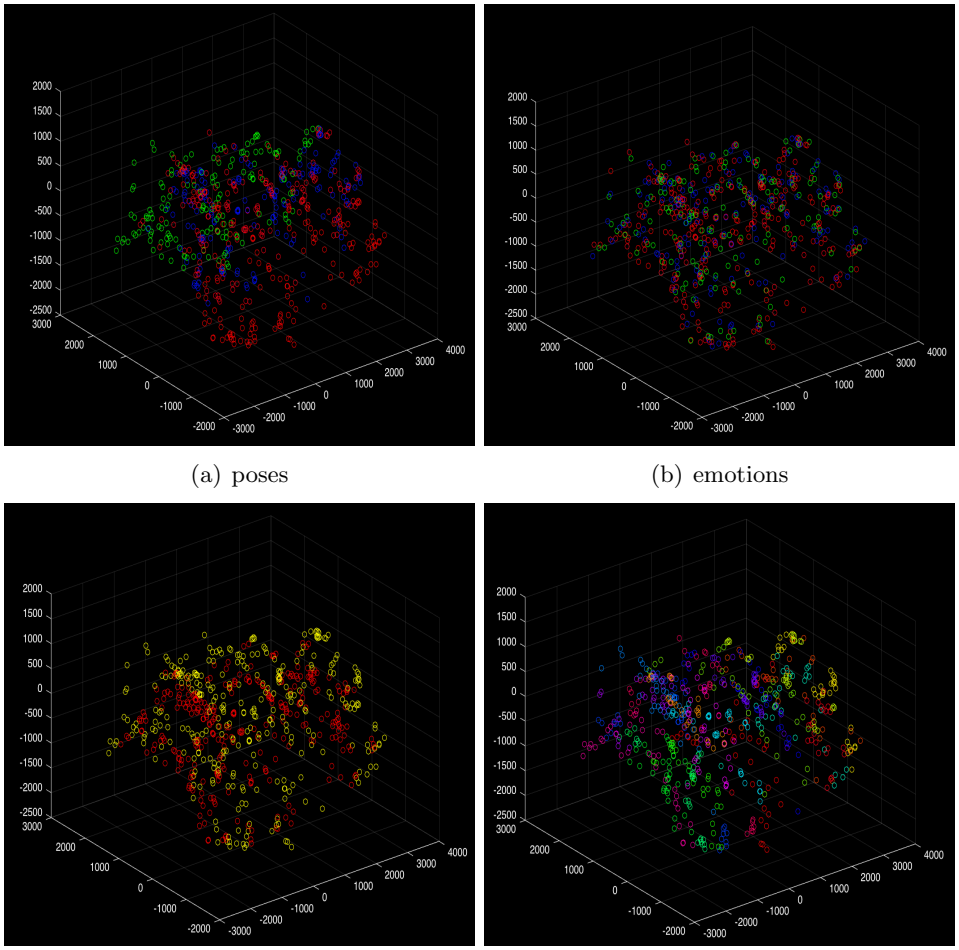


Figure 3: PCA scatter matrix of all 624 images

We also colors the PCA scatter matrix based on 4 poses, 4 emotions, sunglasses or not, and 20 names. They are shown in Figure 4.



(a) poses (b) emotions

(c) glasses or not (d) names

Figure 4: scatter matrix based on poses, emotions, glasses or not and names

In our experiment, we successfully reduce the data set from 3840-dimension to 3-dimension. The projection vectors and scatter matrix are both shown above. Generally, the projection data are widely spreading out, however, due to the significant change of dimension from 3840 to 3, the projection data lose many features comparing with the original images. Therefore, in the experiment we cannot cluster the projection data according the poses, emotions, glasses and names.

2 LLE: locally linear embedding

The LLE algorithm is based on k-nearest neighbors of each data point. We expect each data point and its neighbors to lie on a locally linear patch of the manifold. There are three steps in the LLE algorithm. First, we find k-nearest neighbors. Second, we compute the optimal reconstruction weights for each point by its neighbors. Third, we perform the embedding by perserving the reconstruction weights of any point in the low dimensional space.

There are three steps in our algorithm:
1. Compute the neighbors of each data points;
2. Compute the weight of W_{ij} that best reconstruction each data point using its neighbors;
3. Compute the vectors which best reconstructed by weight W_{ij}.

For our data set, we need to maintain the distance in the manifold space. Thus we record the k nearest neighbor to maintain the structure. Then we need to find the optimal reconstruction for each point by its neighbors, which is our step two. Given the reconstruction weights for each point, an embedding into d dimensional space, in our case $d = 3$, will be performed in step 3.

2.1 LLE Code

The LLE codes are as follows.

```
function [projectedData, eigenVectors] = LLE(data, k, d, solver)
[D, N] = size(data);

%Compute the euclidian distances
columnDistanceSum = double(sum(data.^2, 1));
distance = repmat(columnDistanceSum,N,1) +
 repmat(columnDistanceSum',1,N)-2*double(data')*double(data);

[sortedDistance, index] = sort(distance);
%find k neighbors
neighbors = index(2:(1+k), :);

%define sparse matrix
AW = spalloc(N, N, k*N);
for i =1:N
```

```
    z = double(data(:,neighbors(:,i))-repmat(data(:, i),1,k));
    C = z'*z;
    C = C + eye(k, k)*tol*trace(C);
    w = C\ones(k, 1);
    AW(i, neighbors(:, i)) = w/sum(w);
end
%compute cost matrix M
WI = spdiags(ones(N, 1), 0, N, N) - AW;
M = (WI)'*WI;
options.disp = 0;
options.isreal = 1;
options.issym = 1;
options.v0 = ones(size(M, 1), 1);
% select eignesolver
switch solver
    case 'eig'
      [eigenVectors,eigenValues]=eigs(M, d+1, 0, options);
    case 'svd'
      [eigenVectors,S,V] = svds(M, d+1, 0, options);
end

eigenVectors = eigenVectors(:, 1:d)';
projectedData = eigenVectors* sqrt(N);
eigenVectors = double((eigenVectors * double(data'))');
```

2.2 Valid the correctness of LLE

Again we use swiss roll as test case to valid our LLE algorithm. The figure 5 shows our LLE algorithm should be correct, because it succeeds in reducing dimensions.

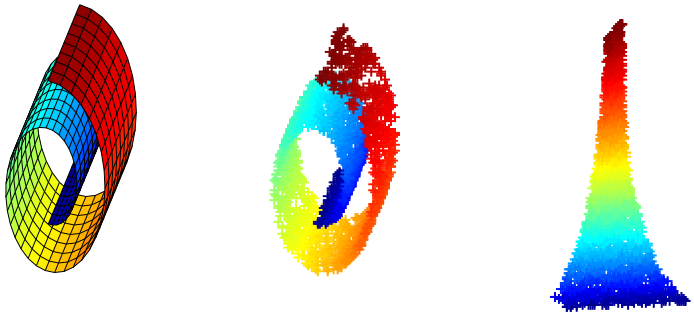


Figure 5: LLE swiss roll test case

2.3 Projection vectors and the scatter matrix

In our experiment we set k the number of neighbors as 30. Here we also use all 624 images as our input data. The three LLE projection vectors are as figure 6 shown.

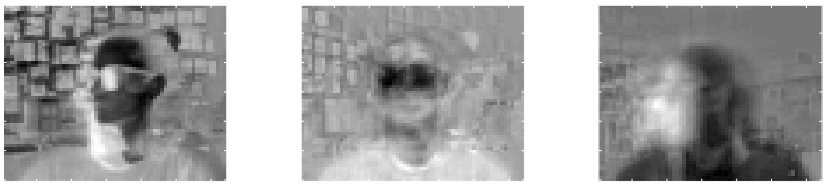


Figure 6: three projection vectors of PCA

The overall scatter matrix of LLE is as Figure 7 shown.

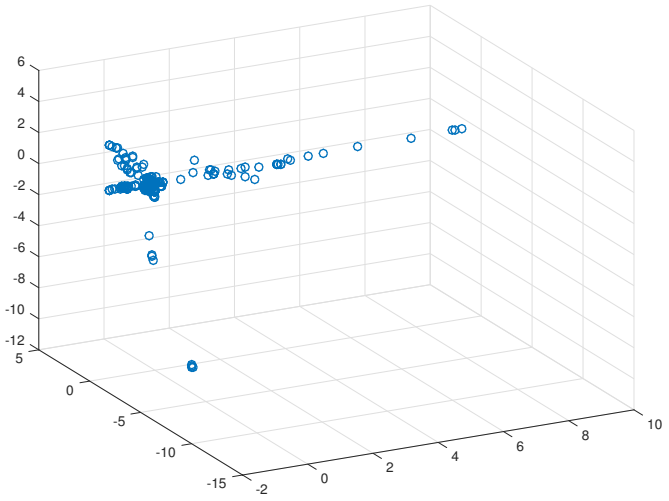
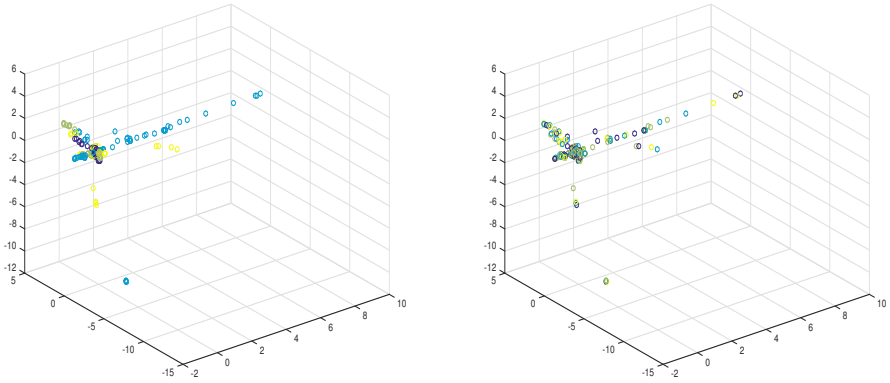


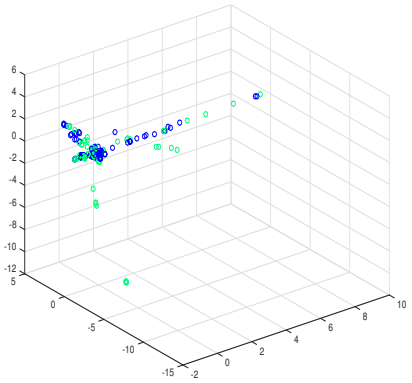
Figure 7: PCA scatter matrix of all 624 images

We also colors the LLE scatter matrix based on 4 poses, 4 emotions, sunglasses or not, and 20 names. They are shown in Figure 8.

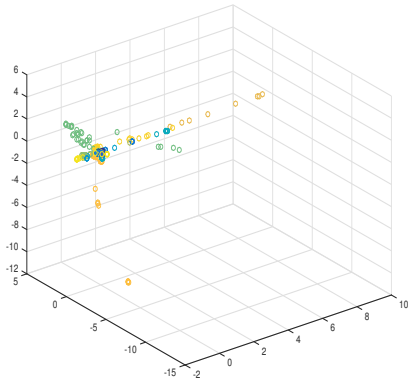


(a) poses

(b) emotions



(c) glasses or not



(d) names

Figure 8: LLE scatter matrix based on poses, emotions, glasses or not and names

3 Kernel PCA

We implement kernel on PCA. We use both polynomial kernel and gaussin kernel. The polynomial kernel equation is

$$\kappa(x, y) = (x^T y + c)^d, \tag{9}$$

where c is a constant. The Gaussin kernel equation is

$$\kappa(x, y) = \exp(-||x - y||^2/2\sigma^2) \tag{10}$$

where σ is a parameter. Denote kernel matrix

$$K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j) \tag{11}$$

The steps of kernel PCA can be summarized as:
1. Construct the kernel matrix K from the data set $X = \{x_i\}$ using

$$K(i, j) = \kappa(x_i, x_j) \tag{12}$$

2. Computer the Gram matrix

$$\tilde{K} = K - I_n K - K I_n + I_n K I_n \tag{13}$$

where I_n is the n by n matrix with all elements equal to $1/n$. This step is used when dataset does not have zero mean.
3. Solve eigenvalue λ_k and eigenvector v_k for $\tilde{K}$.
4. Computer the kernel principle components

$$y_k(x) = \Phi(x)^T v_k \tag{14}$$

3.1 Scatter matrix

The scatter matrix of gaussin kernel PCA is shown in Figure 9. Here we set the σ=2500.

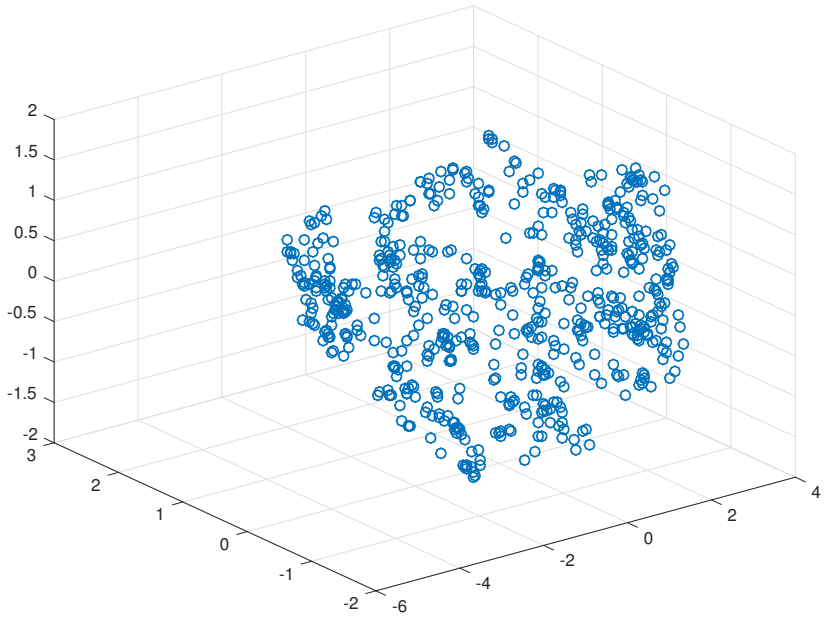


Figure 9: Gaussin kernel PCA Scatter matrix

The scatter matrix of ploynomial kernel PCA is shown in Figure 10. Here we set the d=4.

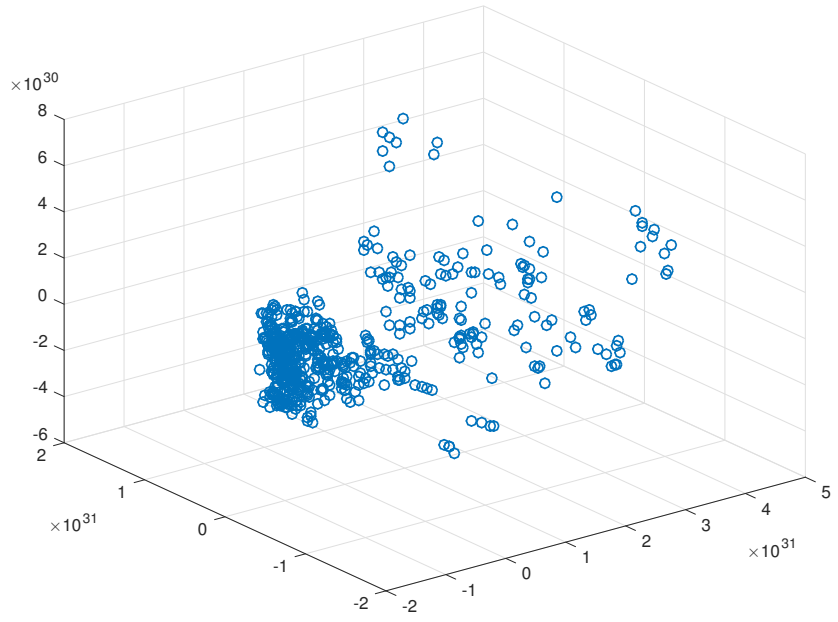


Figure 10: polynomial kernel PCA Scatter matrix

The kernel PCA method first rewrite the dataset into a higher dimensional space, then project the data its a low dimensional space. By applying the kernel, the dataset is changing from linear to non-linear.