

MATH3290 Mathematical Modeling

2016/17 Tutorial 5

12th October 2016

Outline

1 Principal Component Analysis

- Framework of PCA
- Example of PCA
- Details of Implementation

2 k-means Algorithm

- Problem setting
- Implementation

Given a set of data points $x_1, \dots, x_n \in \mathbb{R}^d$, define the $d \times d$ matrix:

$$Q = \frac{1}{n} \sum_{j=1}^n \tilde{x}_j \tilde{x}_j^T, \quad \tilde{x}_j = x_j - m$$

Compute the eigenvectors u_k and eigenvalues λ_k , where $k = 1, \dots, d$ and $m = \frac{1}{n} \sum_{j=1}^n x_j$ is the mean. Assume that $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d$, and the corresponding eigenvectors are u_1, u_2, \dots, u_d .

- Choose k principal directions u_1, \dots, u_k .
- Compute the projection of \tilde{x}_j to these eigenvectors, where $j = 1, \dots, n$, that is

$$c_{js} := \tilde{x}_j^T u_s, \quad s = 1, \dots, k.$$

Storage is $n \times k + d \times k + d$.

- Recover the data using

$$\hat{x}_j = \sum_{s=1}^k c_{js} u_s + m, \quad j = 1, \dots, n$$

- Compute the relative error e_i as follow:

$$e_i = \frac{\|x_j - \hat{x}_j\|}{\|x_j\|}, \quad j = 1, \dots, n.$$

Consider the 10 images of dimension 61×80 . We want to complete the MATLAB file **Q2.m**. First 9 images are stored in the matrix X .

- (a) Perform **PCA** on X .
- (b) Find the four largest eigenvalues and show the corresponding eigenvectors (reshape it into the dimensions of the original image).
- (c) Compute the relative error for data compression using only the eigenvectors in (a).

- (a) Read the raw data and form the matrix X . To obtain Q , we need to calculate the mean vector m and subtract it from X .

```
X = zeros(61*80,9);  
for i = 1:9  
    I = double(imread([num2str(i, '%02d') '.gif']))/256;  
    X(:,i) = I(:);  
end  
av = mean(X')';  
X_tilde = X-repmat(av,1,9);  
% Use the built-in function eig  
Q = (X_tilde) * (X_tilde)' /9;  
[U,L] = eig(Q);  
L = sort(diag(L));
```

Figure: Read data and forming Q .

- (b) Since the build-in function **eig** has already sorted the eigenvalues, we take the last four largest entries from vector L . Also, we plot the corresponding eigenvectors as images. Those eigenvalues are:

$$\lambda_1 = 108.2336, \quad \lambda_2 = 69.1148,$$

$$\lambda_3 = 23.6661 \quad \text{and} \quad \lambda_4 = 9.5920.$$

```
f4 = L(end-3:end)
V = U(:,end-3:end);
for i = 1:4
    subplot(2,2,i);
    imagesc(reshape(V(:,i),[61 80]));
end
```

Figure: Choose principal directions.

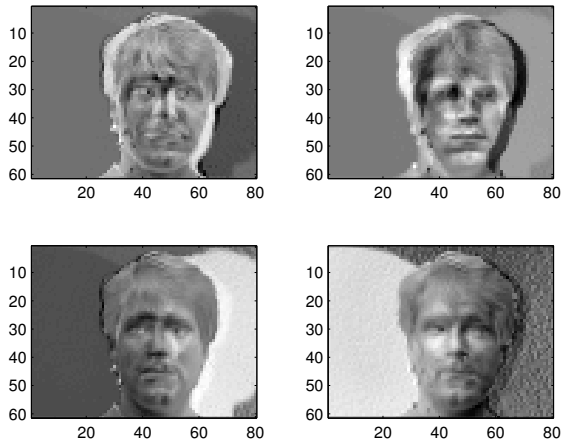


Figure: Corresponding eigenvectors.

- (c) We calculate the projection of X_tilde , that is the matrix of coefficients C . Also, recover the data and denote it as X_appro . Then, we compute the relative error.

```
C = (X_tilde)' * V;  
X_appro = V*C' + repmat(av,1,9);  
error = zeros(1,9);  
for i = 1:9  
    error(1,i) = norm(X(:,i)-X_appro(:,i))/norm(X(:,i));  
end
```

Figure: Obtain projection and error.

Problem setting

We use a data set, which is different to the one in **Q3.m**, as follow:

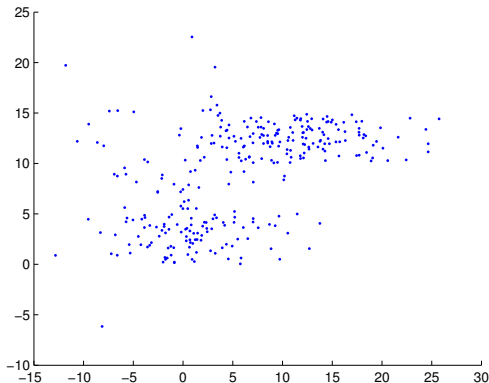


Figure: Data set.

Following the instructions in **Q3.m** and **Lecture notes P.39-40**, we can complete the code as follow:

```
load data.mat;
P = [P, P(:,1)*0];
c = [15 20; -5 -5; -5 10]; %Initial Clusters
num_of_clusters = size(c,1);
for i = 1:size(P,1)
    P(i,3) = closest(P(i,1:2), c);
end
c = update_centers(P, num_of_clusters); %Update centers
sse_old = inf; sse_new = sse(P, c);
count = 1; %Count the number of iteration
while( sse_new < sse_old )
    sse_old = sse_new;
    for i = 1:size(P,1)
        P(i,3) = closest(P(i,1:2), c);
    end
    c = update_centers(P, num_of_clusters);
    sse_new = sse(P, c);
    count = count + 1;
end
```

Figure: Sample code of clustering.

Finally, the result of clustering is as follow. Centers of clusters are:

$$c_1 = (14.7159, 11.9511),$$

$$c_2 = (-0.5844, 3.9448),$$

$$c_3 = (5.1142, 12.6889).$$

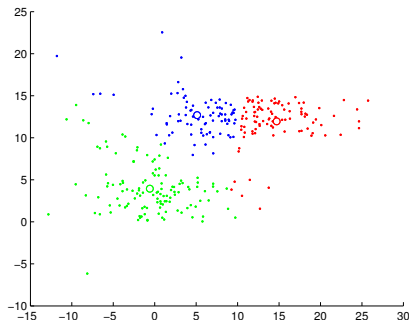


Figure: Data set.