Principal Component Analysis (PCA)

https://github.com/Huafeng-XU

● 1.Significance of PCA

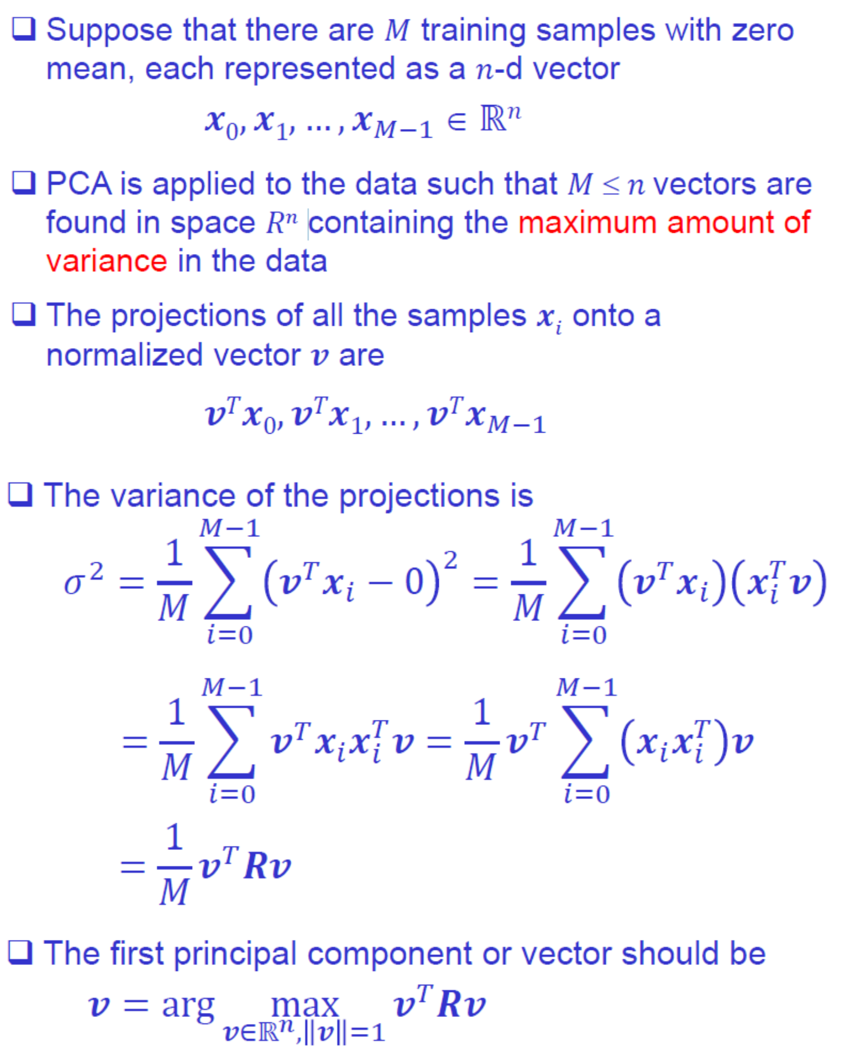
Seek projections that best represent the data in least-squares sense. (Find components that are useful for representing data).

● 2.Formula description

In this part, I will show you the Formula description of PCA from my teacher showed for me. Actually, it is uneasy to understand its principle.

Feeling it difficult? Me too. If you just want to know how PCA work on matlab, you can skip this part and watch part 3.

One way to derive PCA:



- ❑ Suppose that there are M training samples with zero mean, each represented as a n-d vector

$$\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_{M-1} \in \mathbb{R}^n$$

- ❑ PCA is applied to the data such that $M \leq n$ vectors are found in space R^n containing the maximum amount of variance in the data

- ❑ The projections of all the samples $\boldsymbol{x}_i$ onto a normalized vector $\boldsymbol{v}$ are

$$\boldsymbol{v}^T\boldsymbol{x}_0, \boldsymbol{v}^T\boldsymbol{x}_1, \dots, \boldsymbol{v}^T\boldsymbol{x}_{M-1}$$

- ❑ The variance of the projections is

$$\sigma^2 = \frac{1}{M}\sum_{i=0}^{M-1}\left(\boldsymbol{v}^T\boldsymbol{x}_i - 0\right)^2 = \frac{1}{M}\sum_{i=0}^{M-1}\left(\boldsymbol{v}^T\boldsymbol{x}_i\right)\left(\boldsymbol{x}_i^T\boldsymbol{v}\right)$$

$$= \frac{1}{M}\sum_{i=0}^{M-1}\boldsymbol{v}^T\boldsymbol{x}_i\boldsymbol{x}_i^T\boldsymbol{v} = \frac{1}{M}\boldsymbol{v}^T\sum_{i=0}^{M-1}\left(\boldsymbol{x}_i\boldsymbol{x}_i^T\right)\boldsymbol{v}$$

$$= \frac{1}{M}\boldsymbol{v}^T\boldsymbol{R}\boldsymbol{v}$$

- ❑ The first principal component or vector should be

$$\boldsymbol{v} = \arg\max_{\boldsymbol{v}\in\mathbb{R}^n, \|\boldsymbol{v}\|=1} \boldsymbol{v}^T\boldsymbol{R}\boldsymbol{v}$$

- This is a constrained optimization problem, which can be solved by Lagrange multiplier:

$$g(\boldsymbol{v}, \lambda) = \boldsymbol{v}^T \boldsymbol{R} \boldsymbol{v} - \lambda(\boldsymbol{v}^T \boldsymbol{v} - 1)$$

$$\frac{\partial g}{\partial \boldsymbol{v}} = \boldsymbol{v}^T\left(\boldsymbol{R} + \boldsymbol{R}^T\right) - 2\lambda \boldsymbol{v}^T = \boldsymbol{0}^T$$

$$\Rightarrow \boldsymbol{R}\boldsymbol{v} - \lambda \boldsymbol{v} = \boldsymbol{0} \text{ or } \boldsymbol{R}\boldsymbol{v} = \lambda \boldsymbol{v}$$

- The variance: $\sigma^2 = \frac{1}{M}\boldsymbol{v}^T \boldsymbol{R} \boldsymbol{v} = \lambda \frac{1}{M} \boldsymbol{v}^T \boldsymbol{v} = \frac{\lambda}{M}$

- The scatter matrix: $\boldsymbol{R} = [\boldsymbol{x}_0 \quad \cdots \quad \boldsymbol{x}_{M-1}] \begin{bmatrix} \boldsymbol{x}_0^T \\ \vdots \\ \boldsymbol{x}_{M-1}^T \end{bmatrix}$

  If $\boldsymbol{X} = [\boldsymbol{x}_0 \quad \cdots \quad \boldsymbol{x}_{M-1}]$, then $\boldsymbol{R} = \boldsymbol{X}\boldsymbol{X}^T$

Another way to derive PCA:

- The elements in $\boldsymbol{x}$ are highly correlated,

  i.e. $E[x_i x_j]$ is large

- $\boldsymbol{x}$ can be used as a feature vector for representing the face image, but it is desirable to generate features that are mutually uncorrelated in order to avoid information redundancies
  - After the following transformation:

    $\boldsymbol{y} = \boldsymbol{U}^T \boldsymbol{x}$,

    where $\boldsymbol{U} = [\boldsymbol{u}_0\, \boldsymbol{u}_1 \, \ldots \, \boldsymbol{u}_{n-1}]$ and $n = n_1 n_2$,

    $E[y_i y_j] = 0, \ i \neq j$, i.e. y_i and y_j are uncorrelated

- $\boldsymbol{R}_x$ is a symmetric matrix, so its eigenvectors are mutually orthogonal

- If matrix $\boldsymbol{U}$ is chosen so that its columns $\boldsymbol{u}_i$ are the orthonormal eigenvectors of $\boldsymbol{R}_x$, then $\boldsymbol{R}_y$ is diagonal

  i.e. $\boldsymbol{R}_y = \boldsymbol{U}^T \boldsymbol{R}_x \boldsymbol{U} = \boldsymbol{\Lambda}$

  where $\boldsymbol{\Lambda}$ is the diagonal matrix having the elements on its diagonal the respective eigenvalues λ_i, $i = 0, 1, \ldots, n-1$, of $\boldsymbol{R}_x$

3.Sample and code on matlab

Consider the following two sets of training data, where the samples are drawn randomly:

$X_1 = \{(2,6),(3,4),(3,5),(4,3),(5,2),(5,4),(6,0),(6,1)\}$

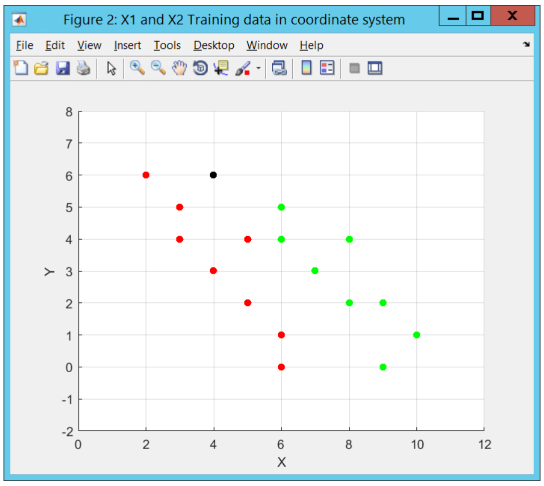
$X_2 = \{(6,4),(6,5),(7,3),(8,2),(8,4),(9,0),(9,2),(10,1)\}$

Assume the $q = (4,6)$ is a query input. You need to find out q belongs to which class.

Question: By using principal component analysis (PCA) method to classify the query input q.

Method:

- **Step 1**: Plot X1 and X2 Training data in coordinate system using Matlab. X_1 marked at red, X_2 marked at green and q marked at black.



You can easily use the code below at Matlab to achieve this:

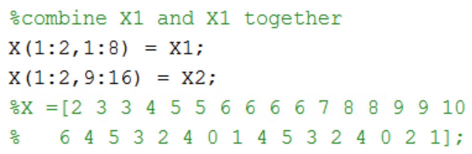
```
clear all;
%row 1 represent x in coordinate system.
%row 2 represent y in coordinate system.
X1=[2 3 3 4 5 5 6 6
   6 4 5 3 2 4 0 1];
X1_x = X1(1,:);
X1_y = X1(2,:);

X2=[6 6 7 8 8 9 9 10
   4 5 3 2 4 0 2 1];
X2_x = X2(1,:);
X2_y = X2(2,:);

%Plot X1 and X2 Training data in coordinate system.
figure('Name','X1 and X2 Training data in coordinate system')
scatter(X1_x,X1_y,'filled','red')
hold on
scatter(X2_x,X2_y,'filled','green')
scatter(4,6,'filled','black')
grid on
axis([0 12 -2 8])
xlabel('X');
ylabel('Y');
```

Hoping that you can type the code yourself. But if you think this may waste you time. You can just download all the code from my **GitHub**: https://github.com/Huafeng-XU. It is in the ***Principal-Component-Analysis-PCA-**/Code*. Actually, step 1 is not necessary. However, it is good for us to observe the training data firstly.

➢ **Step 2**: PCA is unsupervised learning method. Thus, we need to combine X1 and X1 together. Hence, we have 16 training samples X now.

$$X = \{\begin{bmatrix} (2,6),(3,4),(3,5),(4,3),(5,2),(5,4),(6,0),(6,1) \\ (6,4),(6,5),(7,3),(8,2),(8,4),(9,0),(9,2),(10,1) \end{bmatrix}\}$$

Using the Code below you can achieve this easily.



```
%combine X1 and X1 together
X(1:2,1:8) = X1;
X(1:2,9:16) = X2;
%X =[2  3  3  4  5  5  6  6  6  6  7  8  8  9  9  10
%    6  4  5  3  2  4  0  1  4  5  3  2  4  0  2  1];
```

Calculate the mean value of X:

$$mean = \frac{1}{16} \cdot X$$

The code of Matlab is below:

```
mean = 1/16 * sum(X');
```

➢ **Step 3**: Calculate XP:

$$XP = X - mean$$

The code of Matlab is below:

```
XP = X - mean';
```

➢ **Step 4**: Calculate the covariance matrix R:

$$R = \frac{1}{16} XP * XP'$$

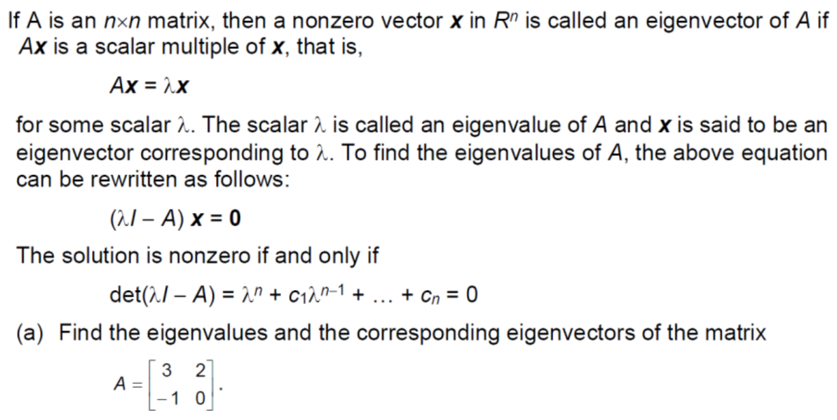
The code of Matlab is below:

```
R = [0  0
     0  0];
for i = 1:16
    R = R + XP(:,i) * XP(:,i)';
end
R = 1/16 * (R);
```

➢ **Step 5**: Find the eigenvalues and eigenvector:
In this step, I want to show you clearly what is **eigenvalues** and **eigenvector** firstly. Because lots of student do not understand this well.

I will like to show you an example that showed by my teacher about the eigenvalues and eigenvector. I hope this example below can help you understand eigenvalues and eigenvector.

Example:


If A is an $n \times n$ matrix, then a nonzero vector $\boldsymbol{x}$ in R^n is called an eigenvector of A if $A\boldsymbol{x}$ is a scalar multiple of $\boldsymbol{x}$, that is,

$$A\boldsymbol{x} = \lambda \boldsymbol{x}$$

for some scalar λ. The scalar λ is called an eigenvalue of A and $\boldsymbol{x}$ is said to be an eigenvector corresponding to λ. To find the eigenvalues of A, the above equation can be rewritten as follows:

$$(\lambda I - A)\,\boldsymbol{x} = \boldsymbol{0}$$

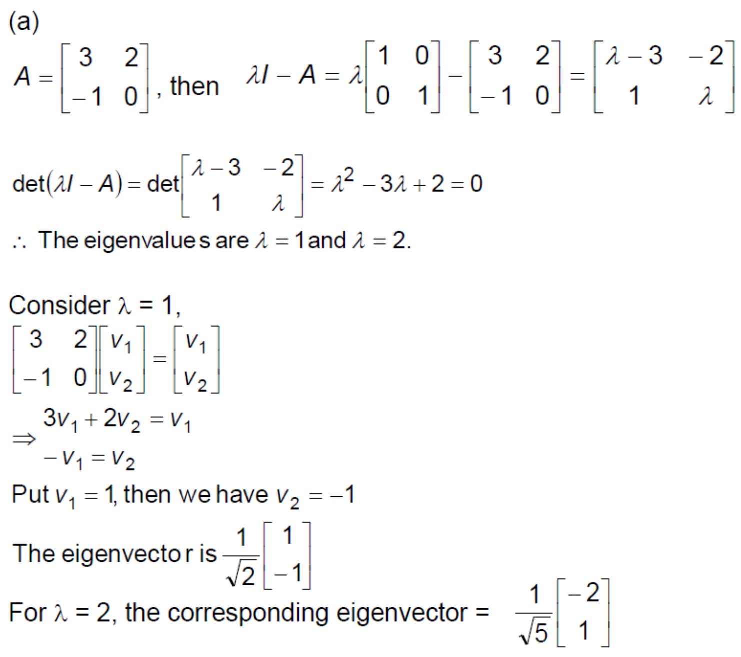
The solution is nonzero if and only if

$$\det(\lambda I - A) = \lambda^n + c_1\lambda^{n-1} + \ldots + c_n = 0$$

(a) Find the eigenvalues and the corresponding eigenvectors of the matrix

$$A = \begin{bmatrix} 3 & 2 \\ -1 & 0 \end{bmatrix}.$$

Solution:


(a)

$$A = \begin{bmatrix} 3 & 2 \\ -1 & 0 \end{bmatrix}, \text{ then } \quad \lambda I - A = \lambda\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 3 & 2 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} \lambda - 3 & -2 \\ 1 & \lambda \end{bmatrix}$$

$$\det(\lambda I - A) = \det\begin{bmatrix} \lambda - 3 & -2 \\ 1 & \lambda \end{bmatrix} = \lambda^2 - 3\lambda + 2 = 0$$

$\therefore$ The eigenvalues are $\lambda = 1$ and $\lambda = 2$.

Consider $\lambda = 1$,

$$\begin{bmatrix} 3 & 2 \\ -1 & 0 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

$$\Rightarrow \begin{array}{l} 3v_1 + 2v_2 = v_1 \\ -v_1 = v_2 \end{array}$$

Put $v_1 = 1$, then we have $v_2 = -1$

The eigenvector is $\dfrac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix}$

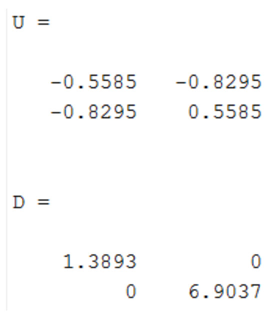
For $\lambda = 2$, the corresponding eigenvector = $\dfrac{1}{\sqrt{5}}\begin{bmatrix} -2 \\ 1 \end{bmatrix}$

After read this, I hope you understand eigenvalues and eigenvector well.

Come back to our example. We have already got the covariance matrix R. In Matlab, we do not need to do those complex calculations. You can use the code below in Matlab to get it.

```
[U,D] = eig(R)
```

Then, you will get the result below:


```
U =

   -0.5585   -0.8295
   -0.8295    0.5585


D =

    1.3893         0
         0    6.9037
```

For this result, U is the eigenvector and D is the eigenvalues.

For eigenvalue $\lambda 1 = 1.3893$, its eigenvector is $e1 = \begin{bmatrix} -0.5585 \\ -0.8295 \end{bmatrix}$.

For eigenvalue $\lambda 2 = 6.9037$, its eigenvector is $e2 = \begin{bmatrix} -0.8295 \\ 0.5585 \end{bmatrix}$.

- **Step 6**: Project X onto e2 to form the set Y. ($\lambda 2 = 6.9037 > \lambda 1 = 1.3893$, so we choose $\lambda 2$ and its eigenvector e2 to project)
  Note that: X was form by X1 and X2. Thus, this Y we assume that it is form by Y1 and Y2.

  The projection value:

$$Y = e2' \cdot XP$$

  Note that: XP can be the original sample X or Demeaned sample. We choose demeaned XP. This won't change the classification result.

  The code of Matlab is below:

```
e1 = U(:,1);
e2 = U(:,2);

%The projection value
Y = e2' * XP;
Y1 = Y(1,1:8);
Y2 = Y(1,9:16);
```

- **Step 7**: Using the minimum distance classifier, calculate the mean of the projected samples of each class:

$$meanY1 = \frac{1}{8} \cdot Y1$$
$$meanY2 = \frac{1}{8} \cdot Y2$$

  The code of Matlab is below:

```
meanY1 = 1/8 * sum(Y1);
meanY2 = 1/8 * sum(Y2);
```

- **Step 8**: Calculate the accuracy or the recognition rate of the minimum distance classifier, based on PCA.

$$Di = |Yi - m|$$

  For class1 ($w1$, data X1 or Y1):

$$X1Di_1 = |Y1i - meanY1|$$
$$\mathrm{X}1Di_2 = |Y1i - meanY2|$$

  IF $X1Di_1 < X1Di_2$, Then $Y1i$ belongs to class1, X1count ++.
  Otherwise, it belongs to class2.
  For class2($w2$, data X2 or Y2) is just the same.

The code of Matlab is below:

```
 %Find the accuracy
%For X1
X1D1 = abs(Y1-meanY1);
X1D2 = abs(Y1-meanY2);
X1count = size(find((X1D1-X1D2)<0),2);
%For X2
X2D1 = abs(Y2-meanY1);
X2D2 = abs(Y2-meanY2);
X2count = size(find((X2D1-X2D2)>0),2);
%The PCA accuracy.
PCA_accuracy = (X1count + X2count)/size(Y,2)
```

- **Step 9**: Find out the query input $q = (4,6)$ belongs to which class.

$$yq = e2' \cdot (q - mean)$$

The distance to Class1

$$qD1 = |yq - meanY1|$$

The distance to Class2

$$qD2 = |yq - meanY2|$$

IF $qD1 < qD2$, Then q belongs to class1;
Otherwise, q belongs to class2.

The code of Matlab is below:

```
 %Find out the query input q=(4,6) belongs to
which class
q = [4 6]';
yq = e2' * (q-mean);
%the distance to class1
qD1 = abs(yq-meanY1);
%the distance to class2
qD2 = abs(yq-meanY2);
if qD1 < qD2
    class =  1;
else
    class = 2;
end
class
```

Reference:
[1]. Pattern Recognition:Theory & Application Prof.Kennetth K.M.Lam.
[2]. Svante Wold, Kim Esbensen, Paul Geladi,Principal component analysis, Chemometrics and Intelligent Laboratory Systems,Volume 2, Issues 1–3,1987,Pages 37-52,ISSN 0169-7439