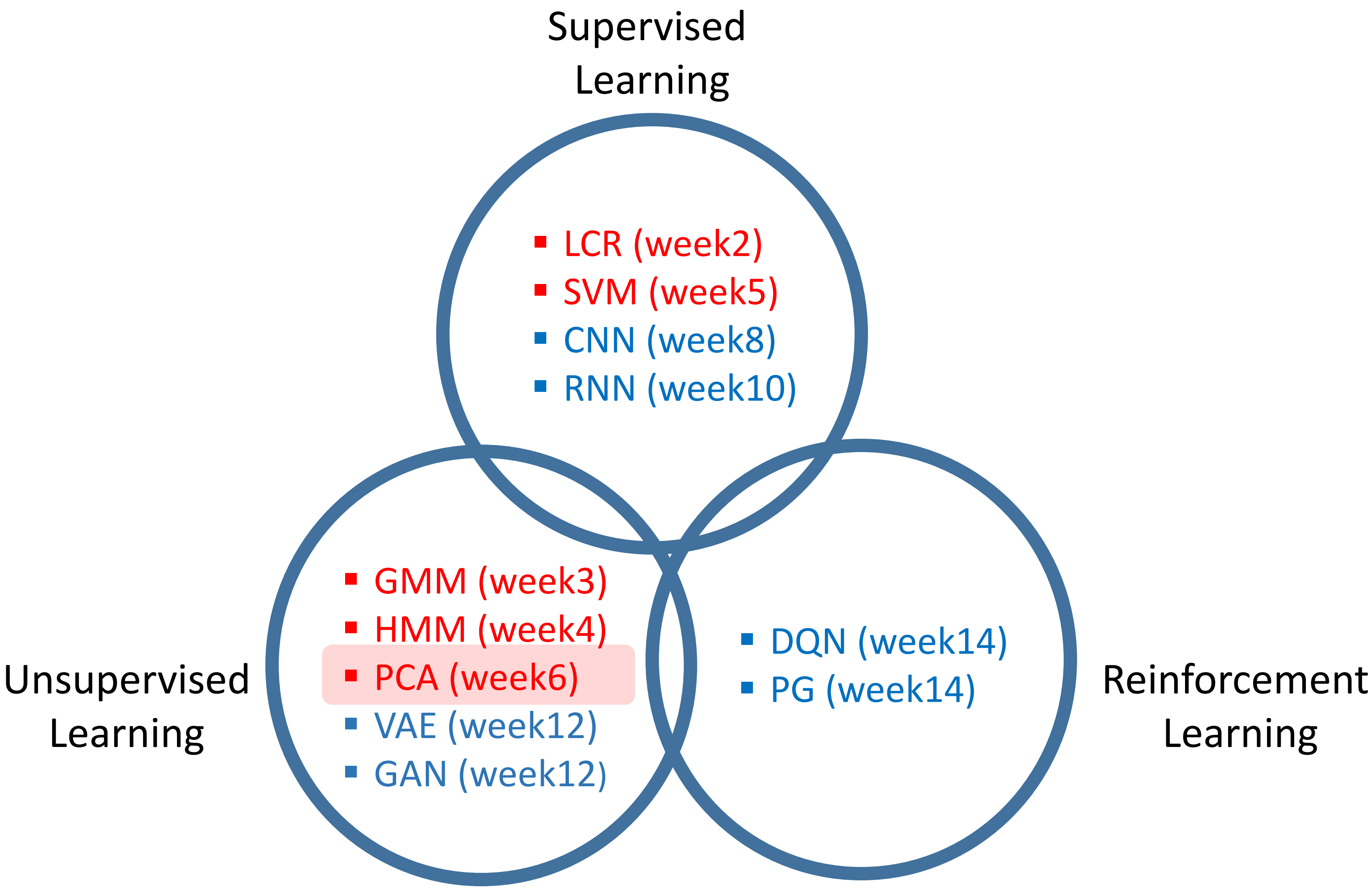
Practical Machine Learning

Lecture 6
Principal Components Analysis (PCA)

Dr. Suyong Eum

OSAKA UNIVERSITY

Where we are



Supervised Learning
LCR (week2)
SVM (week5)
CNN (week8)
RNN (week10)
Unsupervised Learning
GMM (week3)
HMM (week4)
PCA (week6)
VAE (week12)
GAN (week12)
DQN (week14)
PG (week14)
Reinforcement Learning

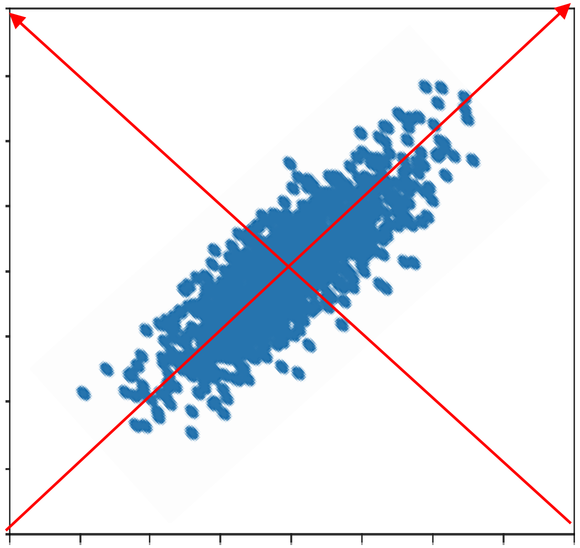
You are going to learn

- ❑ Why we need PCA
- ❑ How to obtain principal components
    - Eigen value decomposition and singular value decomposition
- ❑ SVD: data compression and visualization
- ❑ How to apply PCA for machine learning

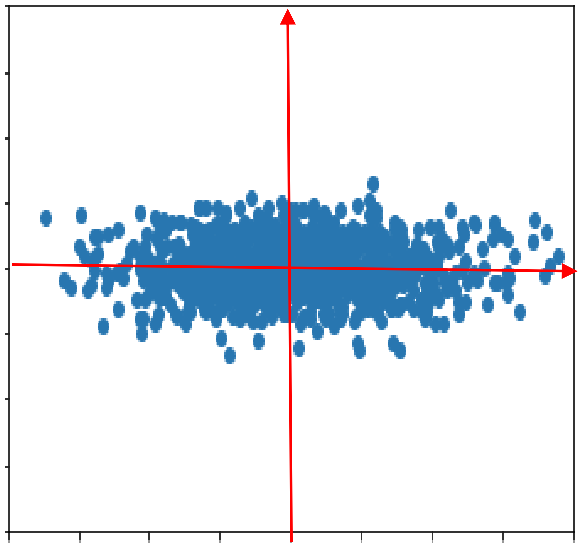
Principal Component Analysis (PCA): definition

A statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.

In Wikipedia:

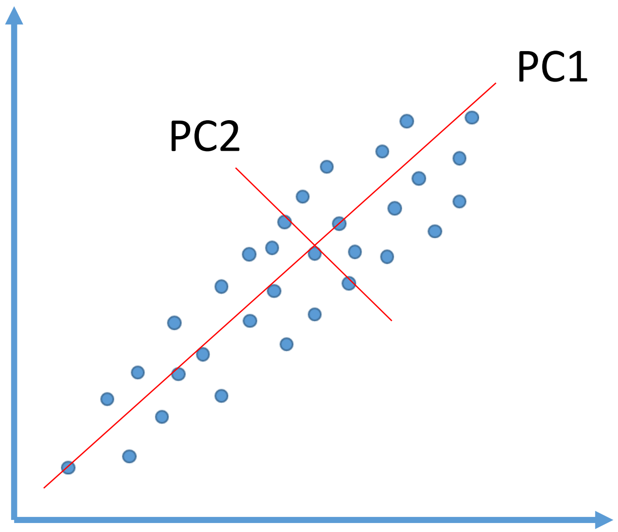


$$\begin{bmatrix} 8 & 1 \\ 1 & 8 \end{bmatrix}$$

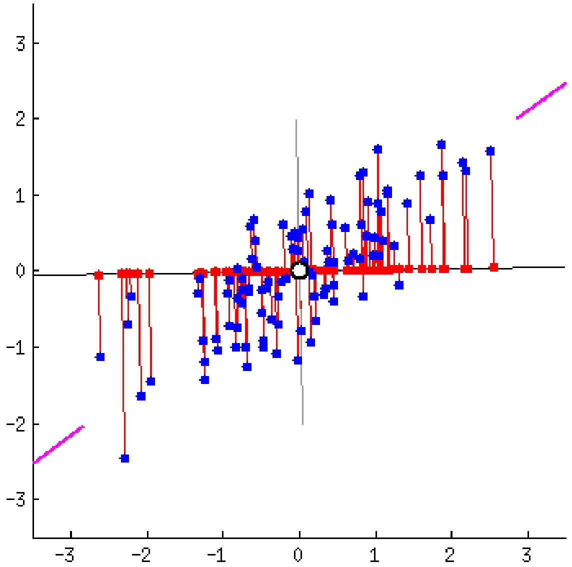


$$\begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}$$

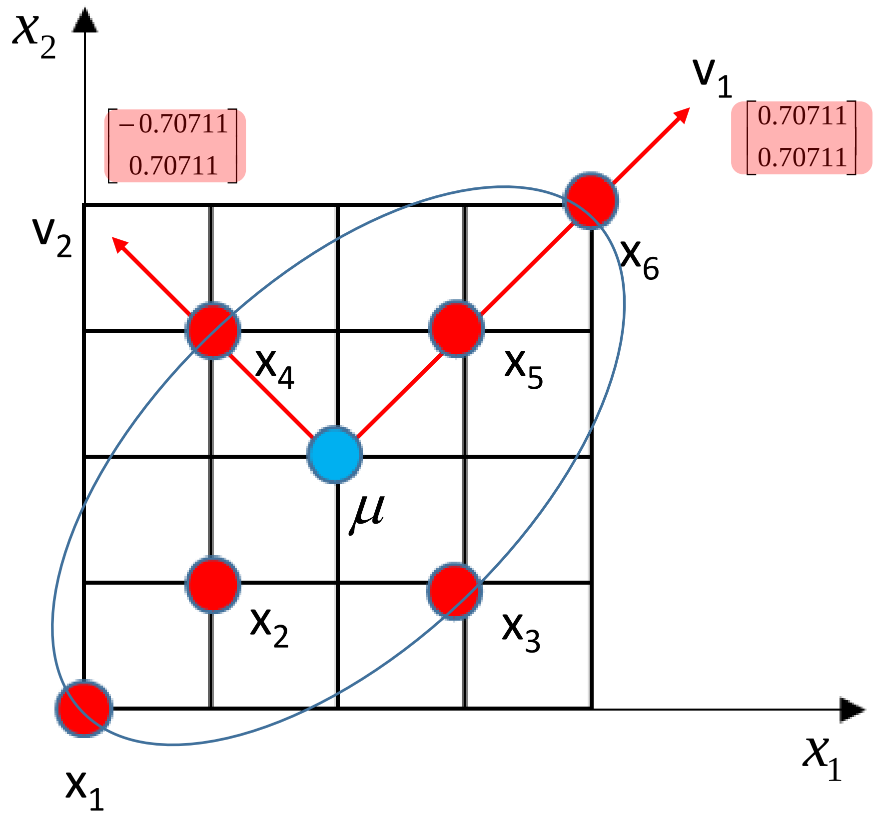
Principal Component Analysis (PCA): intuition



- ❑ How to select a principal component?
  - One that captures the largest variance of the data points.
- ❑ Why?
  - Because we want to clearly see how each data point is related (close) each other.
  - Then, which one (PC1 or PC2) is better?



How to find the principal components showing the largest variance?



$$X=\begin{bmatrix}x_1\\x_2\\x_3\\x_4\\x_5\\x_6\end{bmatrix}=\begin{bmatrix}-2&-2\\-1&-1\\1&-1\\-1&1\\1&1\\2&2\end{bmatrix}$$

- Distance to data points from the mean along the axis of "v_1"

$$=[-2\sqrt{2},-\sqrt{2},0,0,\sqrt{2},2\sqrt{2}] \qquad \text{variance}=4$$

- Distance to data points from the mean along the axis of "v_2"

$$=[0,0,-\sqrt{2},\sqrt{2},0,0] \qquad \text{variance}=0.8$$

- $$\mathrm{cov}(X)=\begin{bmatrix}2.4&1.6\\1.6&2.4\end{bmatrix}$$

variance along the axis of "x_1"

variance along the axis of "x_2"

- $$\mathrm{cov}(x)=V\Lambda V^T$$

```
>> [vec, val] = eig(cov(x))
vec =

  -0.70711   0.70711
   0.70711   0.70711

val =

Diagonal Matrix

   0.80000         0
         0   4.00000
```

V

Λ

Largest eigen value of the covariance matrix == largest variances in the data set?

- "m_i" shows the distance between 0 (mean) to the point where "x_i" is projected on the vector "V".

$$\mathrm{m}_i = \mathrm{x}_i \mathrm{v}$$

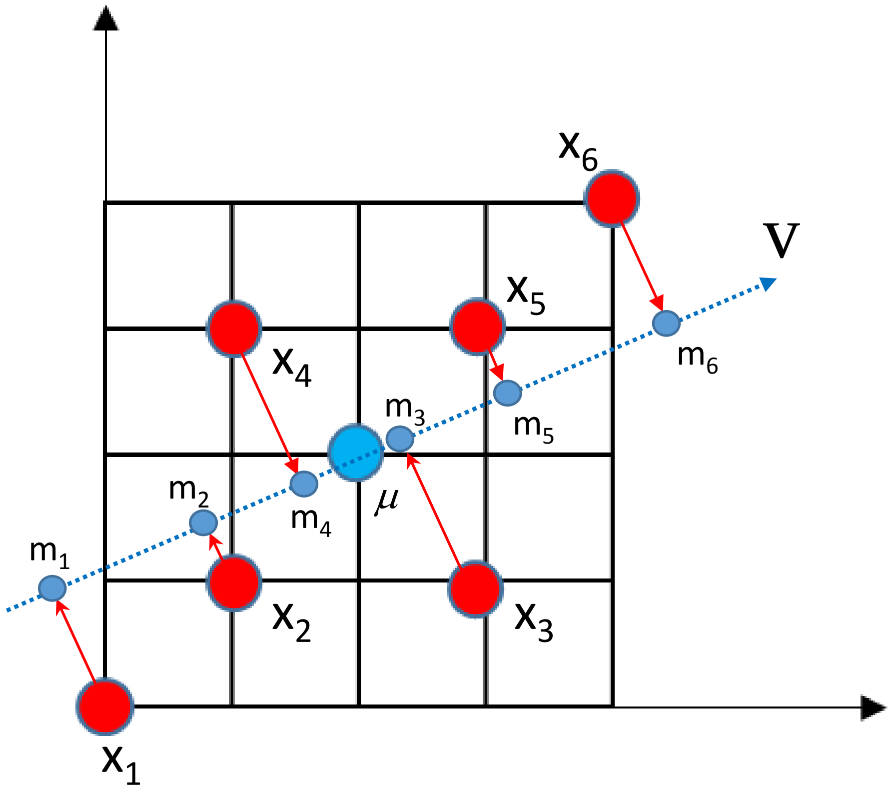
- Let's define the variance of data points "m"

$$\mathrm{var(m)} = \frac{1}{N-1}\sum_{i=1}^{N}(\mathrm{m_i} - \mu\mathrm{v})^2 = \frac{1}{N-1}\sum_{i=1}^{N}(\mathrm{x_i v})^2$$

- Let's maximize the variance with a constraint (v must be an unit vector). Then, see what it would be.

$$\max \frac{1}{N-1}\sum_{i=1}^{N}(\mathrm{x_i v})^2 \qquad s.t. \quad \sum_{\mathrm{i}=1}^{d}\mathrm{v_i^2} = 1$$

An Unit vector



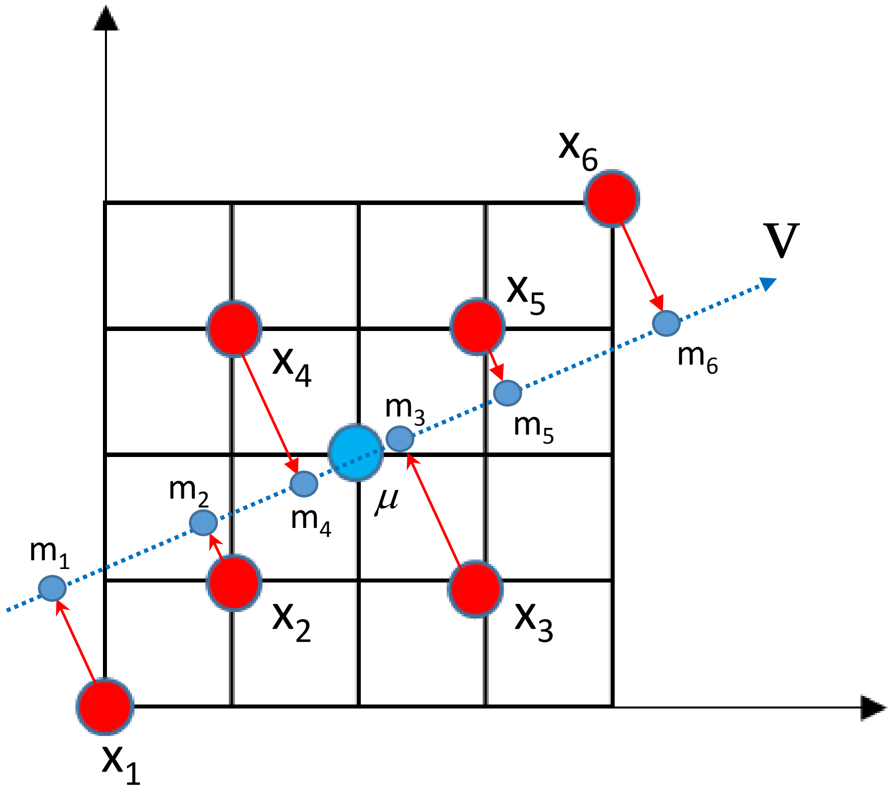
Largest eigen value of the covariance matrix == largest variances in the data set?

- Let's convert the constrained problem to unconstrained problem using Lagrange method (again!).

$$L(\mathrm{v}) = \frac{1}{N-1}\sum_{i=1}^{N}(\mathrm{x}_{\mathrm{i}}\mathrm{v})^2 - \lambda_i\left(\sum_{i=i}^{d}\mathrm{v}_i^2 - 1\right)$$

- We look for the vector "v" which maximizes the variance. Thus, differentiating the above with respect to "v"

$$\frac{\partial L(\mathrm{v})}{\partial \mathrm{v}} = \frac{2}{N-1}\sum_{i=1}^{N}(\mathrm{x}_{\mathrm{i}}\mathrm{v})(\mathrm{x}_{\mathrm{i}}) - 2\lambda_i\left(\sum_{i=i}^{d}\mathrm{v}_i\right) = 0$$

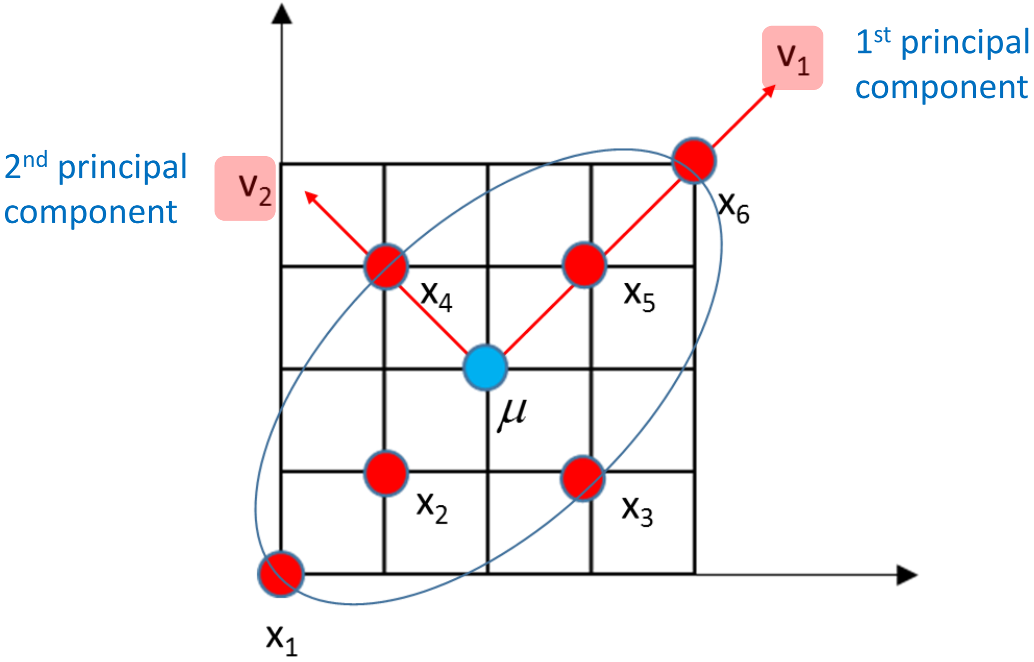


$$\underbrace{\left(\frac{1}{N-1}\sum_{i=1}^{N}(\mathrm{x}_{\mathrm{i}})^2\right)}_{\text{Covariance}}\mathrm{v} = \underbrace{\lambda}_{\text{Eigen value}}\mathrm{v}$$

- When "v" is selected to maximize the variance, covariance matrix becomes equivalent to its own eigen value.
- Eigen value has diagonal elements, which represent variances along eigen vectors – no correlation.

How to find the principal components showing the largest variance?

1) Find the covariance matrix of data points.
2) Obtain the eigen values and vectors of the covariance matrix: eigen decomposition.
3) Sort the eigen vectors in descending order in terms of their corresponding eigen values.
   - an eigen vector with the largest eigen value becomes the first principal component.




```
>> x
x =

  -2  -2
  -1  -1
   1  -1
  -1   1
   1   1
   2   2

>> cov(x)
ans =

   2.4000   1.6000
   1.6000   2.4000
```

```
>> [vec, val] = eig(cov(x))
vec =

  -0.70711   0.70711
   0.70711   0.70711

val =

Diagonal Matrix

   0.80000         0
         0   4.00000
```

2nd principal component: first column of vec (eigen value 0.80000)

1st principal component: second column of vec (eigen value 4.00000)

How to find the principal components showing the largest variance?

- Actually, there is a more convenient way of doing it (finding eigen vectors).
- It is called "Singular Value Decomposition" or SVD.

Eigen decomposition

$$\boxed{X^T X = V \Lambda V^T}$$

```
>> x
x =

  -2  -2
  -1  -1
   1  -1
  -1   1
   1   1
   2   2

>> cov(x)
ans =

   2.4000    1.6000
   1.6000    2.4000
```

```
>> [vec, val] = eig(cov(x))
vec =

  -0.70711   0.70711
   0.70711   0.70711

val =

Diagonal Matrix

   0.80000         0
         0   4.00000
```

```
>> [vec, val]=eig(transpose(x)*x)
vec =

  -0.70711   0.70711
   0.70711   0.70711

val =

Diagonal Matrix

   4.0000         0
        0   20.0000
```

Singular Value Decomposition (SVD)

$$\boxed{X = U \Sigma V^T}$$

$$\begin{aligned} X^T X &= (U \Sigma V^T)^T (U \Sigma V^T) \\ &= V \Sigma^T U^T U \Sigma V^T \\ &= V \Sigma^2 V^T \end{aligned}$$

$$\Lambda = \Sigma^2$$

$$\Lambda = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix} \qquad \Sigma = \begin{bmatrix} \sqrt{\lambda_1} & & 0 \\ & \ddots & \\ 0 & & \sqrt{\lambda_n} \end{bmatrix}$$

Eigen value　　Singular value

How to find the principal components showing the largest variance?

- Actually, there is a more convenient way of doing it (finding eigen vectors).
- It is called “Singular Value Decomposition” or SVD.

Eigen decomposition

$$\boxed{X^T X = V \Lambda V^T}$$

Singular Value Decomposition (SVD)

$$\boxed{X = U \Sigma V^T}$$

```
>> x
x =

   -2   -2
   -1   -1
    1   -1
   -1    1
    1    1
    2    2

>> cov(x)
ans =

    2.4000    1.6000
    1.6000    2.4000
```

```
>> [vec, val] = eig(cov(x))
vec =

  -0.70711   0.70711
   0.70711   0.70711

val =

Diagonal Matrix

   0.80000         0
         0   4.00000
```

```
>> [vec, val]=eig(transpose(x)*x)
vec =

  -0.70711   0.70711
   0.70711   0.70711

val =

Diagonal Matrix

   4.0000         0
        0   20.0000
```

$$\Lambda = \Sigma^2$$

$$4.4721^2 = 20$$

```
>> [U,S,V]=svd(x)
U =

  -0.63246   0.00000   0.30819  -0.30819   0.28637   0.57274
  -0.31623  -0.00000  -0.63635   0.63635   0.13426   0.26851
   0.00000  -0.70711   0.50000   0.50000   0.00000   0.00000
  -0.00000   0.70711   0.50000   0.50000  -0.00000  -0.00000
   0.31623   0.00000  -0.00399   0.00399   0.94140  -0.11720
   0.63246   0.00000  -0.00799   0.00799  -0.11720   0.76560

S =

Diagonal Matrix

   4.4721        0
        0   2.0000
        0        0
        0        0
        0        0
        0        0

V =

   0.70711  -0.70711
   0.70711   0.70711
```

Singular Value Decomposition (SVD): data compression

$X \in \mathbf{R}^{n\times m}$

$$\boxed{X = U\Sigma V^T}$$

$U \in \mathbf{R}^{n\times n}$

$\Sigma \in \mathbf{R}^{n\times m}$

$V \in \mathbf{R}^{m\times m}$

$$\begin{bmatrix} -2 & -2 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix} = \begin{bmatrix} -0.63246 & 0.00000 & 0.30819 & -0.30819 & 0.28637 & 0.57274 \\ -0.31623 & -0.00000 & -0.63635 & 0.63635 & 0.13426 & 0.26851 \\ 0.00000 & -0.70711 & 0.50000 & 0.50000 & 0.00000 & 0.00000 \\ -0.00000 & 0.70711 & 0.50000 & 0.50000 & -0.00000 & -0.00000 \\ 0.31623 & 0.00000 & -0.00399 & 0.00399 & 0.94140 & -0.11720 \\ 0.63246 & 0.00000 & -0.00799 & 0.00799 & -0.11720 & 0.76560 \end{bmatrix} \begin{bmatrix} 4.4721 & 0 \\ 0 & 2.0000 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0.70711 & -0.70711 \\ 0.70711 & 0.70711 \end{bmatrix}$$

X: 6x2 U: 6x6 Σ: 6x2 V: 2x2

Singular Value Decomposition (SVD): data compression

$$X \in \mathbf{R}^{n\times m}$$

$$\boxed{X = U\Sigma V^T}$$

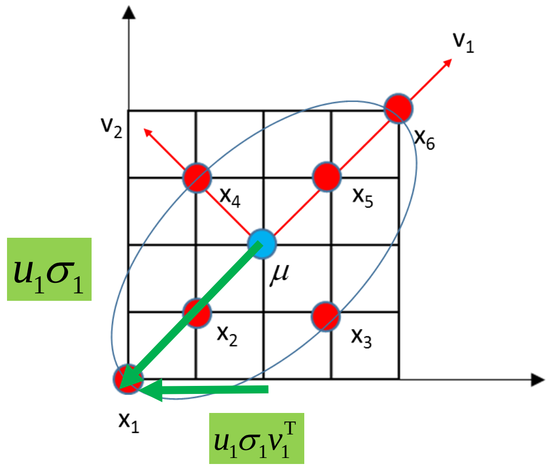
$$U \in \mathbf{R}^{n\times n}$$
$$\Sigma \in \mathbf{R}^{n\times m}$$
$$V \in \mathbf{R}^{m\times m}$$

$$\begin{bmatrix} -2 & -2 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix} = \begin{array}{cc} u_1 & \\ \left[\begin{matrix} -0.63246 & 0.00000 \\ -0.31623 & -0.00000 \\ 0.00000 & -0.70711 \\ -0.00000 & 0.70711 \\ 0.31623 & 0.00000 \\ 0.63246 & 0.00000 \end{matrix}\right] \end{array} \begin{array}{cc} \sigma_1 & \\ \left[\begin{matrix} 4.4721 & 0 \\ 0 & 2.0000 \end{matrix}\right] \end{array} \begin{array}{cc} v_1 & \\ \left[\begin{matrix} 0.70711 & -0.70711 \\ 0.70711 & 0.70711 \end{matrix}\right] \end{array}$$

X: 6x2 U: 6x2 Σ: 2x2 V: 2x2

$$X = u_1\sigma_1 v_1^T + u_2\sigma_2 v_2^T$$

- New coordination system which has two basis (v1 and v2)



Singular Value Decomposition (SVD): data compression

436

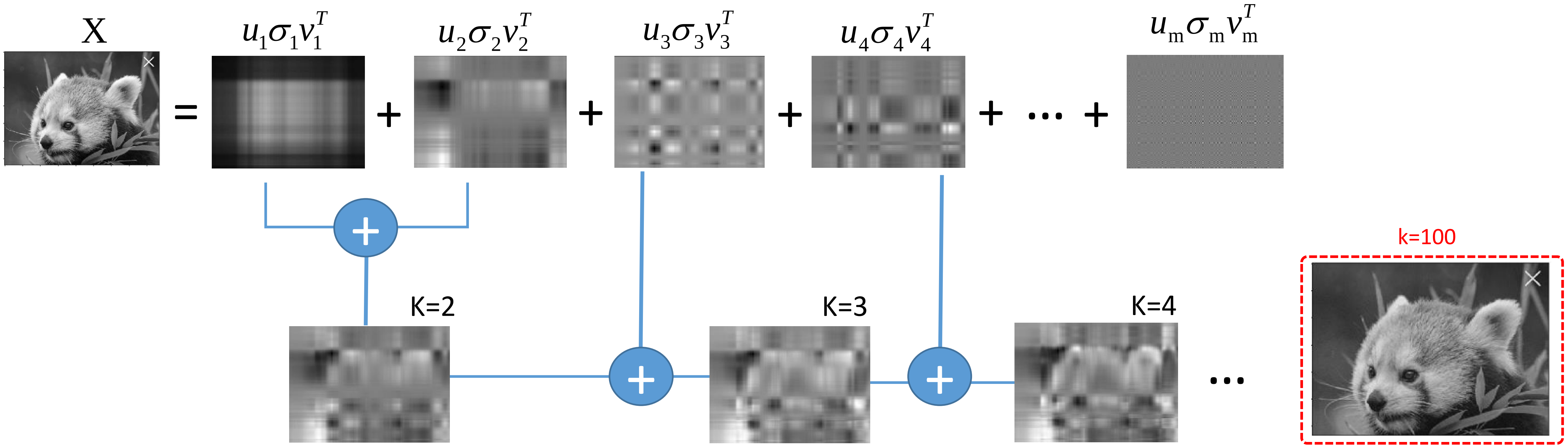
317

X: 317x436 = 138212

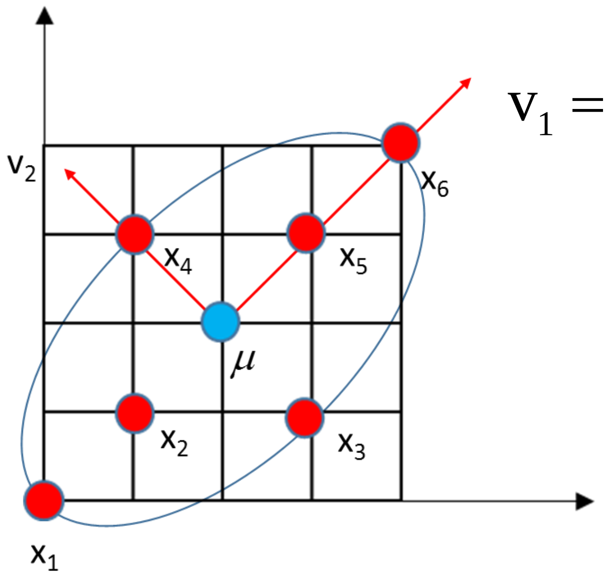
k=100

$$= \underbrace{u_1\sigma_1 v_1^{T} + u_2\sigma_2 v_2^{T} + u_3\sigma_3 v_3^{T} + \cdots}_{k=100} + u_m\sigma_m v^{T}_{m=317}$$

(317+1+436) x 100 = 75400

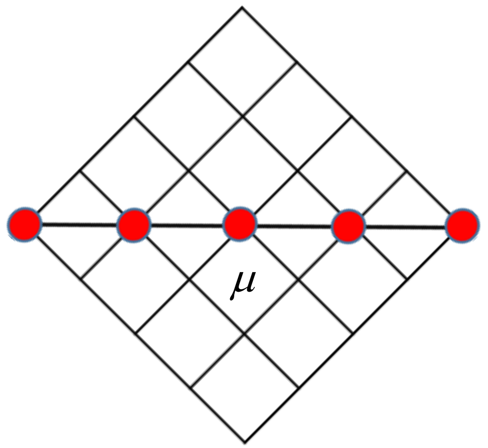


Back to PCA: dimension reduction



$$v_1 = \begin{bmatrix} 0.70711 \\ 0.70711 \end{bmatrix}$$

$$v_2 = \begin{bmatrix} -0.70711 \\ 0.70711 \end{bmatrix}$$



$$v_1 = \begin{bmatrix} 0.70711 \\ 0.70711 \end{bmatrix}$$

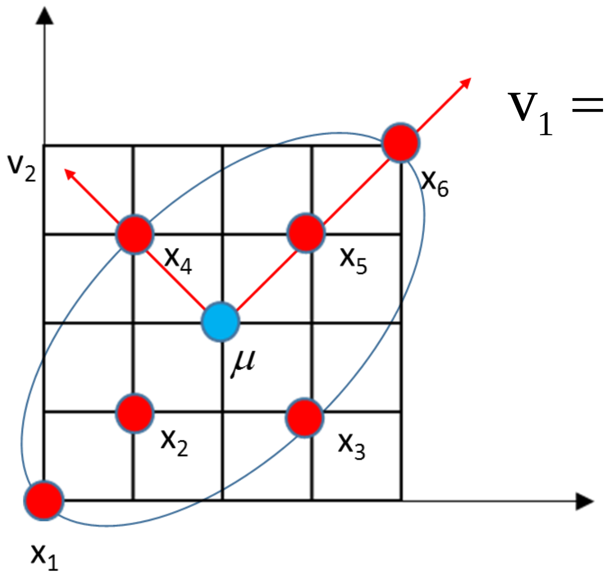
$$\begin{bmatrix} -2 & -2 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix} = \begin{bmatrix} \boxed{u_1} & u_2 \end{bmatrix} \begin{bmatrix} \boxed{\sigma_1} & \\ & \sigma_2 \end{bmatrix} \begin{bmatrix} \boxed{v_1^T} \\ v_2^T \end{bmatrix}$$

$$\mathrm{X} = \boxed{u_1 \sigma_1 v_1^{\mathrm{T}}} + u_2 \sigma_2 v_2^{\mathrm{T}}$$

$$\begin{bmatrix} -2\sqrt{2} & 0 \\ -\sqrt{2} & 0 \\ 0 & 0 \\ 0 & 0 \\ \sqrt{2} & 0 \\ 2\sqrt{2} & 0 \end{bmatrix}$$

2 dimension data points can be represented into one dimension space (v_1)

Back to PCA: dimension reduction

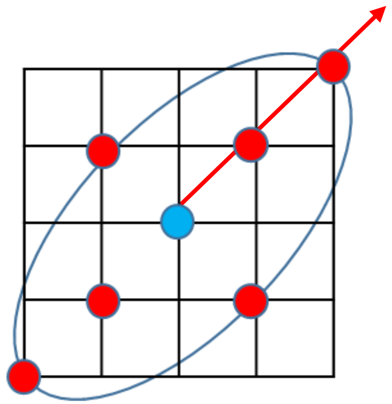


$$v_1 = \begin{bmatrix} 0.70711 \\ 0.70711 \end{bmatrix}$$

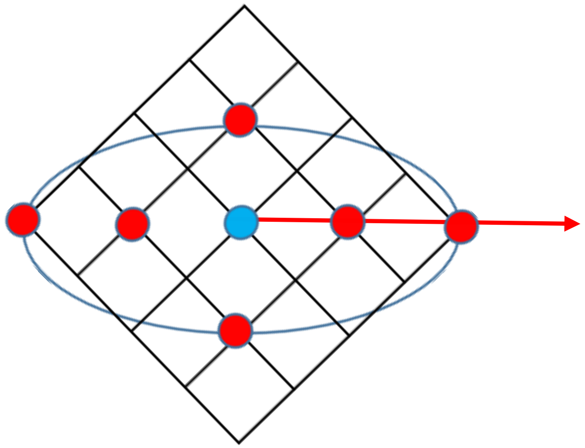
$$\begin{bmatrix} -2 & -2 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix} = \begin{bmatrix} \boxed{u_1} & u_2 \end{bmatrix} \begin{bmatrix} \boxed{\sigma_1} & \\ & \sigma_2 \end{bmatrix} \begin{bmatrix} \boxed{v_1^T} \\ v_2^T \end{bmatrix}$$

$$X = \boxed{u_1 \sigma_1 v_1^T} + u_2 \sigma_2 v_2^T$$

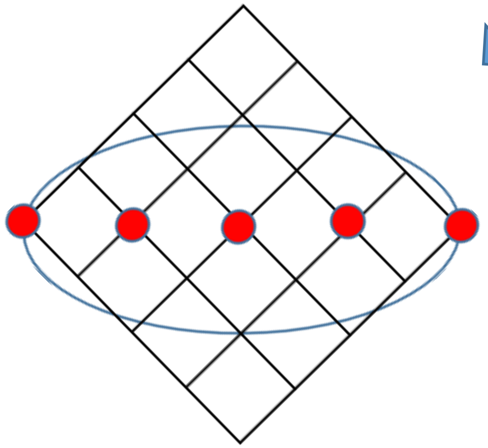
2 dimension data points can be represented into one dimension space (v_1)



1st Principal Component

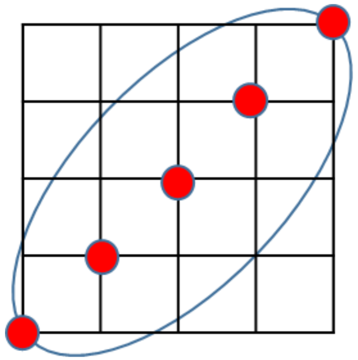


$$X\_rot = X \cdot V$$



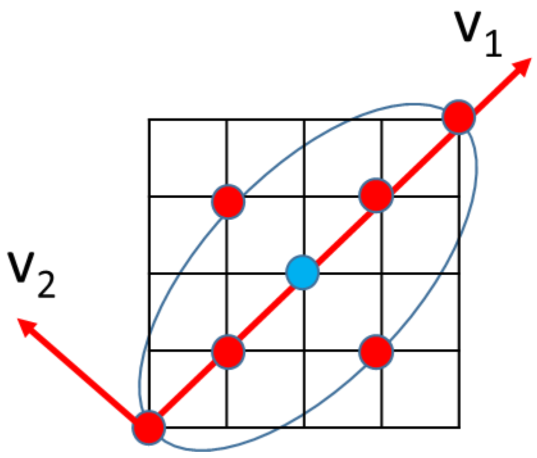
Set the "v_2" into zero

$$X\_rot\_zero = X\_rot \cdot V^{-1}$$

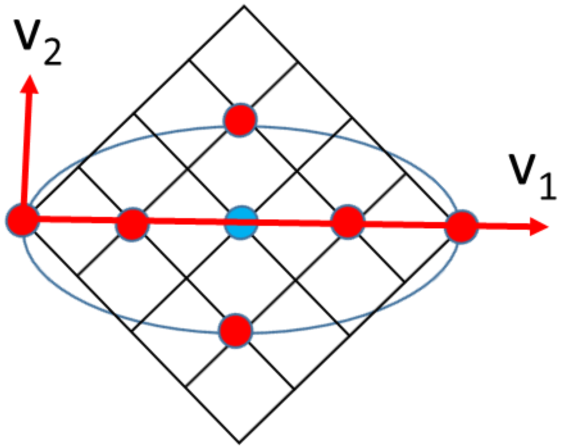


$$X' = X\_rot\_zero \cdot V^{-1}$$
$$= X\_rot\_zero \cdot V^T$$

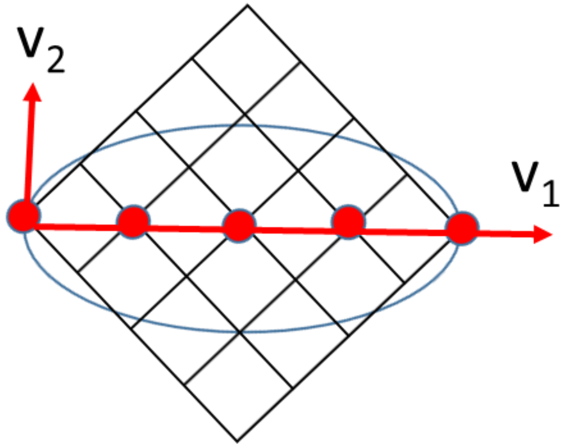
Back to PCA: example



$$X = \begin{bmatrix} -2 & -2 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}$$



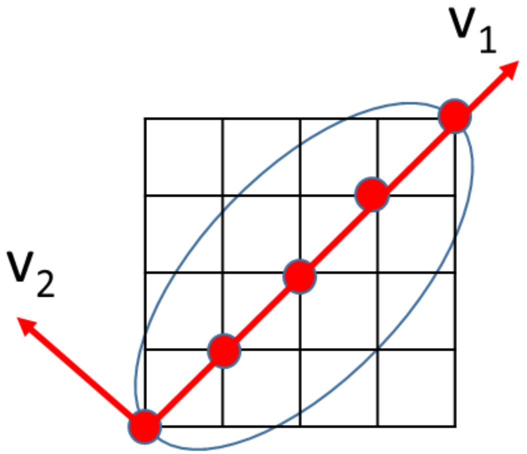
$$\begin{array}{ccc} \text{X\_rot} & & \text{X} \quad\quad \text{V} \\ \begin{bmatrix} -2\sqrt{2} & 0 \\ -\sqrt{2} & 0 \\ 0 & -\sqrt{2} \\ 0 & \sqrt{2} \\ \sqrt{2} & 0 \\ 2\sqrt{2} & 0 \end{bmatrix} & = & \begin{bmatrix} -2 & -2 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} \sqrt{2}/2 & -\sqrt{2}/2 \\ \sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix} \end{array}$$



Set the “v_2” elements into zero

X_rot_zero

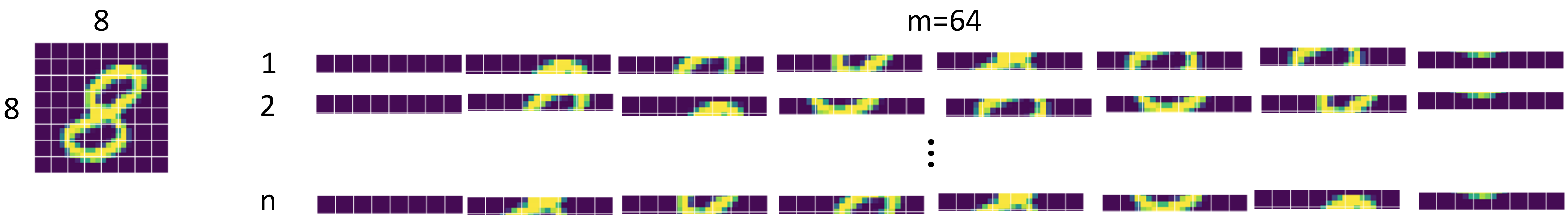
$$\begin{bmatrix} -2\sqrt{2} & 0 \\ -\sqrt{2} & 0 \\ 0 & 0 \\ 0 & 0 \\ \sqrt{2} & 0 \\ 2\sqrt{2} & 0 \end{bmatrix}$$

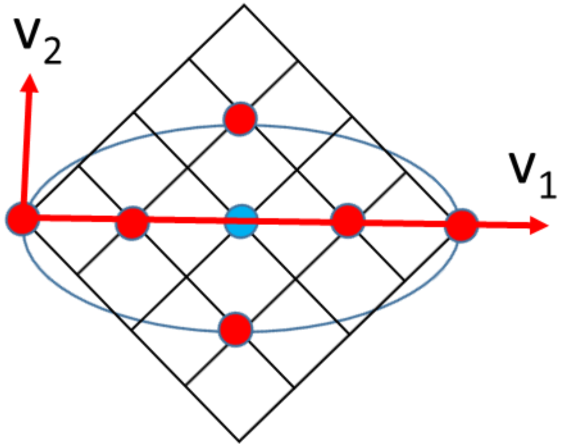


$$\begin{bmatrix} -2 & -2 \\ -1 & -1 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}$$

$$X' = \text{X\_rot\_zero} \cdot V^{-1} = \text{X\_rot\_zero} \cdot V^{T}$$

How to use PCA for machine learning?





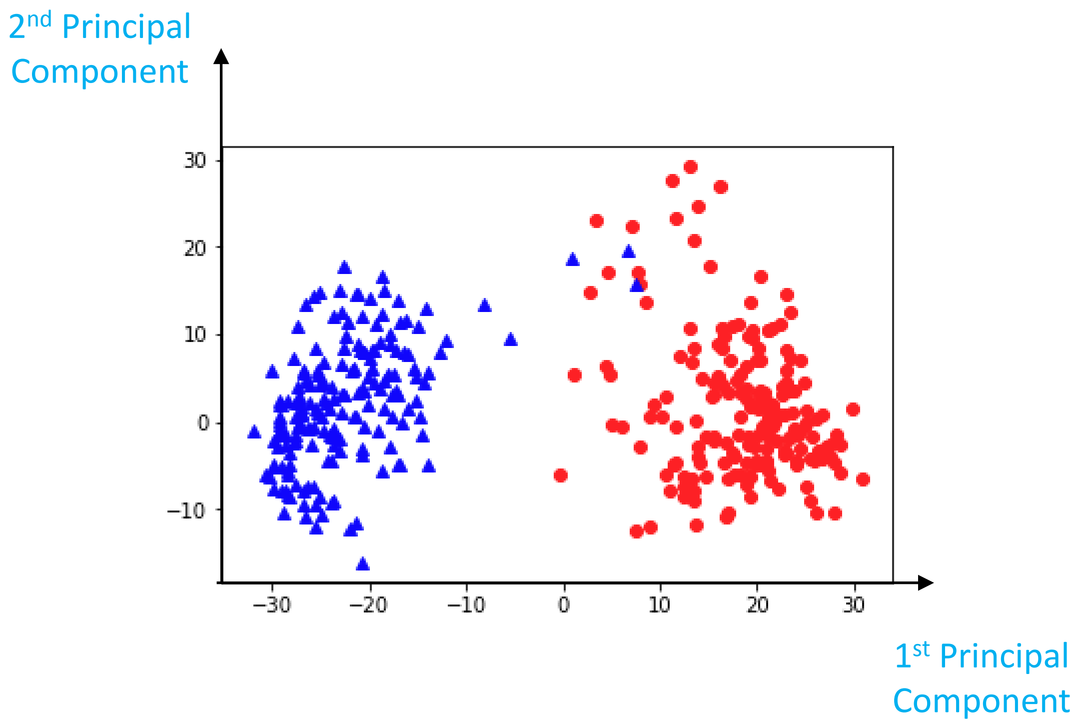
$$X_{rot} = X \cdot V$$

$$\begin{array}{cccc} v_1 & v_2 & & \\ \left[\begin{matrix} a_{11} & a_{12} & \cdots & a_{1m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{matrix}\right] \end{array}$$

A digit number with 64 dimension can be shown in 2 dimension space (v_1 and v_2).

How to use PCA for machine learning?

- Each digit number has 8 by 8 = 64 dimensions.
- After SVD, the first two principal components are selected, and the data points with 64 dimension are plotted in two dimension.



One that you need to be careful when carrying out PCA

- Centering the data before applying PCA.
- Normalizing or standardizing the data when features have different scale.

	population	area
Country 1	5*10^7	92
Country 2	2*10^7	74
…	…	…
Country n	5 *10^8	150

$$
\begin{array}{cc}
 & \begin{array}{cccc} \text{feature1} & \text{feature2} & \cdots & \text{feature m} \end{array} \\
\begin{array}{c} \text{Data 1} \\ \text{Data 2} \\ \vdots \\ \text{Data n} \end{array} &
\begin{bmatrix} \boxed{\begin{matrix} a_{11} \\ \vdots \\ a_{m1} \end{matrix}} & \begin{matrix} a_{12} \\ \vdots \\ a_{m2} \end{matrix} & \begin{matrix} \cdots \\ \vdots \\ \cdots \end{matrix} & \begin{matrix} a_{1n} \\ \vdots \\ a_{mn} \end{matrix} \end{bmatrix}
\end{array}
$$

That needs to be normalized

Backup Slides

Singular Value Decomposition (SVD): data compression

436

317

X: 317x436 = 138212

$$= u_1\sigma_1 v_1^T + u_2\sigma_2 v_2^T + u_3\sigma_3 v_3^T + \cdots + u_m\sigma_m v_{m=317}^T$$

k=100

(317+1+436) x 100 = 75400

k=100

```
img = imread("sample_BW.png")[:,:]
imshow(img)
show()

# Top 100
U,S,Vt = svd(img)
S = resize(S, [m,1])*eye(m,n)
imshow(dot(U[:,0:100], dot(S[0:100,0:100], Vt[0:100,:])))
show()
```