Practical Machine Learning

Workshop 2

Principal Components Analysis (PCA) & Support Vector Machine (SVM)

Dr. Suyong Eum

OSAKA UNIVERSITY

Principal Component Analysis (PCA)

Principal Component Analysis (PCA): definition

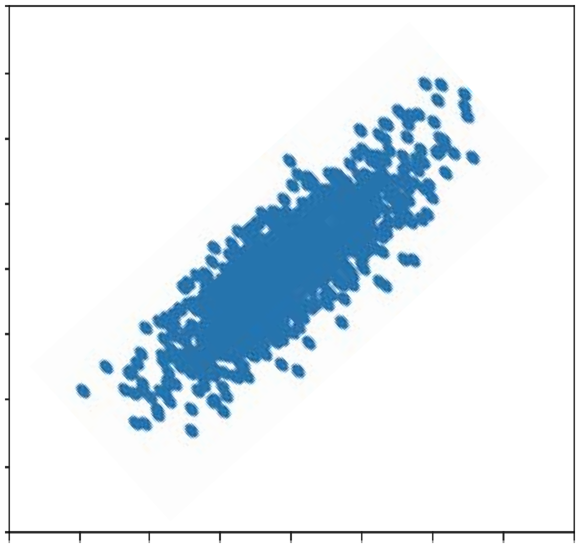
A statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.

In Wikipedia:

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:

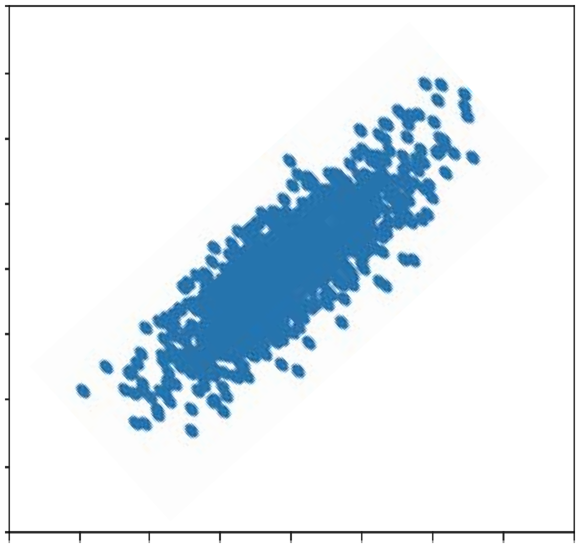


$$\begin{bmatrix} 8 & 1 \\ 1 & 8 \end{bmatrix}$$

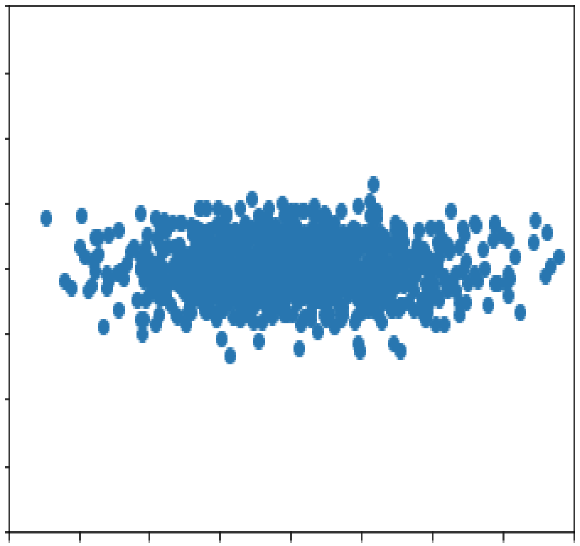
Principal Component Analysis (PCA): definition

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:



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

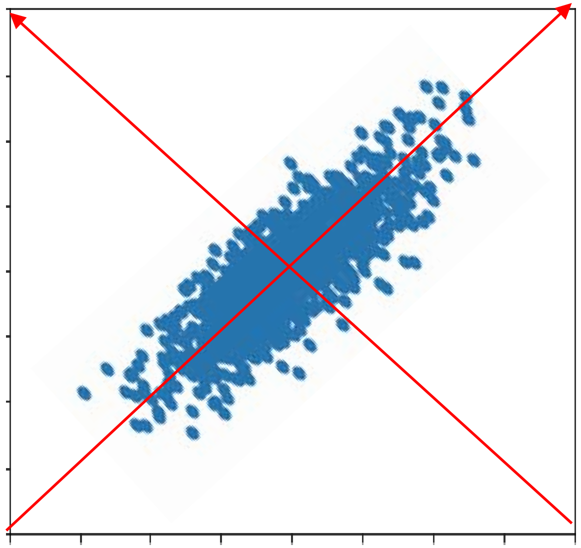


$$\begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}$$

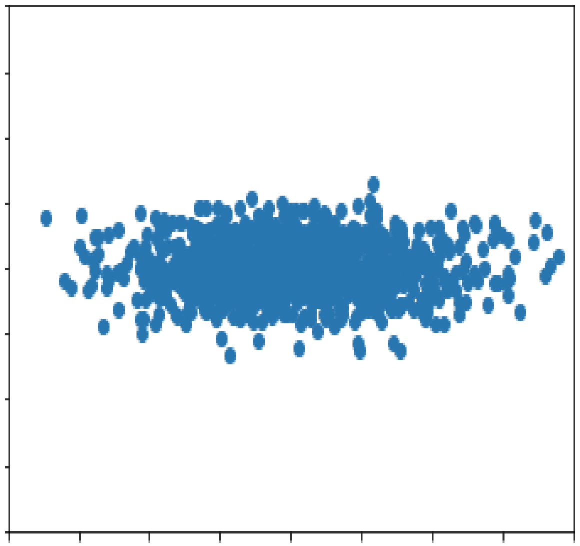
Principal Component Analysis (PCA): definition

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:



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

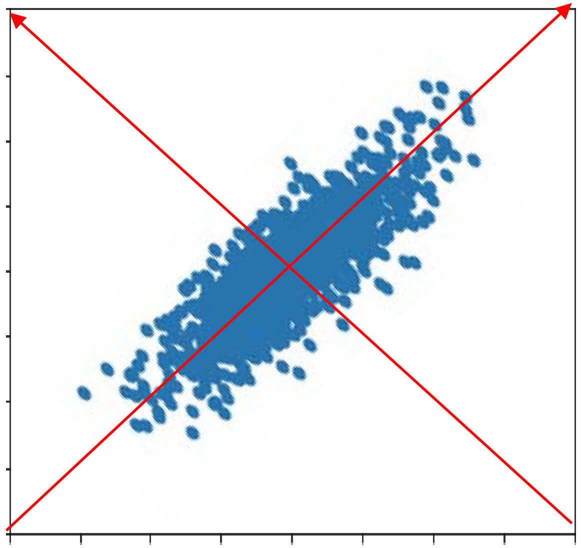


$$\begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}$$

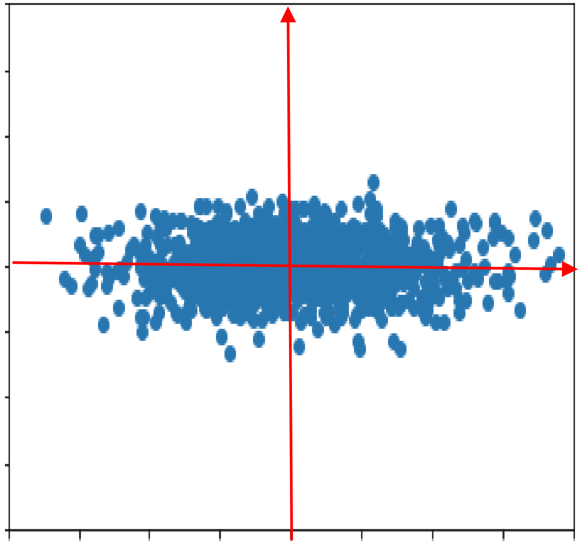
Principal Component Analysis (PCA): definition

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:

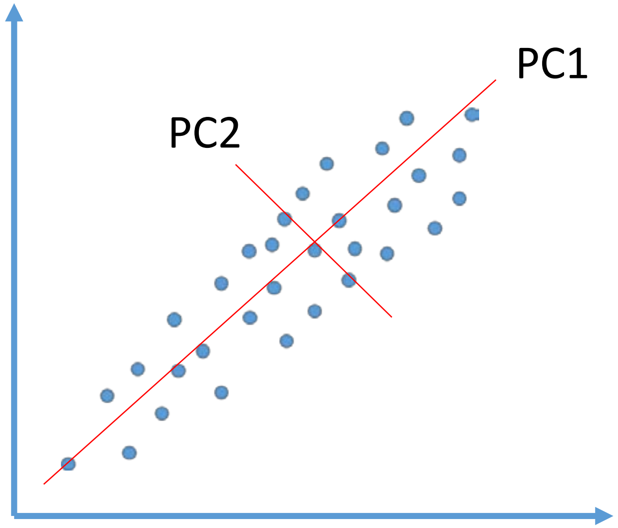


$$\begin{bmatrix} 8 & 1 \\ 1 & 8 \end{bmatrix}$$

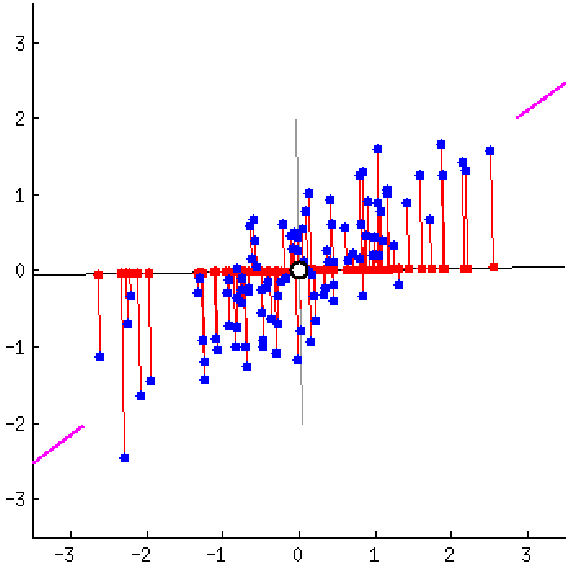


$$\begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}$$

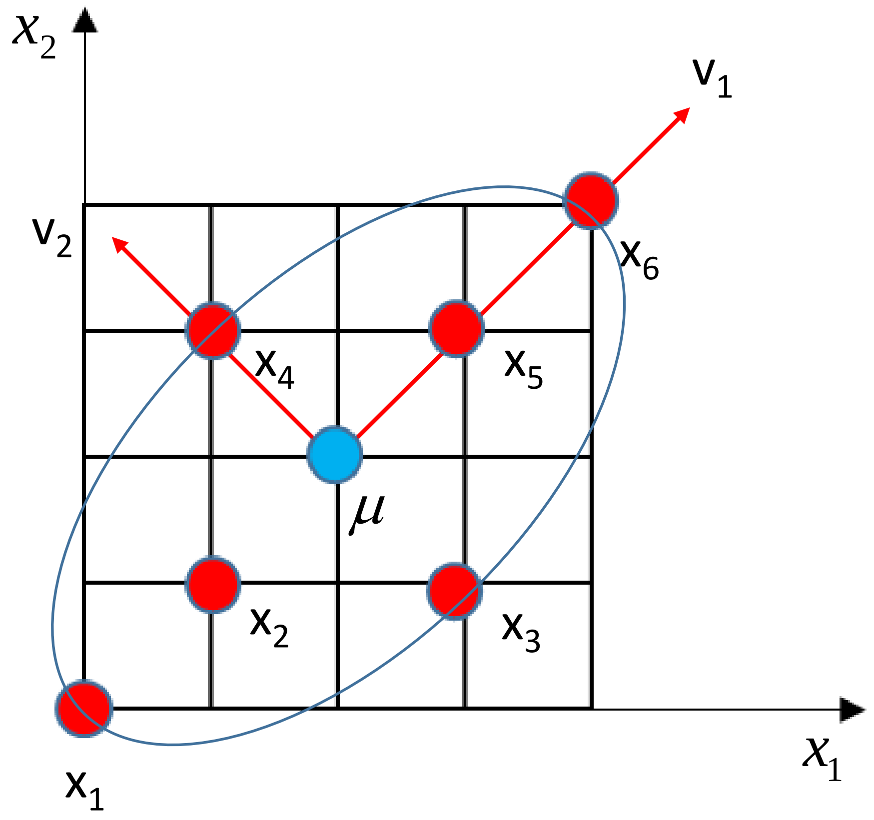
Principal Component Analysis (PCA): intuition



- 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?



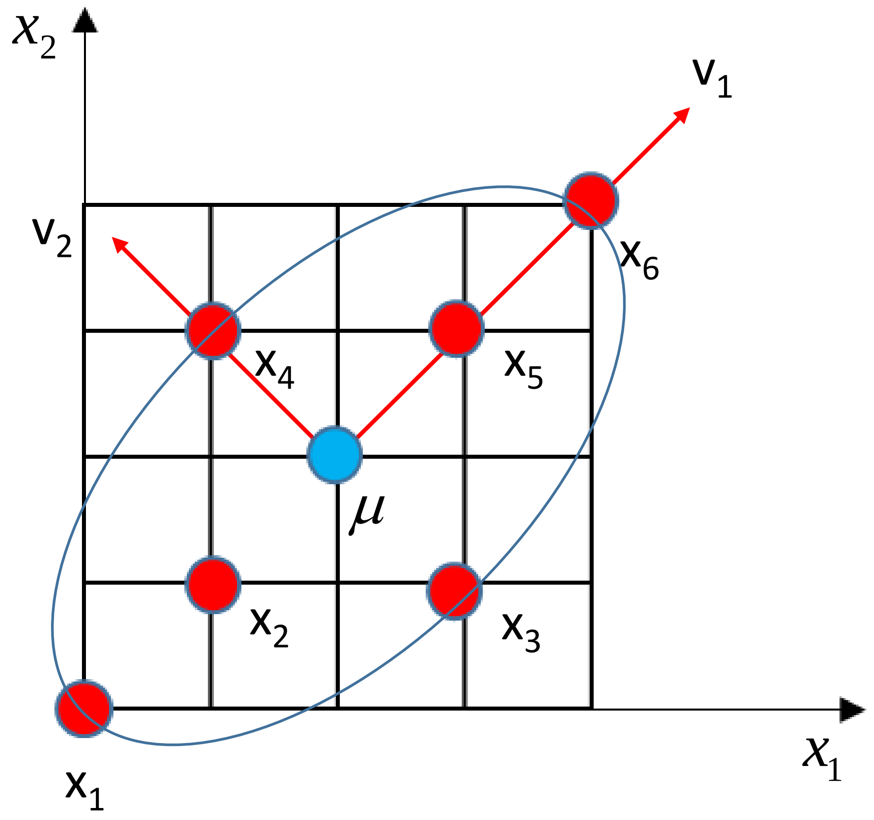
$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} -2 & -2 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}$$

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

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

$$\text{var} = \frac{\sum (x - \mu)^2}{N-1}$$

How to find the principal components showing the largest variance?



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

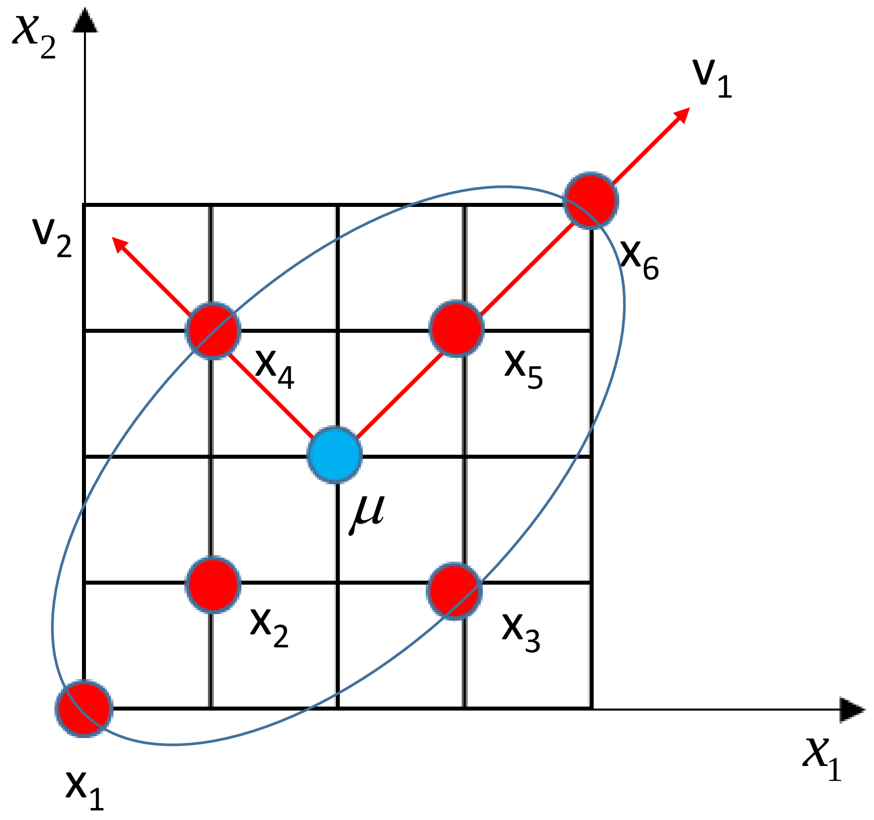
- Distance to data points from the mean along the axis of "v_1"

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

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

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

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"

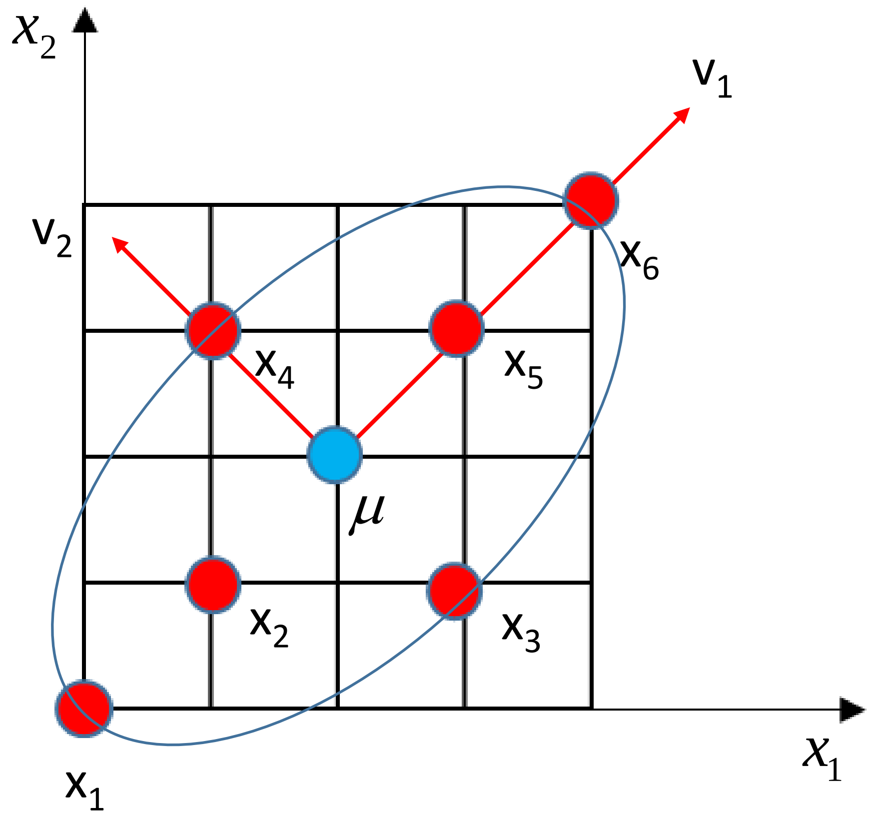
$$= [-2\sqrt{2}, -\sqrt{2}, 0, 0, \sqrt{2}, 2\sqrt{2}] \qquad \text{variance} = 4$$

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

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

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

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"

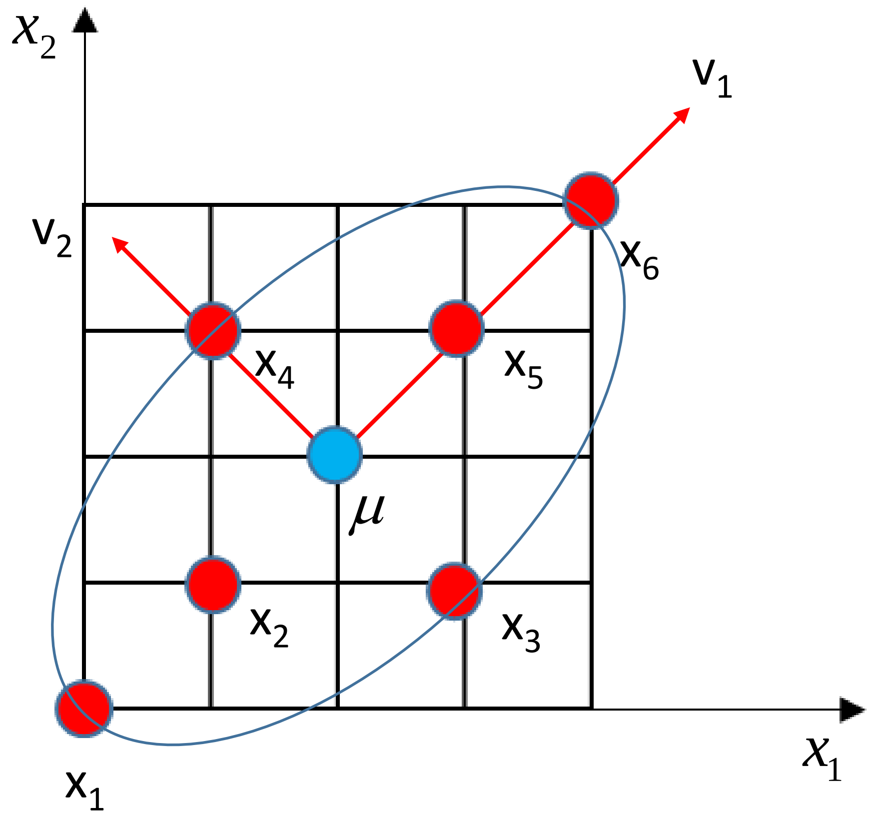
$$= [0, 0, -\sqrt{2}, \sqrt{2}, 0, 0] \qquad \text{variance} = 0.8$$

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

variance along the axis of "x_1"

variance along the axis of "x_2"

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"

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

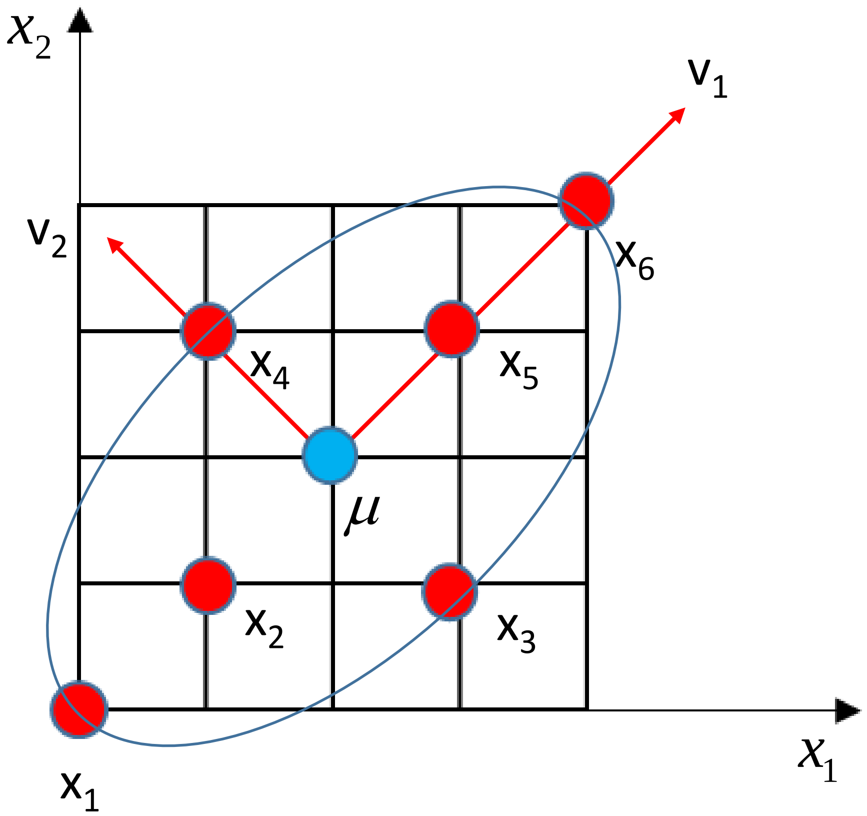
- $$\text{cov}(X) = \begin{bmatrix} 2.4 & 1.6 \\ 1.6 & 2.4 \end{bmatrix}$$

variance along the axis of "x_1"

variance along the axis of "x_2"

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

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"

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

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

variance along the axis of "x_1"

variance along the axis of "x_2"

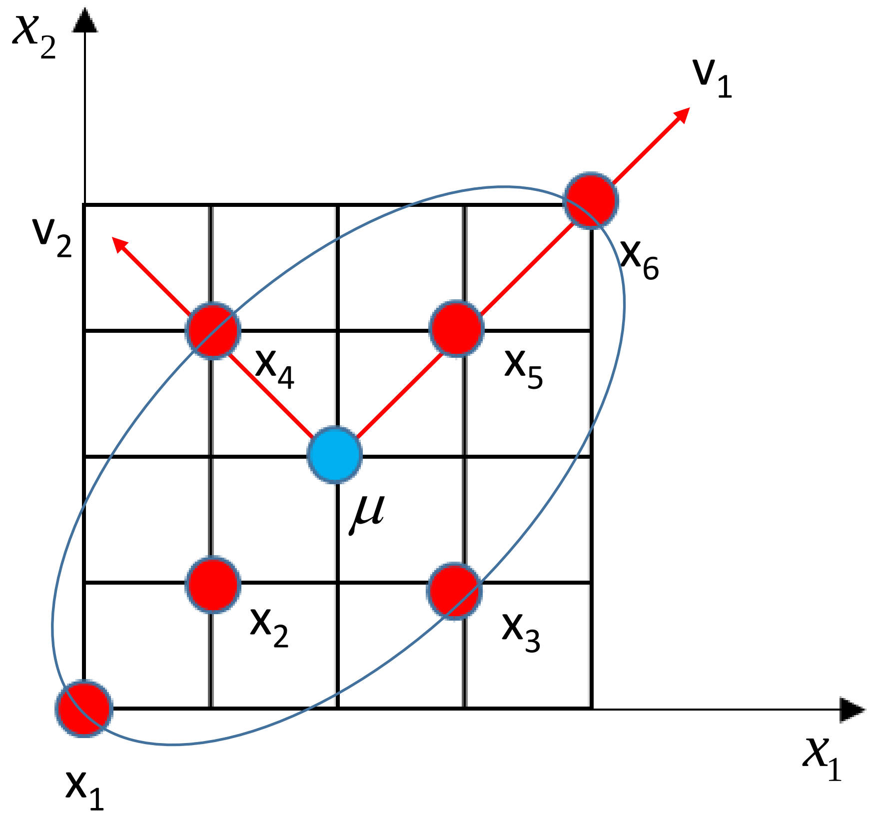
- $$\text{cov}(x) = V \Lambda V^T$$

```
>> [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

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"

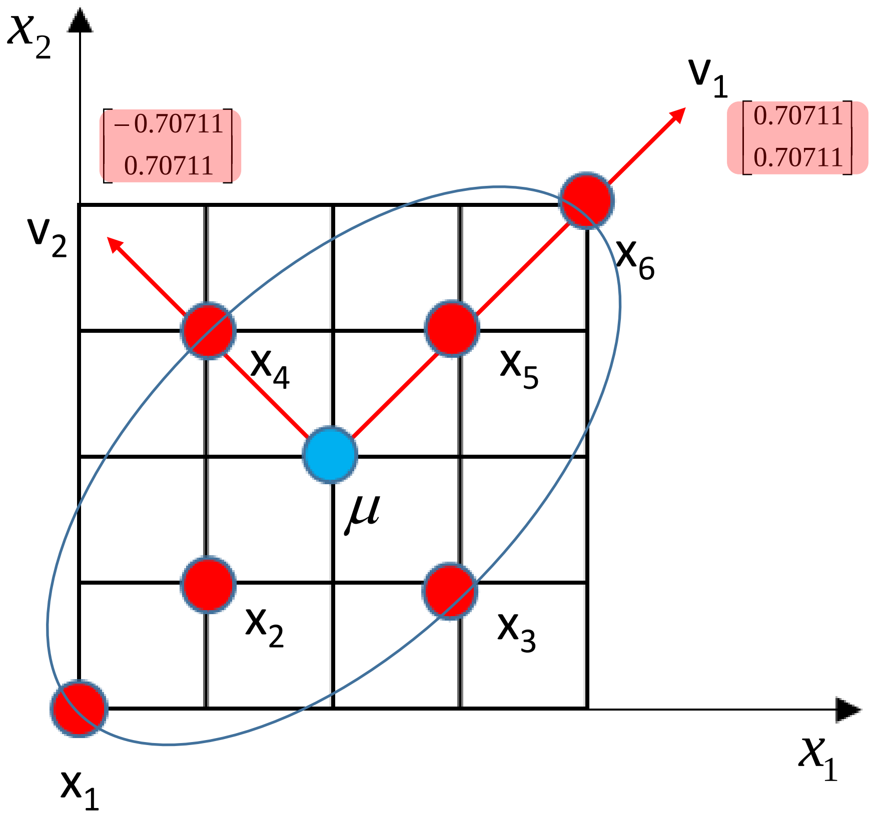
- $\mathrm{cov}(x) = V \Lambda V^T$

```
>> [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

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"

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

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

variance along the axis of "x_1"

variance along the axis of "x_2"

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

```
>> [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

How to find the principal components showing the largest variance?

1) Find the covariance matrix of data points.

```
>> 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
```

How to find the principal components showing the largest variance?

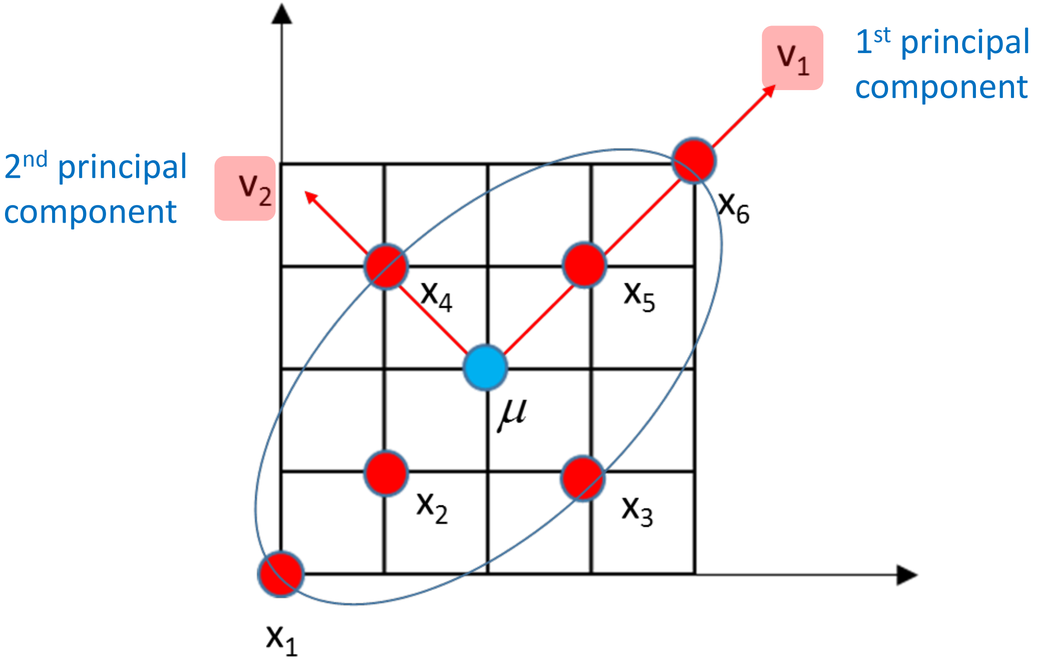
1) Find the covariance matrix of data points.
2) Obtain the eigen values and vectors of the covariance matrix: eigen decomposition.

```
>> 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
```

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
```

2nd principal component

1st principal component

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

  -0.70711   0.70711
   0.70711   0.70711

val =

Diagonal Matrix

   0.80000         0
         0   4.00000
```

How to find the principal components showing the largest variance?

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

Eigen Value 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
```

How to find the principal components showing the largest variance?

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

Eigen Value 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
```

?

How to find the principal components showing the largest variance?

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

Eigen Value 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
```

$$X^T X = (U \Sigma V^T)^T (U \Sigma V^T)$$

How to find the principal components showing the largest variance?

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

Eigen Value 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)

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

$$X^T X = (U\Sigma V^T)^T (U\Sigma V^T)$$

$$= V\Sigma^T U^T U\Sigma V^T$$

$$= V\Sigma^2 V^T$$

How to find the principal components showing the largest variance?

- Actually, there is a more convenient way of doing it.
- 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)

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

$$X^T X = (U \Sigma V^T)^T (U \Sigma V^T)$$

$$= V \Sigma^T U^T U \Sigma V^T$$

$$= V \Sigma^2 V^T \qquad \Lambda = \Sigma^2$$

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

Eigen value   Singular value

How to find the principal components showing the largest variance?

- Actually, there is a more convenient way of doing it.
- 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
```

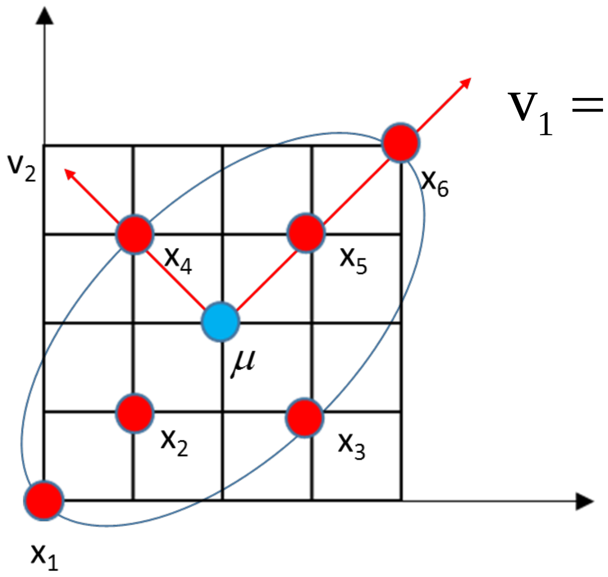
```
>> [vec, val]=eig(transpose(x)*x)
vec =

  -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
```

Now we know how to find the principal components

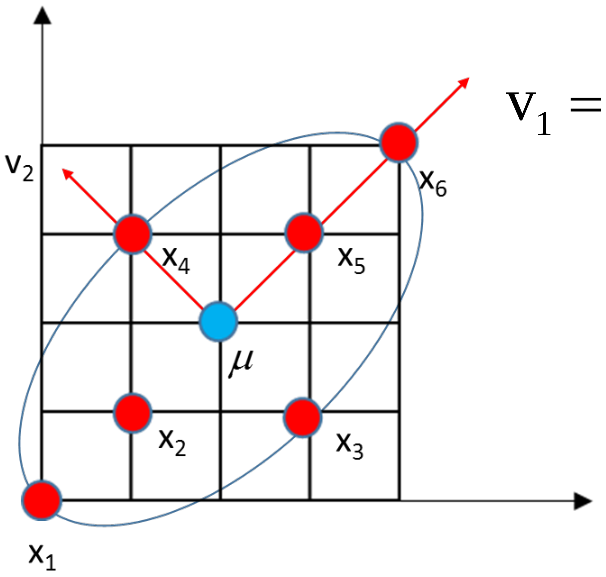


$$v_1 = \begin{bmatrix} 0.70711 \\ 0.70711 \end{bmatrix}$$

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

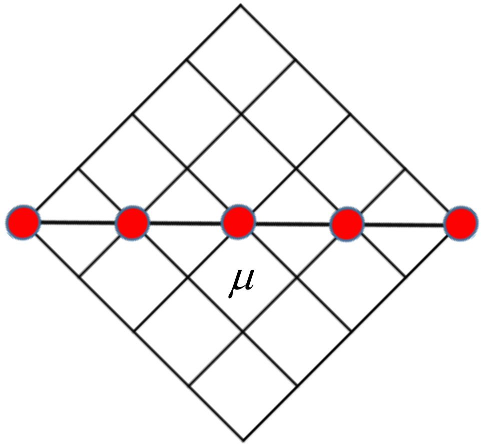
2 dimension data points can be represented into one dimension space (v_1)

Dimension reduction



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

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



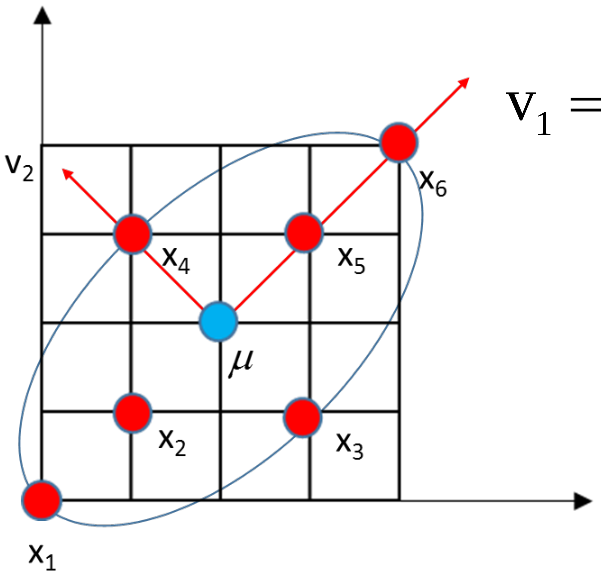
$$v_1 = \begin{bmatrix} 0.70711 \\ 0.70711 \end{bmatrix}$$

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

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

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

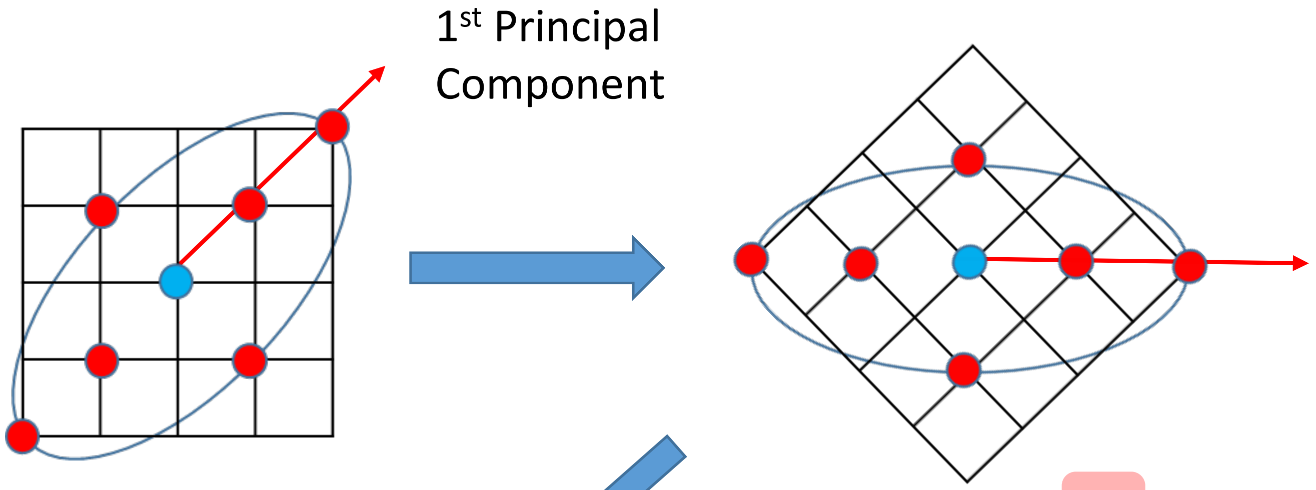
Dimension reduction



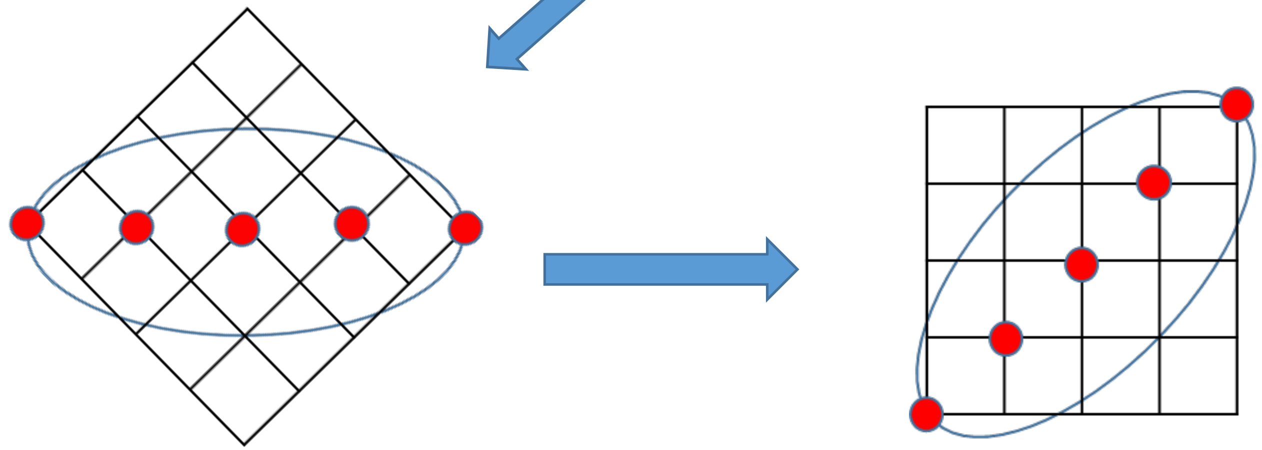
$$v_1 = \begin{bmatrix} 0.70711 \\ 0.70711 \end{bmatrix}$$

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

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



$$X\_rot = X \cdot \boxed{V}$$

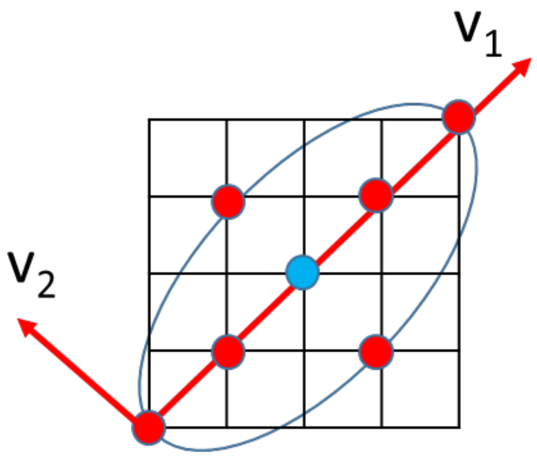


Set the "v_2" into zero

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

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

Example

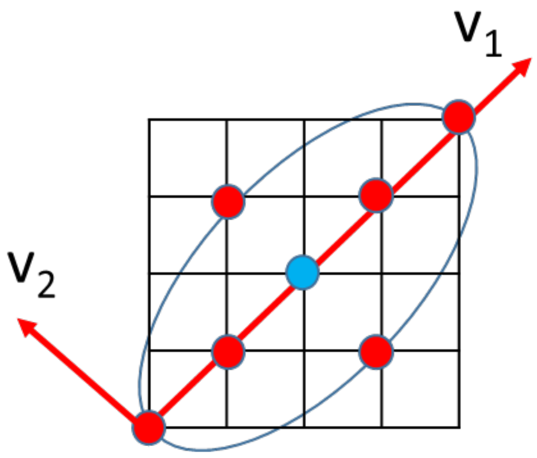


$$X = \begin{bmatrix} -2 & -2 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}$$

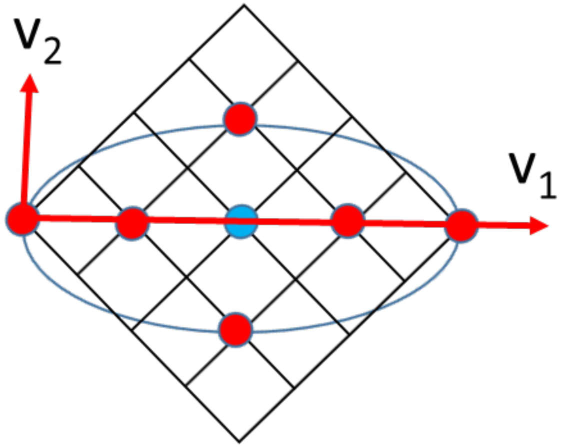
$$X = U\Sigma V^T$$

```
>> [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
```

Example



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

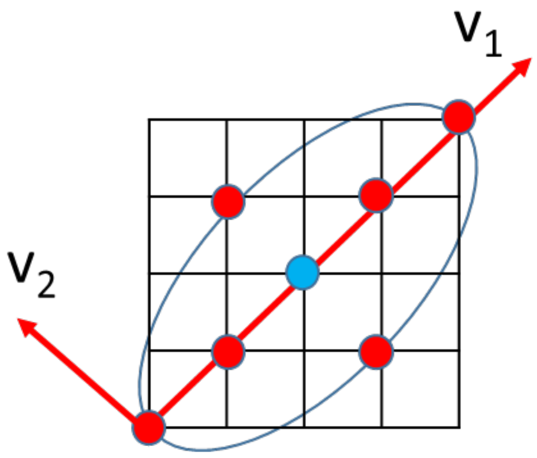


$$\begin{array}{ccc} \text{X\_rot} & & \text{X} \quad\quad\quad \text{V} \\ \begin{bmatrix} -2\sqrt{2} & 0 \\ -\sqrt{2} & 0 \\ 0 & -\sqrt{2} \\ 0 & \sqrt{2} \\ \sqrt{2} & 0 \\ 2\sqrt{2} & 0 \end{bmatrix} & = & \begin{bmatrix} -2 & -2 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix} \begin{bmatrix}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

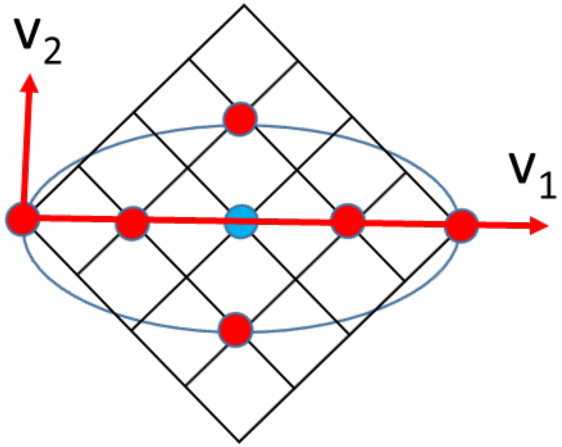
$$X = U \Sigma V^T$$

```
>> [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
```

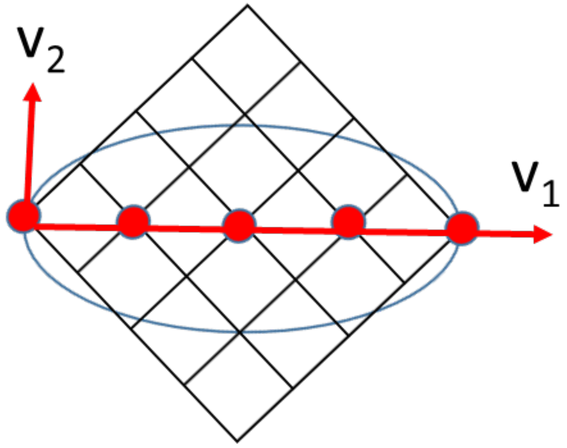
Example



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



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

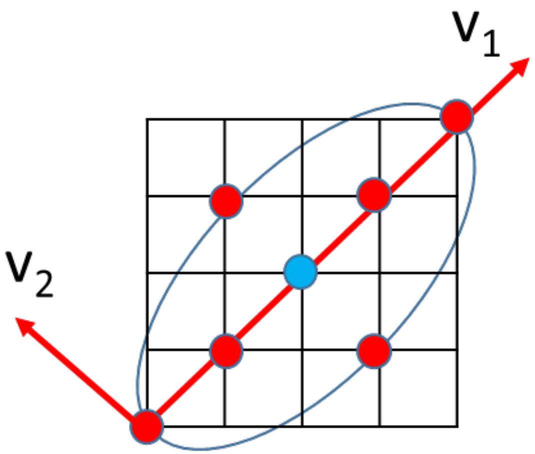


Set the "v_2" elements into zero

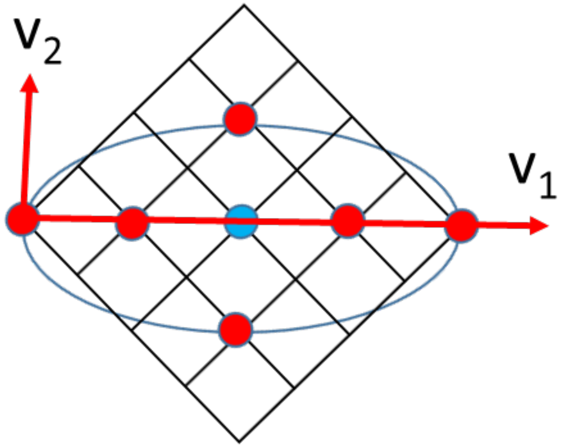
X_rot_zero

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

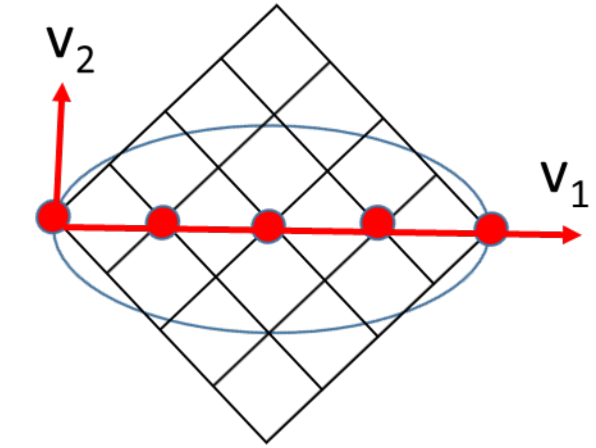
Example



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



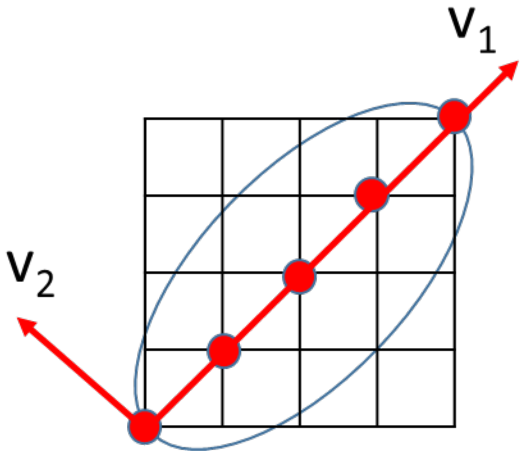
$$\begin{array}{ccc} \text{X\_rot} & & \text{X} \quad\quad \text{V} \\ \begin{bmatrix} -2\sqrt{2} & 0 \\ -\sqrt{2} & 0 \\ 0 & -\sqrt{2} \\ 0 & \sqrt{2} \\ \sqrt{2} & 0 \\ 2\sqrt{2} & 0 \end{bmatrix} & = & \begin{bmatrix} -2 & -2 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix} \begin{bmatrix}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



Set the “v_2” elements into zero

X_rot_zero

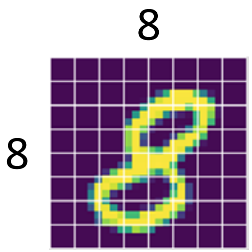
$$\begin{bmatrix} -2\sqrt{2} & 0 \\ -\sqrt{2} & 0 \\ 0 & 0 \\ 0 & 0 \\ \sqrt{2} & 0 \\ 2\sqrt{2} & 0 \end{bmatrix}$$

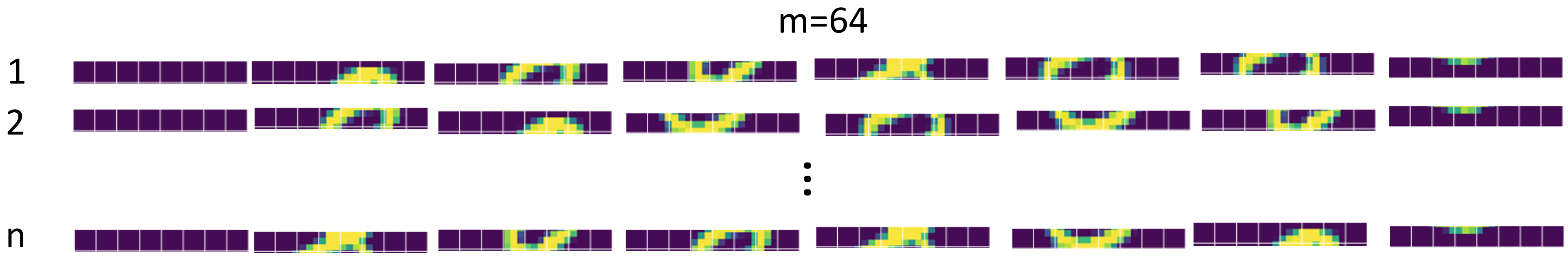


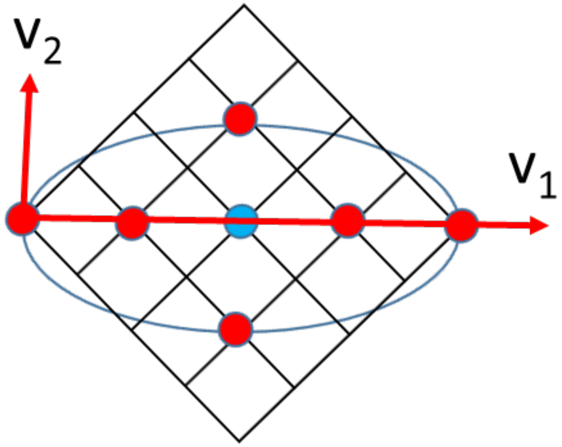
$$\begin{bmatrix} -2 & -2 \\ -1 & -1 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}$$

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

How to use PCA for machine learning?







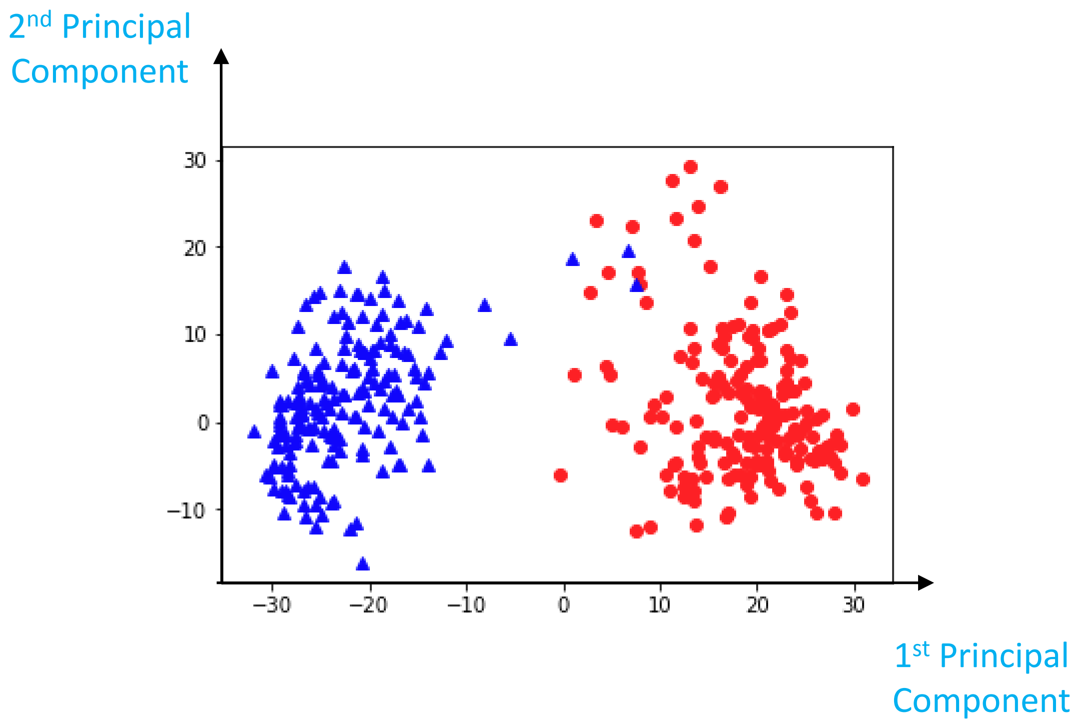
$$X_{\text{rot}} = X \cdot V$$

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

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

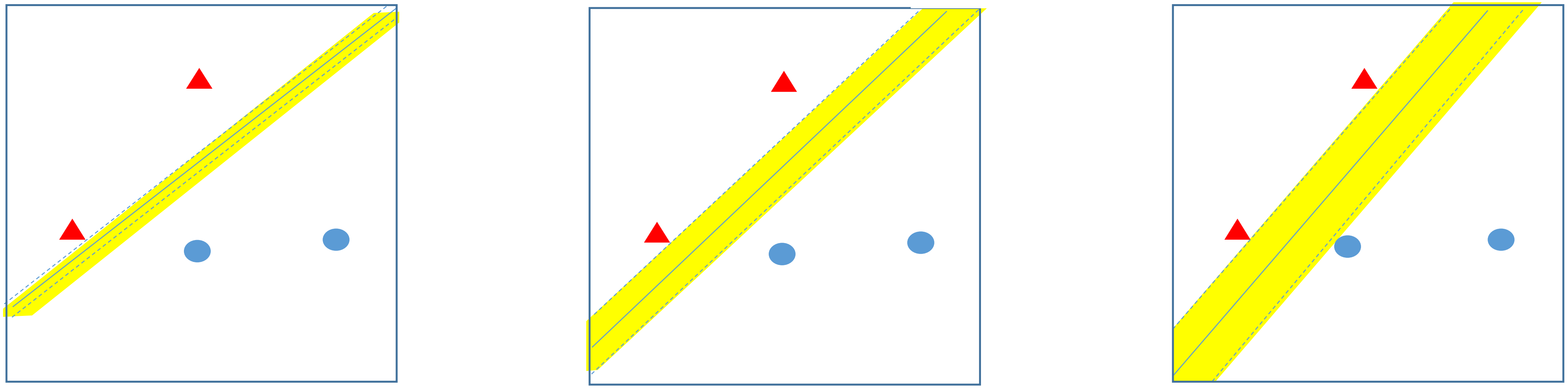
Demo

- 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.

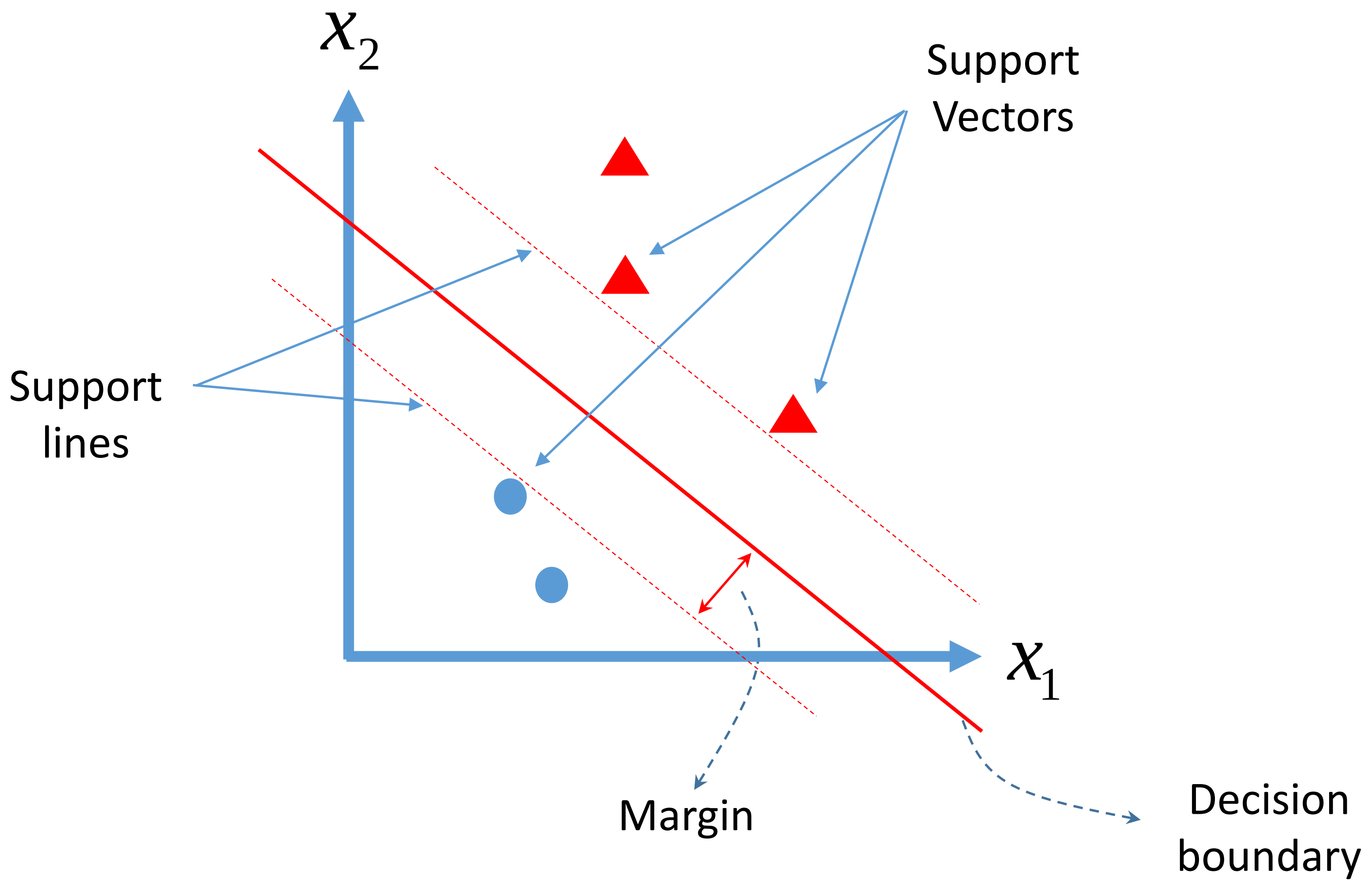


Support Vector Machine (SVM)

Which one is better for classification?



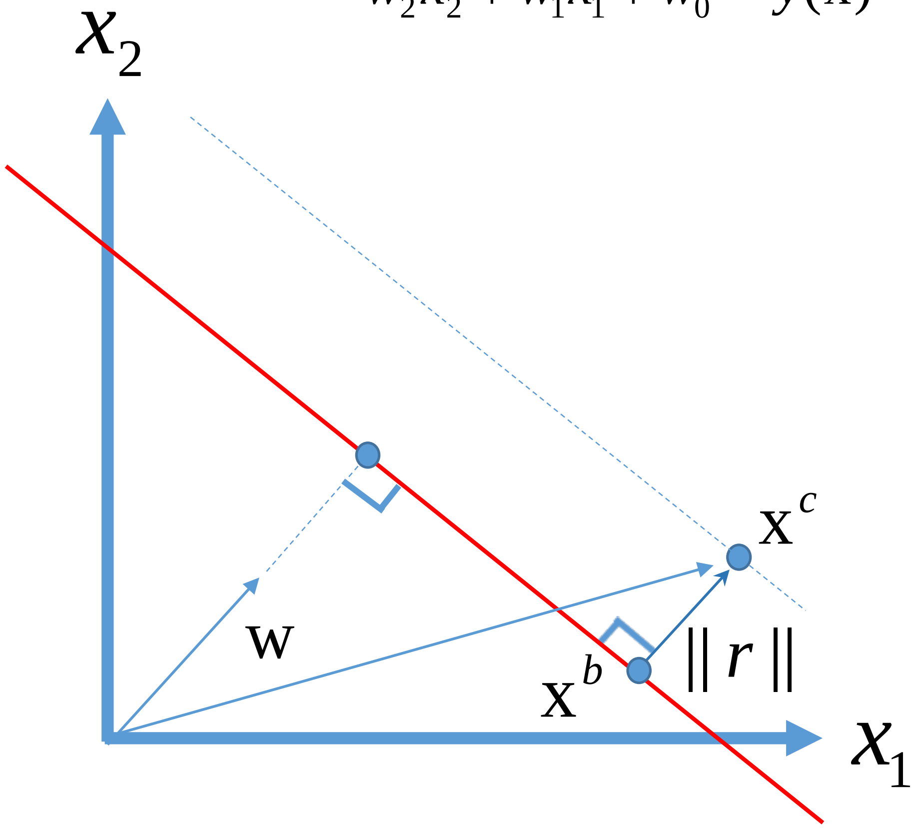
Terminology used in this lecture



Margin distance

$$\mathrm{x}^{c} = \mathrm{x}^{b} + \| r \| \frac{\mathrm{w}}{\| \mathrm{w} \|}$$

$$w_2 x_2 + w_1 x_1 + w_0 = y(\mathrm{x})$$



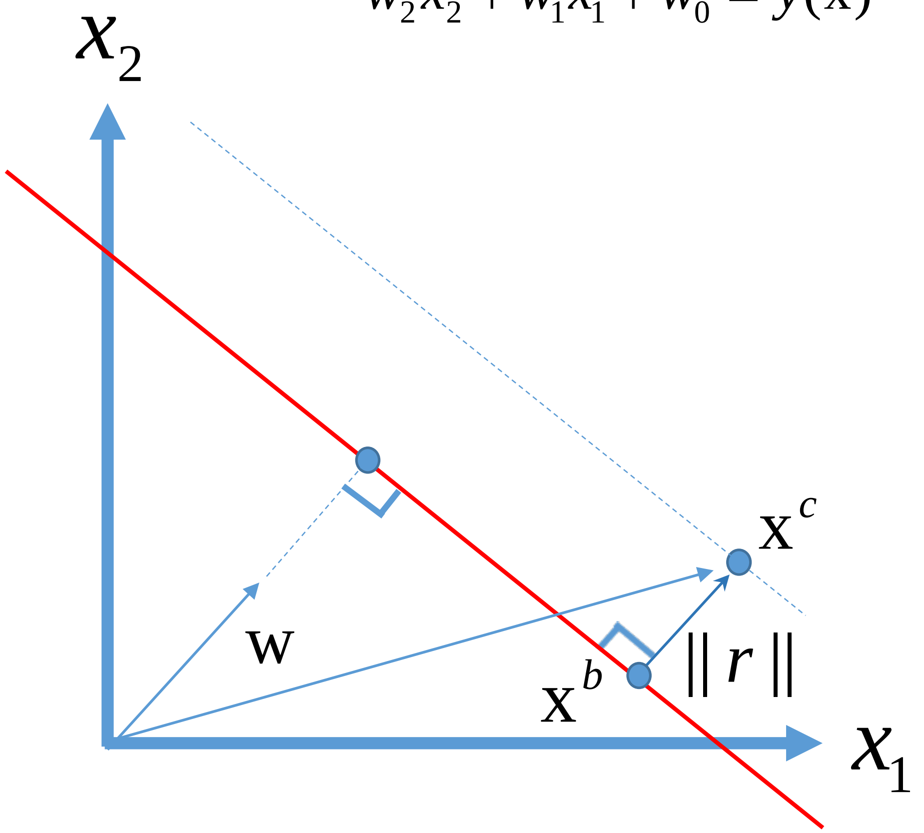
Margin distance

Size of the vector (x^b->x^c)

$$\mathrm{x}^c = \mathrm{x}^b + \boxed{\| r \|}\boxed{\frac{\mathrm{w}}{\| \mathrm{w} \|}}$$

Unit vector showing the direction only

$$w_2 x_2 + w_1 x_1 + w_0 = y(\mathrm{x})$$



Margin distance

Size of the vector (x^b->x^c)

$$\mathrm{x}^c = \mathrm{x}^b + \|r\| \frac{\mathrm{w}}{\|\mathrm{w}\|}$$

Unit vector showing the direction only

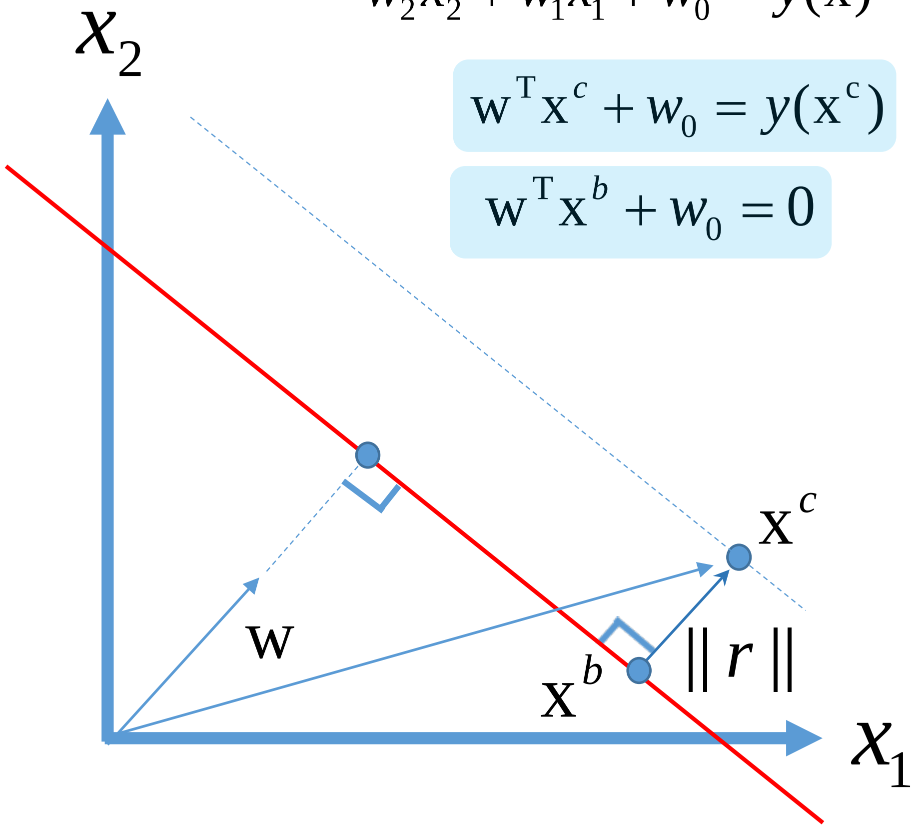
- Let's multiply w^T and add w_0 in both sides.

$$\mathrm{w}^T\mathrm{x}^c + w_0 = \mathrm{w}^T\mathrm{x}^b + w_0 + \mathrm{w}^T \|r\| \frac{\mathrm{w}}{\|\mathrm{w}\|}$$

$$w_2 x_2 + w_1 x_1 + w_0 = y(\mathrm{x})$$

$$\mathrm{w}^T\mathrm{x}^c + w_0 = y(\mathrm{x}^c)$$

$$\mathrm{w}^T\mathrm{x}^b + w_0 = 0$$



Margin distance

Size of the vector (x^b->x^c)

$$\mathrm{x}^c = \mathrm{x}^b + \|r\| \frac{\mathrm{w}}{\|\mathrm{w}\|}$$

Unit vector showing the direction only

- Let's multiply w^T and add w_0 in both sides.

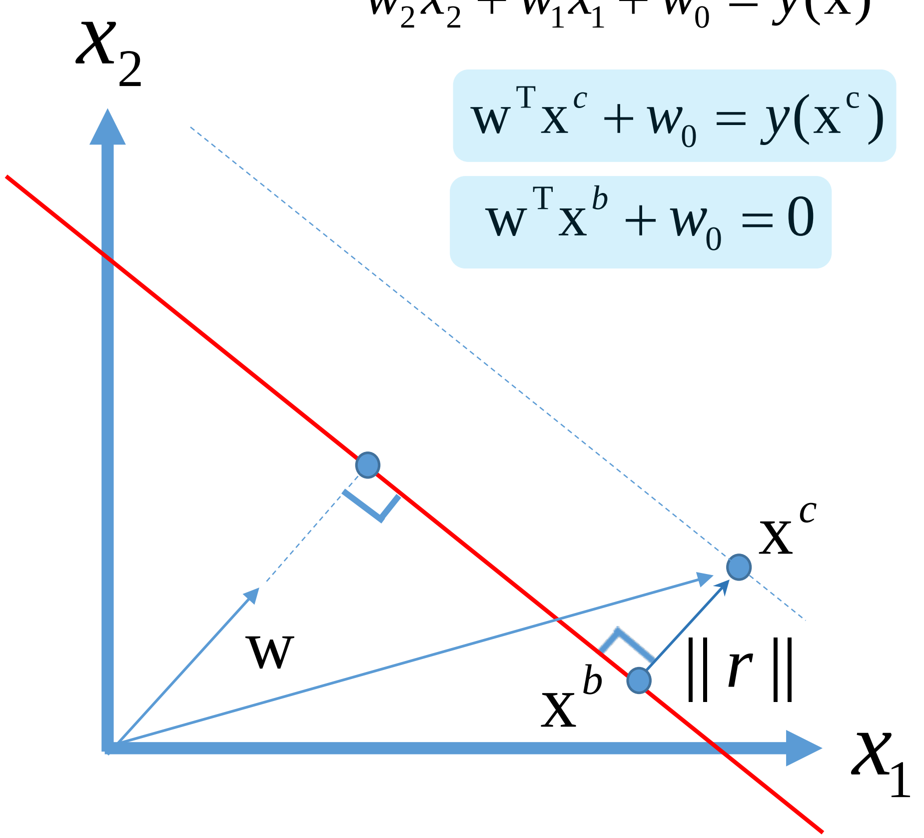
$$\mathrm{w}^T\mathrm{x}^c + w_0 = \mathrm{w}^T\mathrm{x}^b + w_0 + \mathrm{w}^T \|r\| \frac{\mathrm{w}}{\|\mathrm{w}\|}$$

$$\mathrm{y}(\mathrm{x}^c) = \mathrm{w}^T \|r\| \frac{\mathrm{w}}{\|\mathrm{w}\|}$$

$$w_2x_2 + w_1x_1 + w_0 = y(\mathrm{x})$$

$$\mathrm{w}^T\mathrm{x}^c + w_0 = y(\mathrm{x}^c)$$

$$\mathrm{w}^T\mathrm{x}^b + w_0 = 0$$



Margin distance

Size of the vector (x^b->x^c)

$$\mathrm{x}^c = \mathrm{x}^b + \boxed{\| r \|}\, \boxed{\frac{\mathrm{w}}{\| \mathrm{w} \|}}$$

Unit vector showing the direction only

- ❑ Let's multiply w^T and add w_0 in both sides.

$$\mathrm{w}^{\mathrm{T}}\mathrm{x}^c + w_0 = \mathrm{w}^{\mathrm{T}}\mathrm{x}^b + w_0 + \mathrm{w}^{\mathrm{T}} \| r \| \frac{\mathrm{w}}{\| \mathrm{w} \|}$$

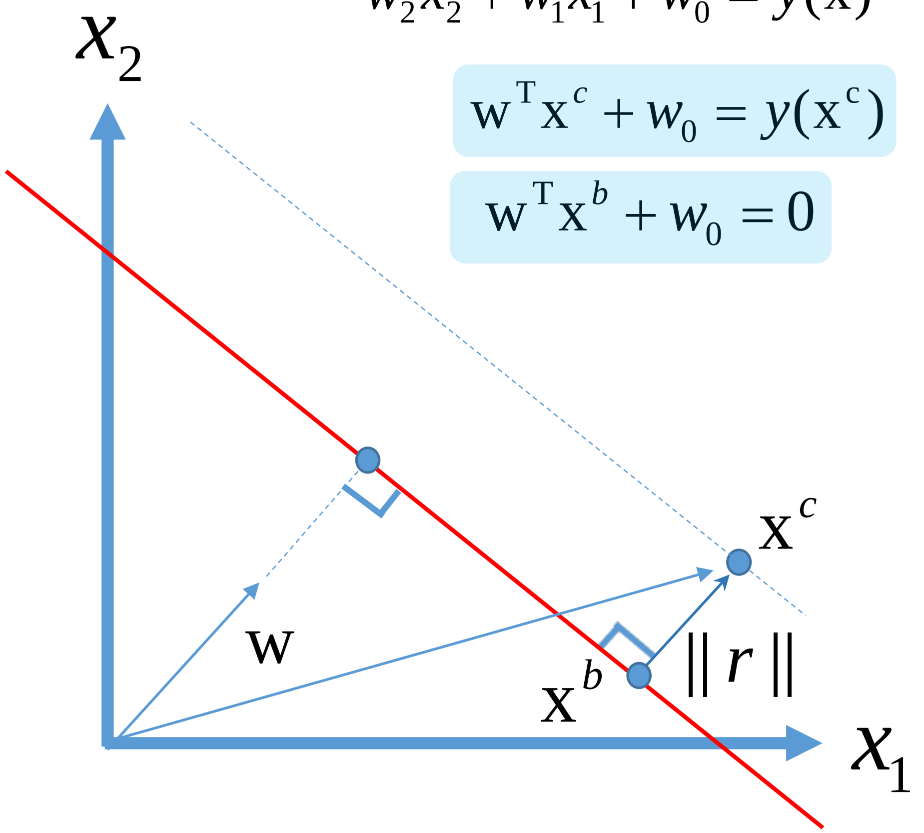
$$\mathrm{y}(\mathrm{x}^{\mathrm{c}}) = \mathrm{w}^{\mathrm{T}} \| r \| \frac{\mathrm{w}}{\| \mathrm{w} \|}$$

$$\| r \| = \frac{\mathrm{y}(\mathrm{x}^{\mathrm{c}})}{\| \mathrm{w} \|}$$

$$w_2 x_2 + w_1 x_1 + w_0 = y(\mathrm{x})$$

$$\mathrm{w}^{\mathrm{T}}\mathrm{x}^c + w_0 = y(\mathrm{x}^{\mathrm{c}})$$

$$\mathrm{w}^{\mathrm{T}}\mathrm{x}^b + w_0 = 0$$



Margin distance

Size of the vector (x^b->x^c)

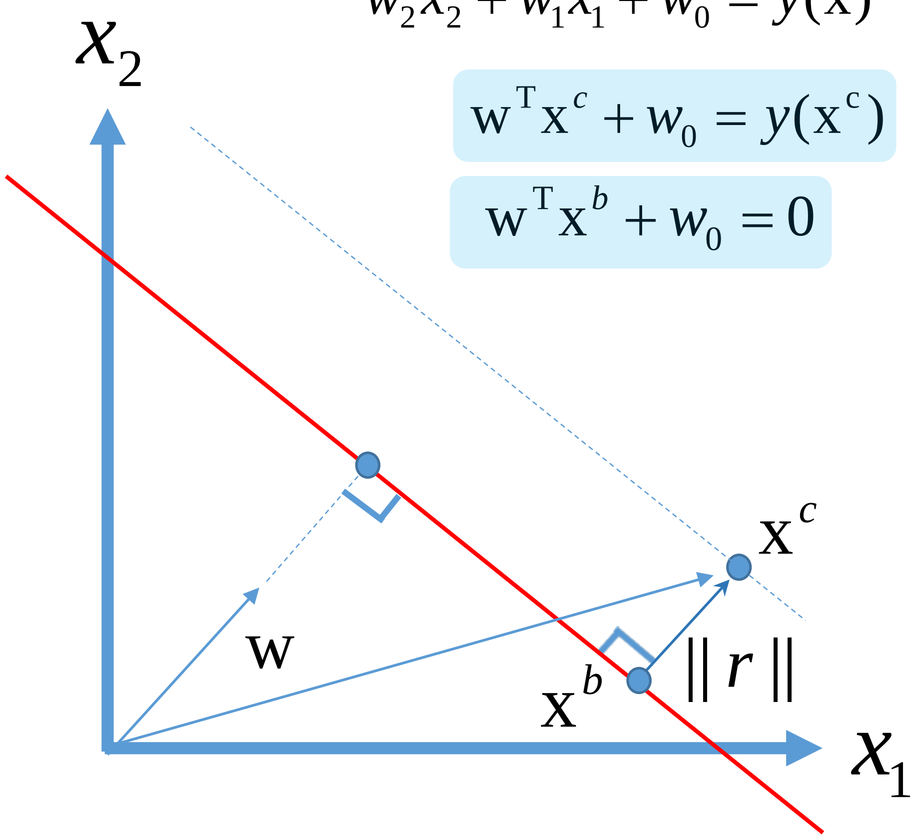
$$\mathrm{x}^c = \mathrm{x}^b + \|r\| \frac{\mathrm{w}}{\|\mathrm{w}\|}$$

Unit vector showing the direction only

$w_2 x_2 + w_1 x_1 + w_0 = y(\mathrm{x})$

$\mathrm{w}^{\mathrm{T}}\mathrm{x}^c + w_0 = y(\mathrm{x}^c)$

$\mathrm{w}^{\mathrm{T}}\mathrm{x}^b + w_0 = 0$



- Let's multiply w^{T} and add w_0 in both sides.

$$\mathrm{w}^{\mathrm{T}}\mathrm{x}^c + w_0 = \mathrm{w}^{\mathrm{T}}\mathrm{x}^b + w_0 + \mathrm{w}^{\mathrm{T}} \|r\| \frac{\mathrm{w}}{\|\mathrm{w}\|}$$

$$y(\mathrm{x}^c) = \mathrm{w}^{\mathrm{T}} \|r\| \frac{\mathrm{w}}{\|\mathrm{w}\|}$$

$$\|r\| = \frac{y(\mathrm{x}^c)}{\|\mathrm{w}\|}$$

$$\|r\| = \frac{1}{\|\mathrm{w}\|}$$

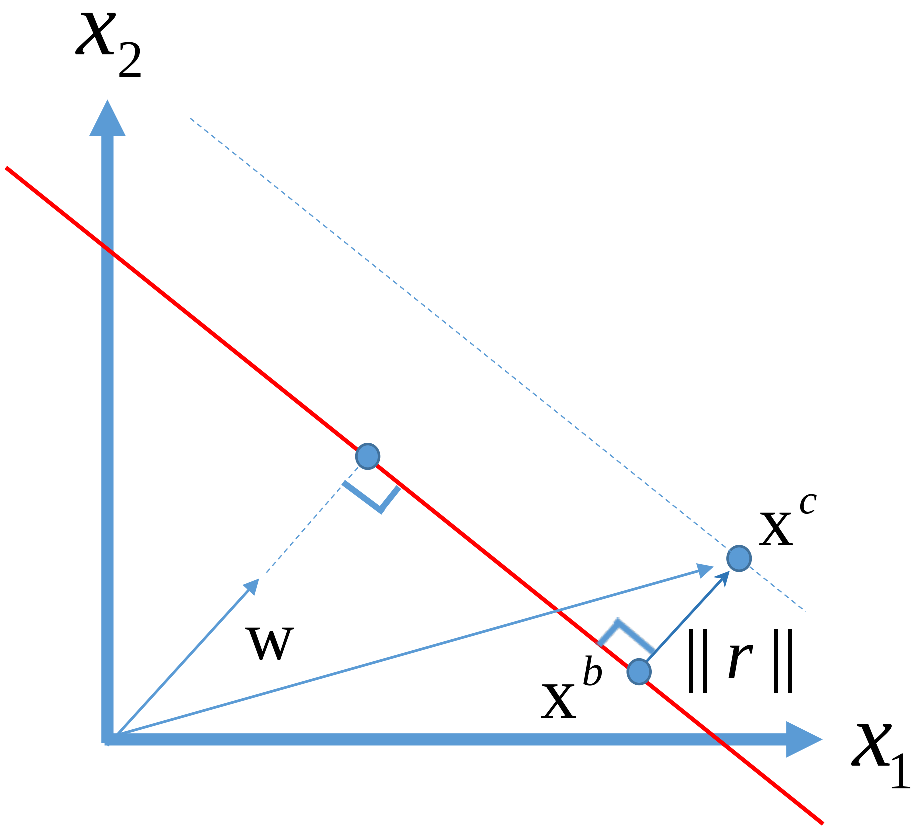
Let's say

$$|y(\mathrm{x}^c)| = 1$$

Problem formulation

- Finding a decision boundary which maximizes the margin.

$$\max \| r \| = \frac{1}{\| \mathrm{w} \|}$$



Problem formulation

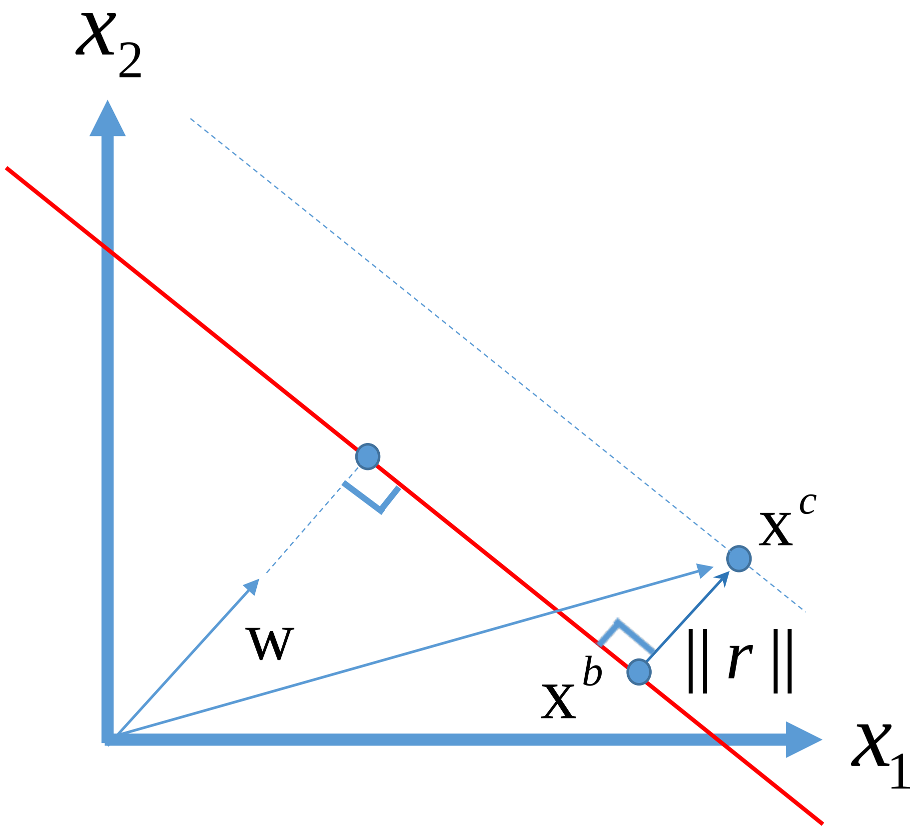
- Finding a decision boundary which maximizes the margin.

$$\max \|r\| = \frac{1}{\|\mathrm{w}\|}$$

s.t.

$$t_n y(\mathrm{x}_n) > 0$$ ➡ Every data points are classified correctly.

$$\begin{cases} t_n = +1, & y(\mathrm{x}_n) > 0 \\ t_n = -1, & y(\mathrm{x}_n) < 0 \end{cases}$$



$$w_2 x_2 + w_1 x_1 + w_0 = 0$$

Problem formulation

- Let’s make it a quadratic programming problem.

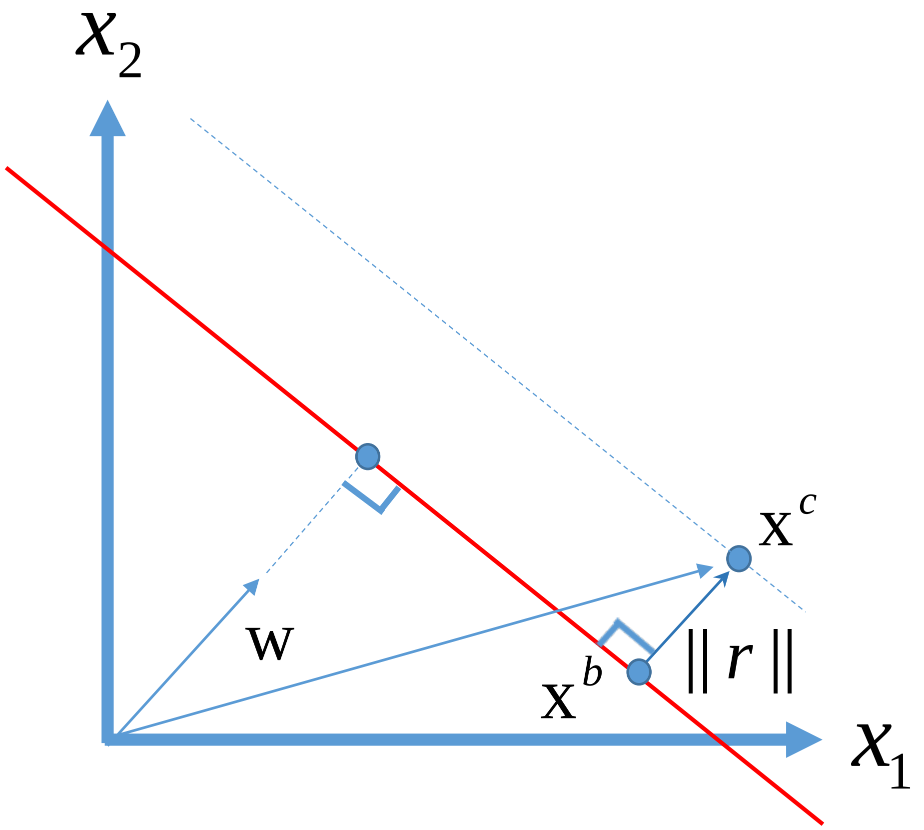
$$\max \frac{1}{\|\mathrm{w}\|}$$

$$s.t. \quad t_n y(\mathrm{x}_n) > 0, \quad \forall n$$

- Finally

$$\min \frac{1}{2}\|\mathrm{w}\|^2$$

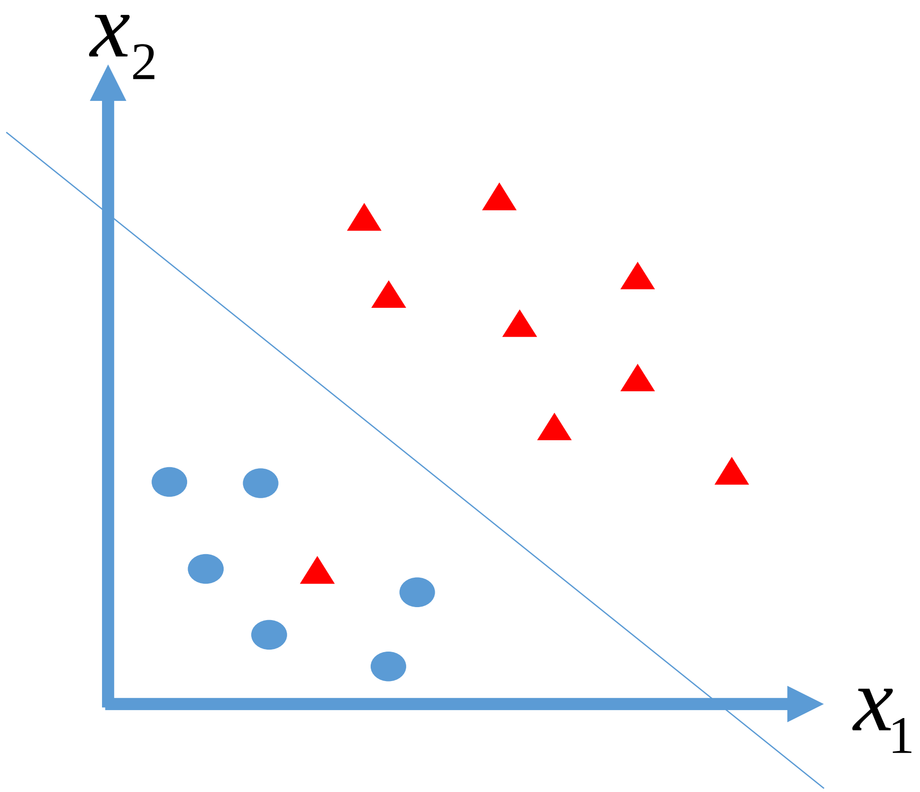
$$s.t. \quad t_n(\mathrm{w}^{\mathrm{T}}\mathrm{x}_n + w_0) \geq 1, \quad \forall n$$



$$w_2 x_2 + w_1 x_1 + w_0 = 0$$

How about non-linearly separable case?

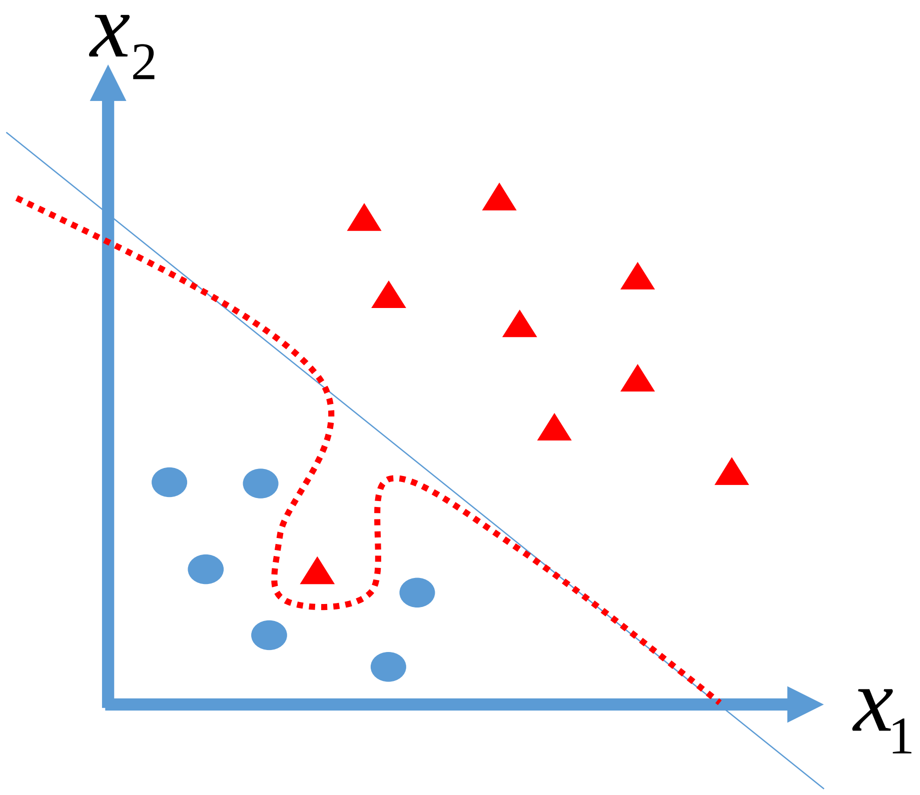
$$\min \frac{1}{2}\|\mathrm{w}\|^2$$
$$s.t.\quad t_n(\mathrm{w}^{\mathrm{T}}\mathrm{x}_n + w_0) \geq 1, \quad \forall n$$

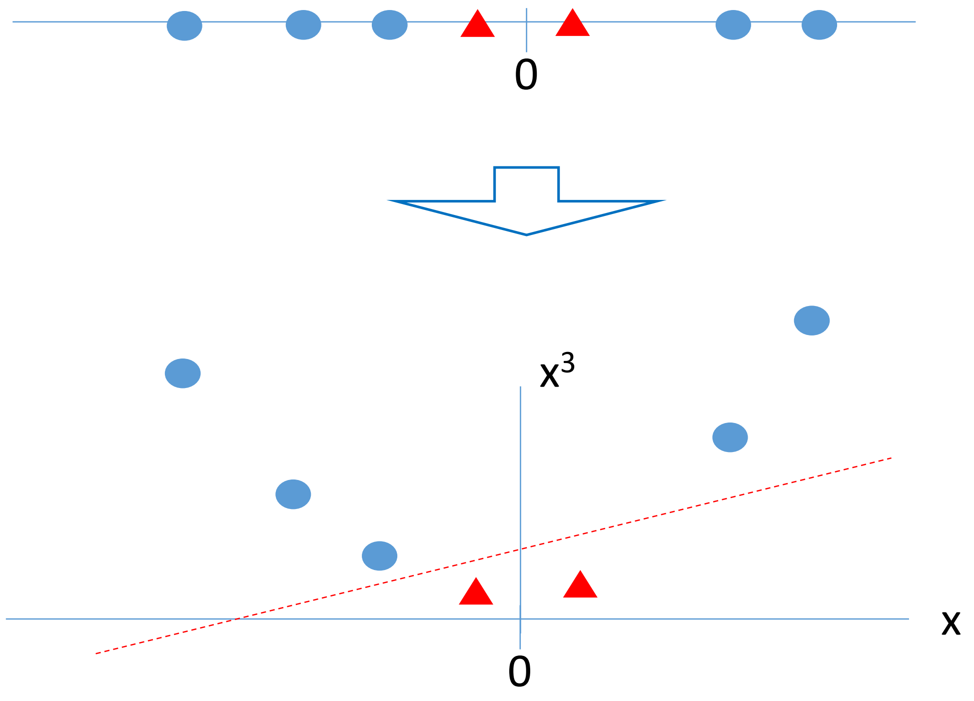


How about non-linearly separable case?

$$\min \frac{1}{2} \| \mathrm{w} \|^2$$
$$s.t. \quad t_n(\mathrm{w}^{\mathrm{T}}\mathrm{x}_n + w_0) \geq 1, \quad \forall n$$

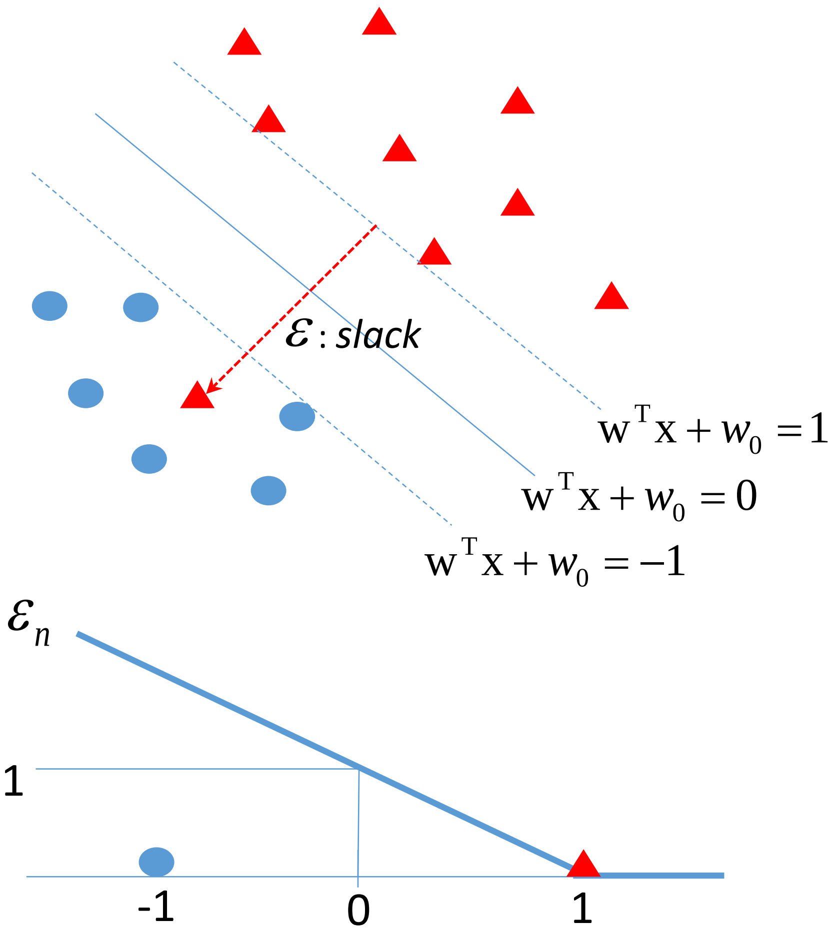
	Approaches
Option 1	Soft margin SVM
Option 2	Kernel trick





Soft margin SVM

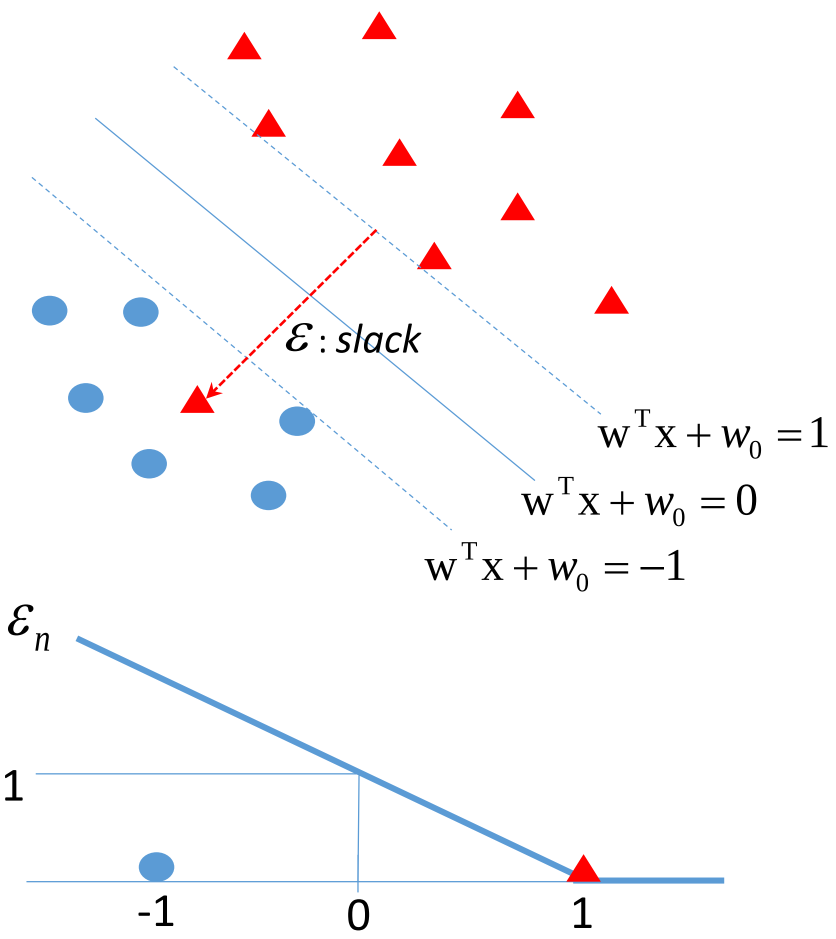
Option 1: soft margin SVM



- Remember the constraint below?

$$t_n(w^T x_n + w_0) \geq 1, \quad \forall n$$

Option 1: soft margin SVM



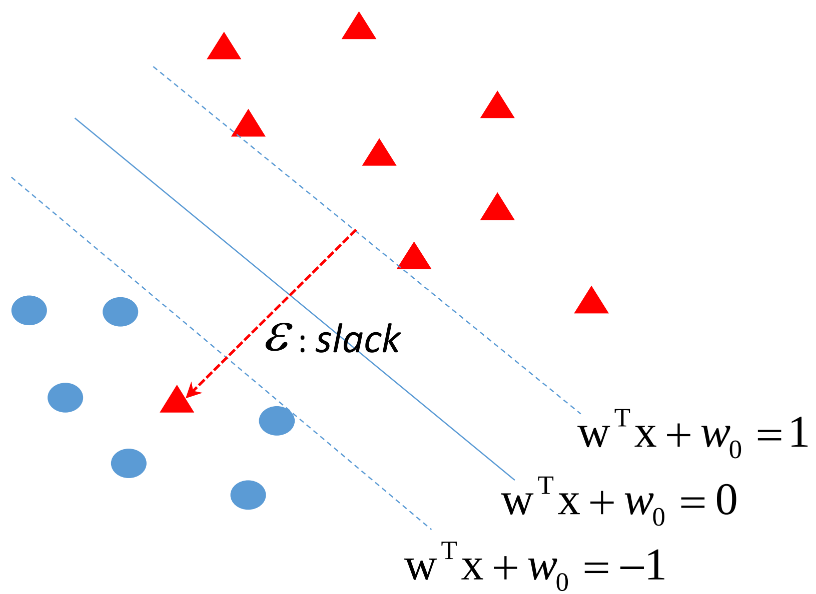
- Remember the constraint below?

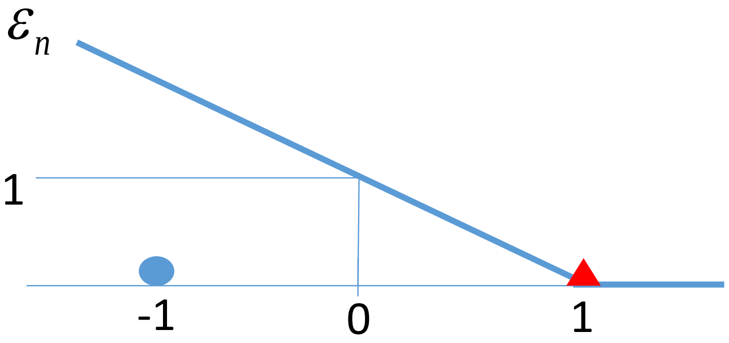
$$t_n(\mathrm{w}^{\mathrm{T}}\mathrm{x}_n + w_0) \geq 1, \quad \forall n$$

- For the data points which are non-separable, we relax the constraint:

$$t_n(\mathrm{w}^{\mathrm{T}}\mathrm{x}_n + w_0) \geq 1 \boxed{-\varepsilon_n}, \quad \forall n \quad \varepsilon_n \geq 0$$

Option 1: soft margin SVM





- Remember the constraint below?

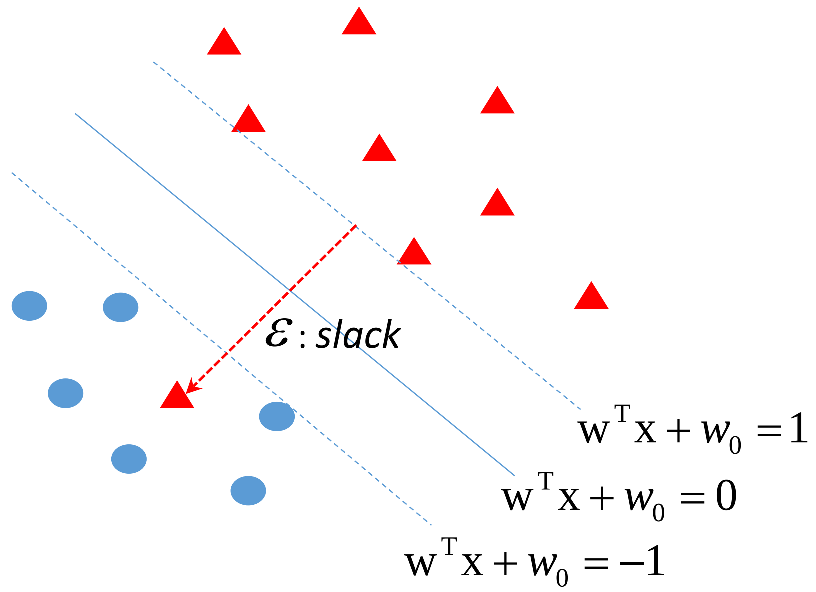
$$t_n(\mathrm{w}^T\mathrm{x}_n + w_0) \geq 1, \quad \forall n$$

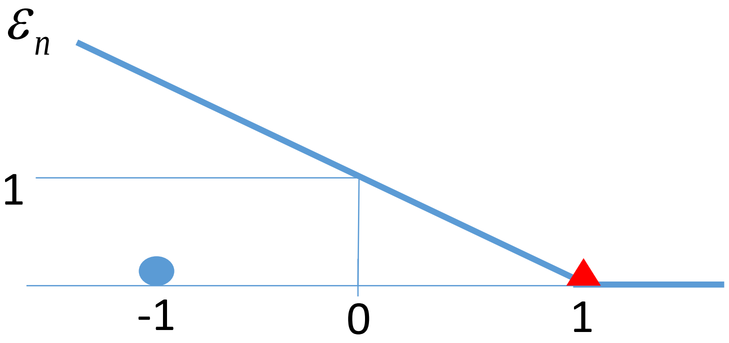
- For the data points which are non-separable, we relax the constraint:

$$t_n(\mathrm{w}^T\mathrm{x}_n + w_0) \geq 1 \boxed{-\varepsilon_n}, \quad \forall n \quad \varepsilon_n \geq 0$$

- It says that the distance between a data point and the decision boundary is allowed to be less than 1.

Option 1: soft margin SVM





- Remember the constraint below?

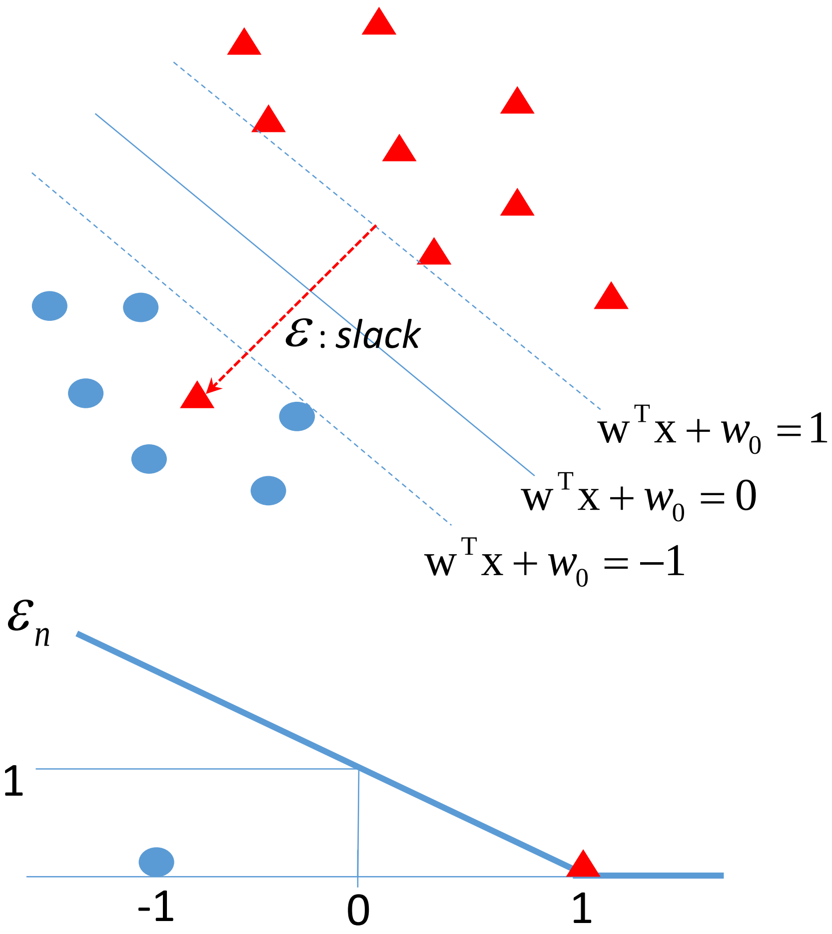
$$t_n(\mathrm{w}^T\mathrm{x}_n+w_0)\geq 1, \quad \forall n$$

- For the data points which are non-separable, we relax the constraint:

$$t_n(\mathrm{w}^T\mathrm{x}_n+w_0)\geq 1\boxed{-\varepsilon_n}, \quad \forall n \quad \varepsilon_n\geq 0$$

- It says that the distance between a data point and the decision boundary is allowed to be less than 1.
- ε_n is called slack variables.

Option 1: soft margin SVM



- Remember the constraint below?

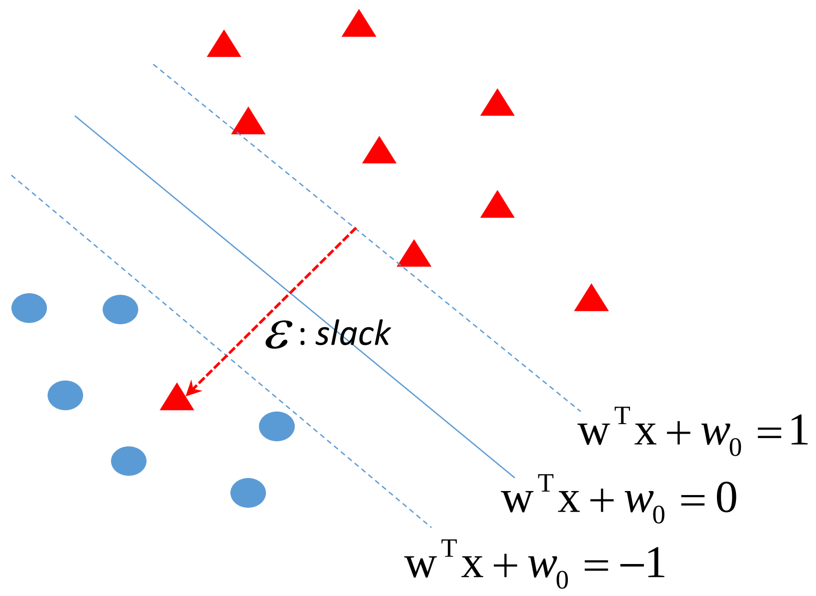
$$t_n(\mathrm{w}^{\mathrm{T}}\mathrm{x}_n + w_0) \geq 1, \quad \forall n$$

- For the data points which are non-separable, we relax the constraint:

$$t_n(\mathrm{w}^{\mathrm{T}}\mathrm{x}_n + w_0) \geq 1 \boxed{- \varepsilon_n}, \quad \forall n \quad \varepsilon_n \geq 0$$

- It says that the distance between a data point and the decision boundary is allowed to be less than 1.
- ε_n is called slack variables.
- Question. Where is a data point when $\varepsilon_n = 1$?

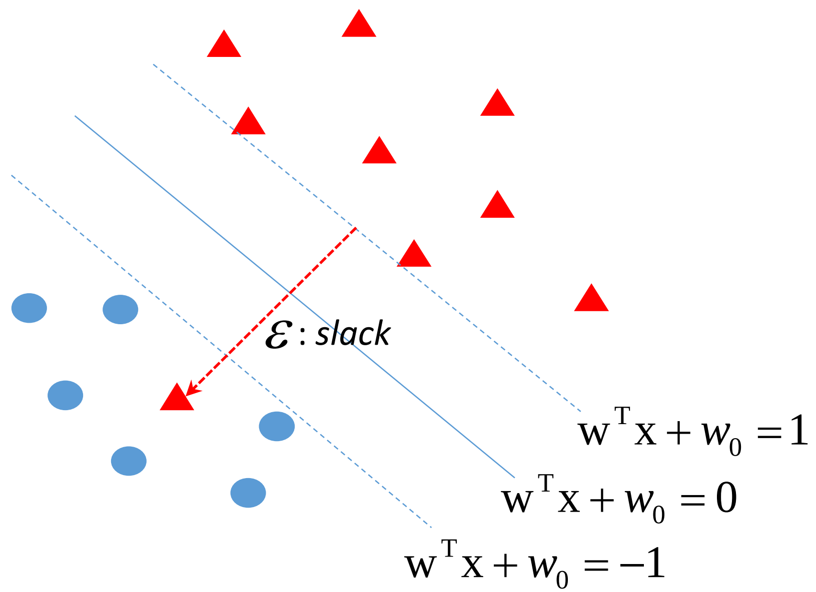
Option 1: soft margin SVM



- So we have the constraint below. How about the objective function?

$$t_n(\mathrm{w}^{\mathrm{T}} x_n + w_0) \geq 1 \boxed{- \varepsilon_n}, \quad \forall n \qquad \varepsilon_n \geq 0$$

Option 1: soft margin SVM



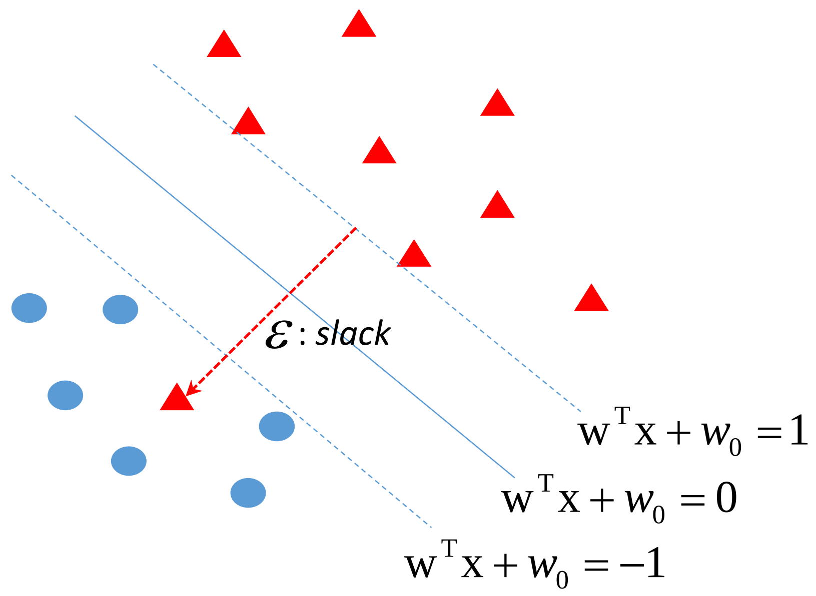
- So we have the constraint below. How about the objective function?

$$t_n(\mathrm{w}^{\mathrm{T}} x_n + w_0) \geq 1 \boxed{- \varepsilon_n}, \quad \forall n \quad \varepsilon_n \geq 0$$

- We want to minimize the slack.

$$\min \frac{1}{2} \| \mathrm{w} \|^2 + C \sum_n \varepsilon_n$$

Option 1: soft margin SVM



- So we have the constraint below. How about the objective function?

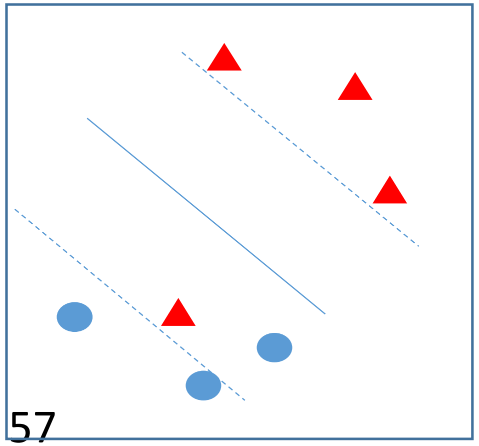
$$t_n(\mathrm{w}^T x_n + w_0) \geq 1 \boxed{- \varepsilon_n}, \quad \forall n \quad \varepsilon_n \geq 0$$

- We want to minimize the slack.

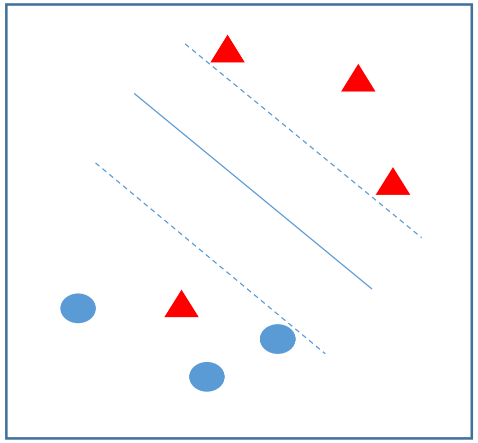
$$\min \frac{1}{2} \| \mathrm{w} \|^2 + C \sum_n \varepsilon_n$$

- If "C" is small, the dominant factor is $\| w \|^2 / 2$
  1) Prefer large margin
  2) May cause large # of misclassified data points.

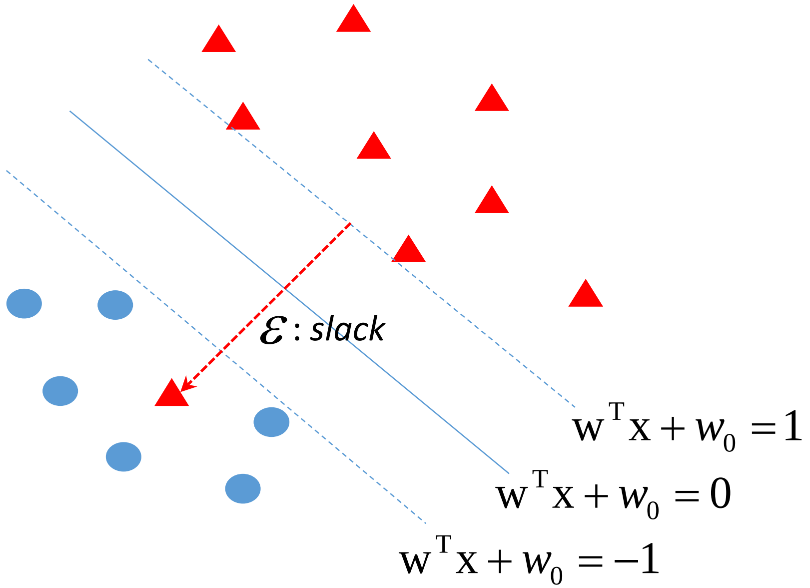
"C" is small



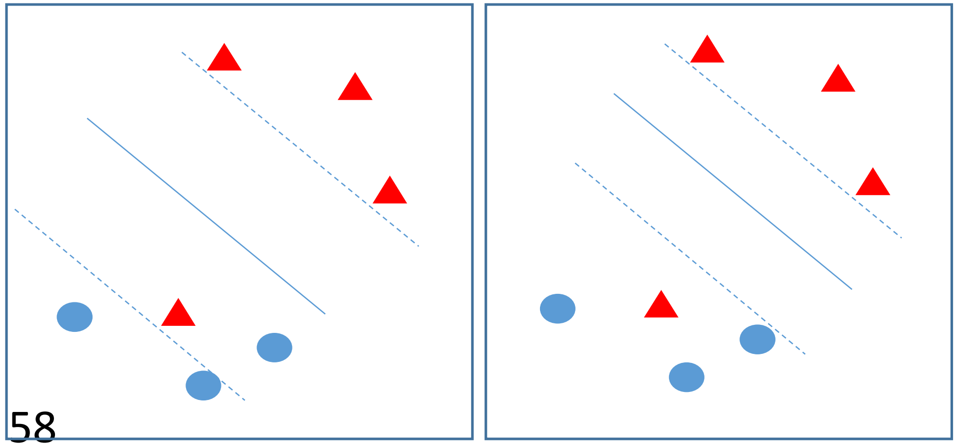
"C" is large



Option 1: soft margin SVM



"C" is small "C" is large



- So we have the constraint below. How about the objective function?

$$t_n(\mathrm{w}^T x_n + w_0) \geq 1 \boxed{- \varepsilon_n}, \quad \forall n \quad \varepsilon_n \geq 0$$

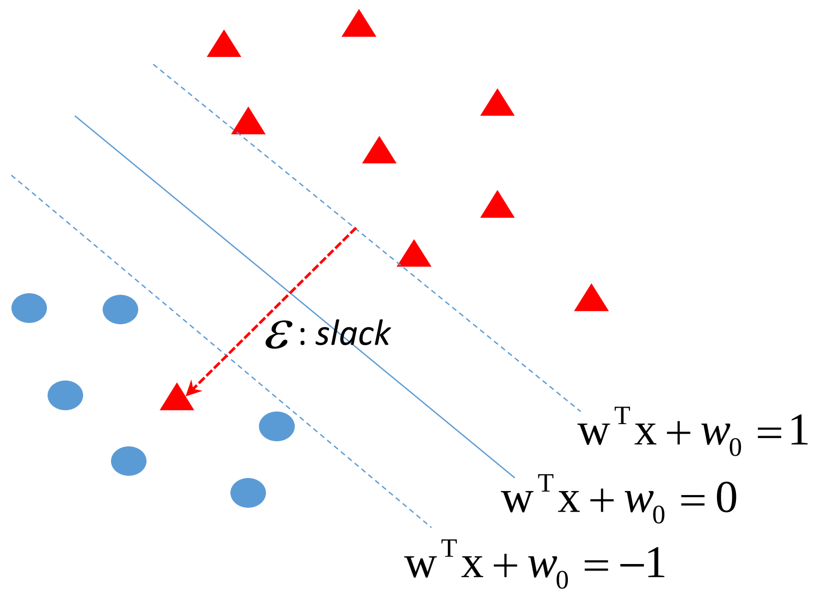
- We want to minimize the slack.

$$\min \frac{1}{2} \| \mathrm{w} \|^2 + C \sum_n \varepsilon_n$$

- If "C" is small, the slack contributes more
  1) Prefer large margin
  2) May cause large # of misclassified data points.

- If "C" is large, the slack contributes less
  1) Prefer less # of misclassified data points.
  2) May cause small margin.

Option 1: soft margin SVM



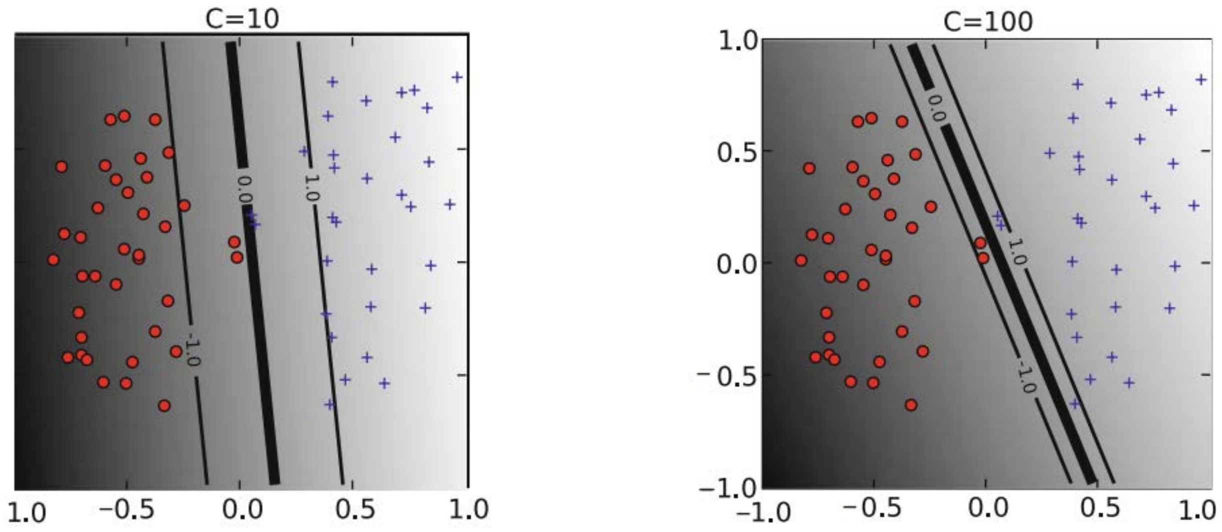
- The formulation finally becomes

$$\min \frac{1}{2}\|w\|^2 + C\sum_n \varepsilon_n$$

$$s.t.$$

$$t_n(w^T x_n + w_0) \geq 1 - \varepsilon_n, \forall n$$

$$\varepsilon_n \geq 0$$



Kernel trick

Kernel trick

$$\min \frac{1}{2} \mathbf{w}^T \mathbf{w}$$

$$s.t. \quad t_n(\mathbf{w}^T x_n + w_0) \geq 1$$

Primal problem

$$\min_{\lambda} \quad L(\lambda) = \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} t_n t_m \lambda_n \lambda_m \mathbf{x}_n^T \mathbf{x}_m - \sum_{n=1}^{N} \lambda_n$$

$$s.t. \quad \lambda \geq 0, \qquad t^T \lambda = 0$$

Dual problem

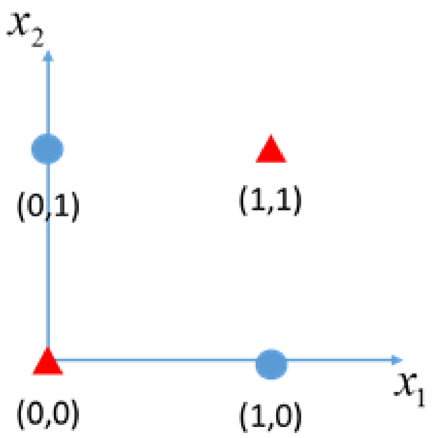
- They are the same problem.
- λ: Lagrange multipliers which corresponding to data points.
- t: label (-1 or 1)
- It looks complicated why we border to use dual problem???

Kernel trick

$$\min_{\lambda} \quad L(\lambda) = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} t_n t_m \lambda_n \lambda_m \mathbf{x}_n^T \mathbf{x}_m - \sum_{n=1}^{N} \lambda_n$$

$$s.t. \quad \lambda \geq 0, \qquad t^T \lambda = 0$$

- If data x_n are not linearly separable, what should we do?



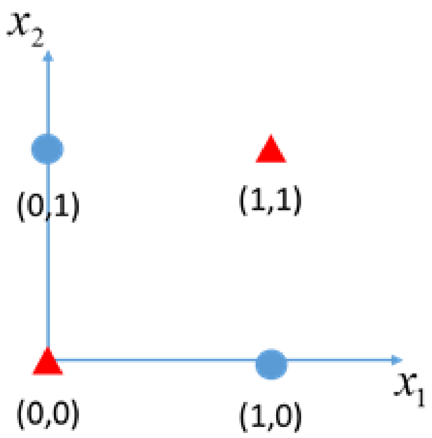
Space X

Kernel trick

$$\min_{\lambda} \quad L(\lambda) = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} t_n t_m \lambda_n \lambda_m \mathrm{x}_n^T \mathrm{x}_m - \sum_{n=1}^{N} \lambda_n$$

$$s.t. \quad \lambda \geq 0, \qquad t^T \lambda = 0$$

- If data x_n are not linearly separable, what should we do?

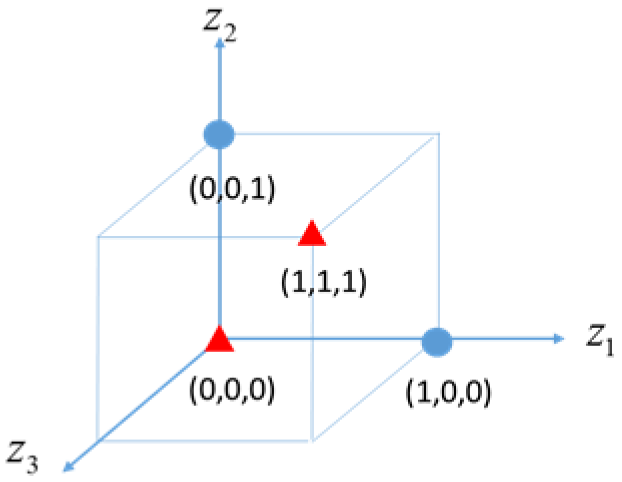


Space X

$$\phi(\mathrm{x}) = \begin{pmatrix} x_1^2 \\ x_1 x_2 \\ x_2^2 \end{pmatrix}$$

$$\phi(\mathrm{x})$$

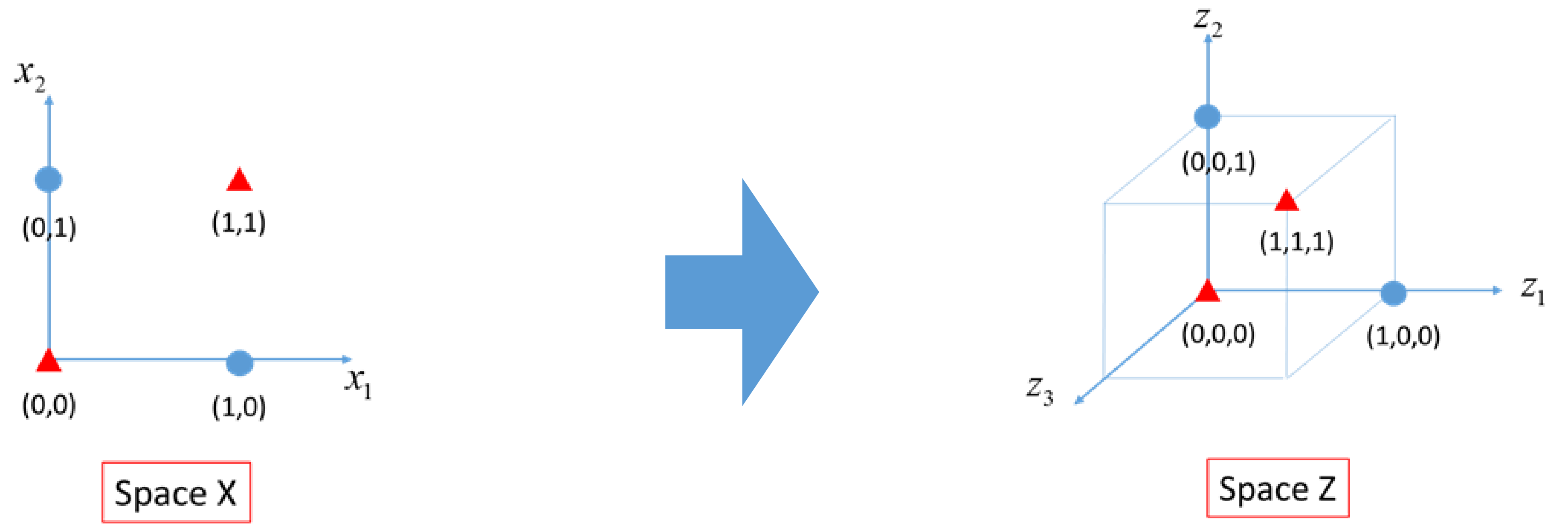
$$X \to Z$$



Space Z

Kernel trick

- The idea of Kernel trick begins from here: to find the scalar values (the inner product of two vectors: z_n and z_m) and so we can formulate the quadratic problem which can be linearly separable.



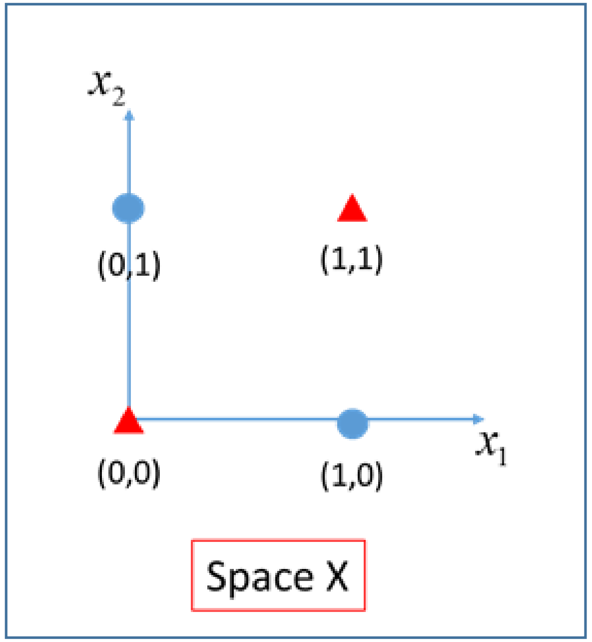
$$\min_{\lambda} \quad L(\lambda) = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} t_n t_m \lambda_n \lambda_m \mathbf{x}_n^T \mathbf{x}_m - \sum_{n=1}^{N} \lambda_n$$
$$s.t. \quad \lambda \geq 0, \qquad t^T \lambda = 0$$

$$\min_{\lambda} \quad L(\lambda) = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} t_n t_m \lambda_n \lambda_m \mathbf{z}_n^T \mathbf{z}_m - \sum_{n=1}^{N} \lambda_n$$
$$s.t. \quad \lambda \geq 0, \qquad t^T \lambda = 0$$

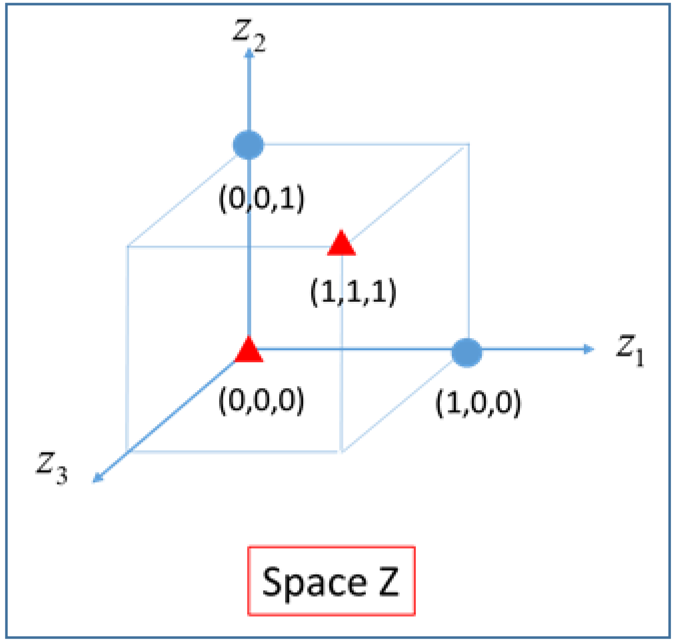
Kernel trick

- Kernel function K() is a function which returns the scalar values (the inner product of two vectors: z_n and z_m in Z space) when the data points (x_n and x_m in X space) are given.

$$K(\mathrm{x}_n^T, \mathrm{x}_m) = \phi(\mathrm{x}_n^T)\phi(\mathrm{x}_m) = \mathrm{z}_n^T \mathrm{z}_m$$



$$\mathrm{z}_n^T = \phi(\mathrm{x}_n^T)$$
$$\mathrm{z}_m = \phi(\mathrm{x}_m)$$



Finally

- With the Kernel function defined previously, we want to change the quadratic problem as follows:
  - Because the Kernel function is a function of data points (x_n and x_m) which we already have.

$$\min_{\lambda} \; L(\lambda) = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} t_n t_m \lambda_n \lambda_m \mathrm{z}_n^T \mathrm{z}_m - \sum_{n=1}^{N} \lambda_n$$
$$s.t. \quad \lambda \geq 0, \qquad t^T \lambda = 0$$

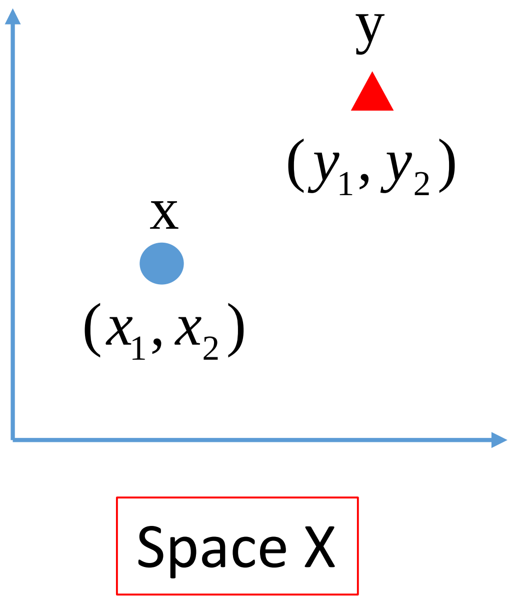
Z space problem

$$\min_{\lambda} \; L(\lambda) = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} t_n t_m \lambda_n \lambda_m \mathrm{K}(\mathrm{x}_n^T \mathrm{x}_m) - \sum_{n=1}^{N} \lambda_n$$
$$s.t. \quad \lambda \geq 0, \qquad t^T \lambda = 0$$

X space problem

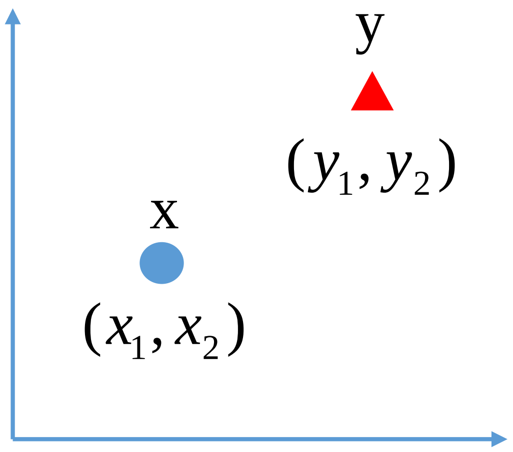
Z space problem can be formulated with data in X space

Polynomial kernel of degree 2



$$K(\mathrm{x}, y) = (\mathrm{xy})^2$$

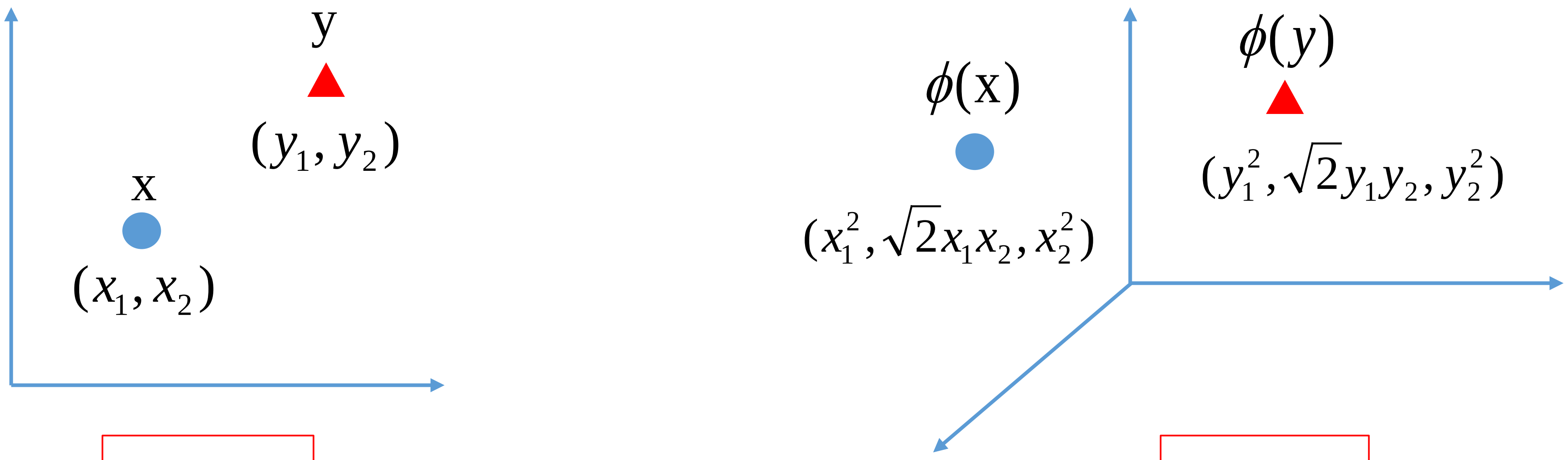
Polynomial kernel of degree 2



Space X

$$
\begin{aligned}
K(\mathrm{x}, y) &= (\mathrm{xy})^2 \\
&= \left((x_1, x_2)\cdot(y_1, y_2)\right)^2 \\
&= \left(x_1 y_1 + x_2 y_2\right)^2 \\
&= x_1^2 y_1^2 + 2x_1 x_2 y_1 y_2 + x_2^2 y_2^2
\end{aligned}
$$

Polynomial kernel of degree 2

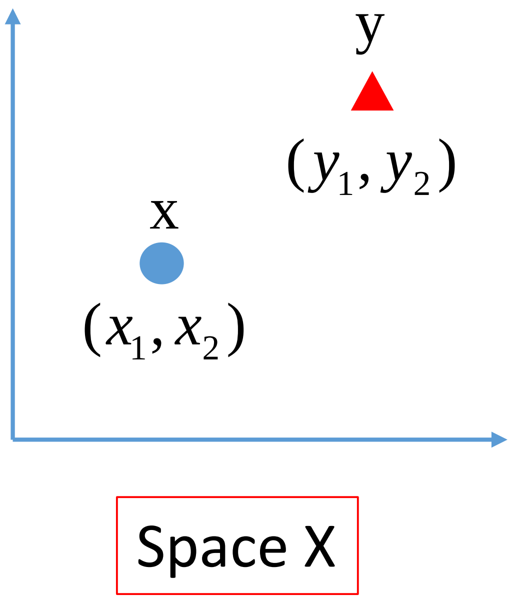


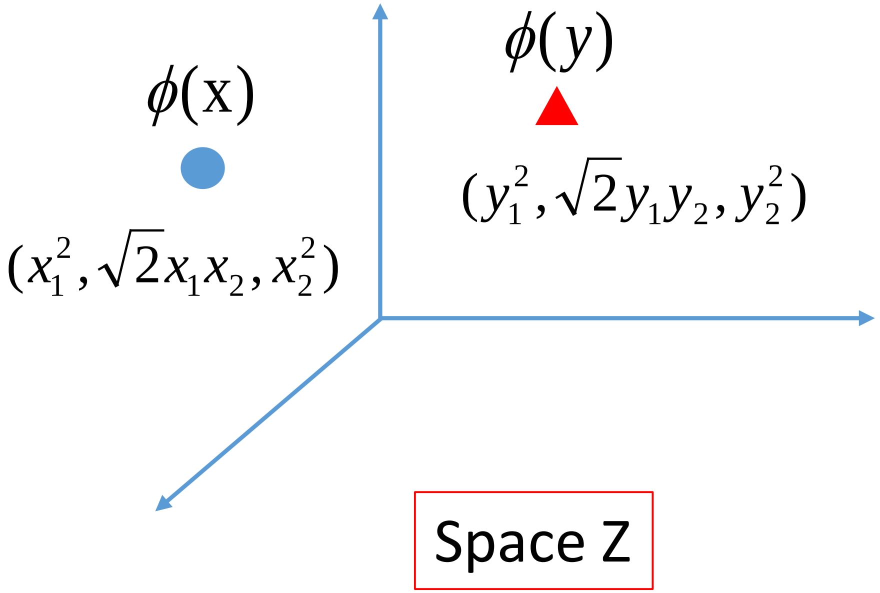
Space X

Space Z

$$
\begin{aligned}
K(\mathrm{x}, y) &= (\mathrm{xy})^2 \\
&= \left((x_1, x_2)\cdot(y_1, y_2)\right)^2 \\
&= \left(x_1y_1 + x_2y_2\right)^2 \\
&= x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2
\end{aligned}
$$

Polynomial kernel of degree 2

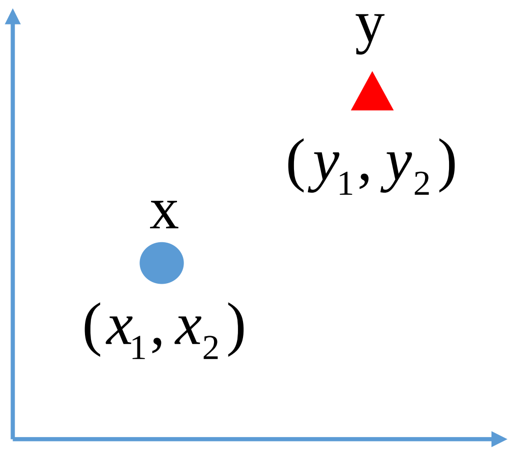




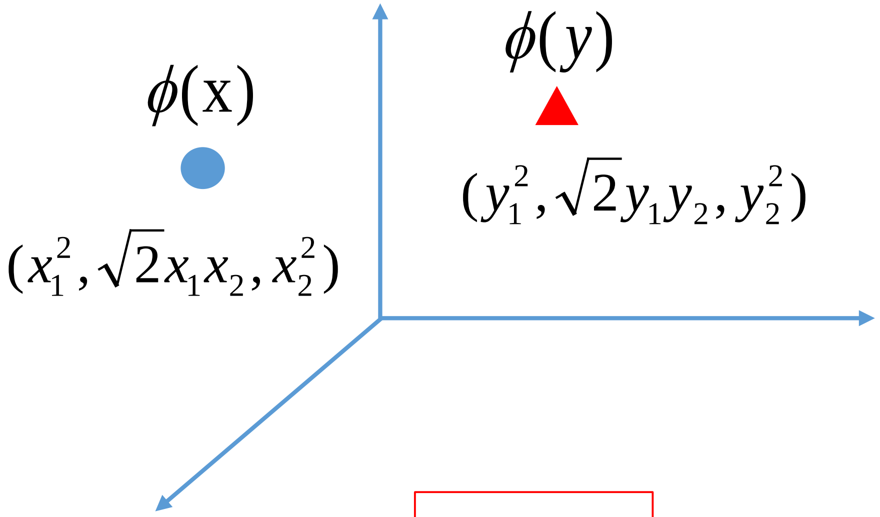
$$\begin{aligned} K(\mathrm{x}, y) &= (\mathrm{xy})^2 \\ &= ((x_1, x_2) \cdot (y_1, y_2))^2 \\ &= (x_1y_1 + x_2y_2)^2 \\ &= x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2 \end{aligned}$$

$$\phi(\mathrm{x})\phi(y) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \cdot (y_1^2, \sqrt{2}y_1y_2, y_2^2)$$

Polynomial kernel of degree 2



Space X



Space Z

$$
\begin{aligned}
K(\mathrm{x}, y) &= (\mathrm{x}y)^2 \\
&= \left((x_1, x_2) \cdot (y_1, y_2)\right)^2 \\
&= \left(x_1y_1 + x_2y_2\right)^2 \\
&= x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2
\end{aligned}
$$

$$
\begin{aligned}
\phi(\mathrm{x})\phi(y) &= (x_1^2, \sqrt{2}x_1x_2, x_2^2) \cdot (y_1^2, \sqrt{2}y_1y_2, y_2^2) \\
&= x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2
\end{aligned}
$$

Mapping to 3-dimension

Gaussian Kernel: derivation (inner product in the infinite z space)

$$K(\mathrm{x}_n, \mathrm{x}_m) = \exp\left(-\alpha \| \mathrm{x}_\mathrm{n} - \mathrm{x}_\mathrm{m} \|^2\right)$$

Gaussian Kernel: derivation (inner product in the infinite z space)

$$K(\mathrm{x}_n, \mathrm{x}_m) = \exp\left(-\alpha \| \mathrm{x}_\mathrm{n} - \mathrm{x}_\mathrm{m} \|^2\right)$$

$$= \exp\left(-\alpha \mathrm{x}_n^2\right)\exp\left(-\alpha \mathrm{x}_m^2\right)\exp\left(2\alpha \mathrm{x}_n \mathrm{x}_m\right)$$

Gaussian Kernel: derivation (inner product in the infinite z space)

Taylor series expansion of an exponential function

$$\exp(x) = \frac{x^0}{0!} + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

$$K(\mathrm{x}_n, \mathrm{x}_m) = \exp\left(-\alpha \, \| \mathrm{x}_n - \mathrm{x}_m \|^2\right)$$

$$= \exp\left(-\alpha \mathrm{x}_n^2\right) \exp\left(-\alpha \mathrm{x}_m^2\right) \exp\left(2\alpha \mathrm{x}_n \mathrm{x}_m\right)$$

$$= \exp\left(-\alpha \mathrm{x}_n^2\right) \exp\left(-\alpha \mathrm{x}_m^2\right) \sum_{k=0}^{\infty} \frac{(2\alpha)^k (\mathrm{x}_n)^k (\mathrm{x}_m)^k}{\mathrm{k}!}$$

Gaussian Kernel: derivation (inner product in the infinite z space)

Taylor series expansion of an exponential function

$$\exp(x) = \frac{x^0}{0!} + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

$$K(\mathrm{x}_n, \mathrm{x}_m) = \exp\left(-\alpha \| \mathrm{x}_\mathrm{n} - \mathrm{x}_\mathrm{m} \|^2\right)$$

$$= \exp\left(-\alpha \mathrm{x}_n^2\right)\exp\left(-\alpha \mathrm{x}_m^2\right)\exp\left(2\alpha \mathrm{x}_n \mathrm{x}_m\right)$$

$$= \exp\left(-\alpha \mathrm{x}_n^2\right)\exp\left(-\alpha \mathrm{x}_m^2\right)\sum_{k=0}^{\infty} \frac{(2\alpha)^k (\mathrm{x}_\mathrm{n})^k (\mathrm{x}_\mathrm{m})^k}{\mathrm{k}!}$$

$$= \sum_{k=0}^{\infty} \sqrt{\frac{(2\alpha)^k}{k!}} \exp\left(-\alpha \mathrm{x}_n^2\right)(\mathrm{x}_\mathrm{n})^k \sqrt{\frac{(2\alpha)^k}{k!}} \exp\left(-\alpha \mathrm{x}_m^2\right)(\mathrm{x}_\mathrm{m})^k$$

Gaussian Kernel: derivation (inner product in the infinite z space)

Taylor series expansion of
an exponential function

$$\exp(x) = \frac{x^0}{0!} + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

$$K(\mathrm{x}_n, \mathrm{x}_m) = \exp\left(-\alpha \| \mathrm{x}_\mathrm{n} - \mathrm{x}_\mathrm{m} \|^2\right)$$

$$= \exp\left(-\alpha \mathrm{x}_n^2\right)\exp\left(-\alpha \mathrm{x}_m^2\right)\exp\left(2\alpha \mathrm{x}_n \mathrm{x}_m\right)$$

$$= \exp\left(-\alpha \mathrm{x}_n^2\right)\exp\left(-\alpha \mathrm{x}_m^2\right)\sum_{k=0}^{\infty} \frac{(2\alpha)^k (\mathrm{x}_\mathrm{n})^k (\mathrm{x}_\mathrm{m})^k}{\mathrm{k}!}$$

$$= \sum_{k=0}^{\infty} \sqrt{\frac{(2\alpha)^k}{k!}} \exp\left(-\alpha \mathrm{x}_n^2\right)(\mathrm{x}_\mathrm{n})^k \sqrt{\frac{(2\alpha)^k}{k!}} \exp\left(-\alpha \mathrm{x}_m^2\right)(\mathrm{x}_\mathrm{m})^k$$

Gaussian Kernel: derivation (inner product in the infinite z space)

Taylor series expansion of an exponential function

$$\exp(x) = \frac{x^0}{0!} + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

$$K(\mathrm{x}_n, \mathrm{x}_m) = \exp\left(-\alpha \| \mathrm{x}_\mathrm{n} - \mathrm{x}_\mathrm{m} \|^2\right)$$

$$= \exp\left(-\alpha \mathrm{x}_n^2\right)\exp\left(-\alpha \mathrm{x}_m^2\right)\exp\left(2\alpha \mathrm{x}_n \mathrm{x}_m\right)$$

$$= \exp\left(-\alpha \mathrm{x}_n^2\right)\exp\left(-\alpha \mathrm{x}_m^2\right)\sum_{k=0}^{\infty} \frac{(2\alpha)^k (\mathrm{x}_\mathrm{n})^k (\mathrm{x}_\mathrm{m})^k}{\mathrm{k}!}$$

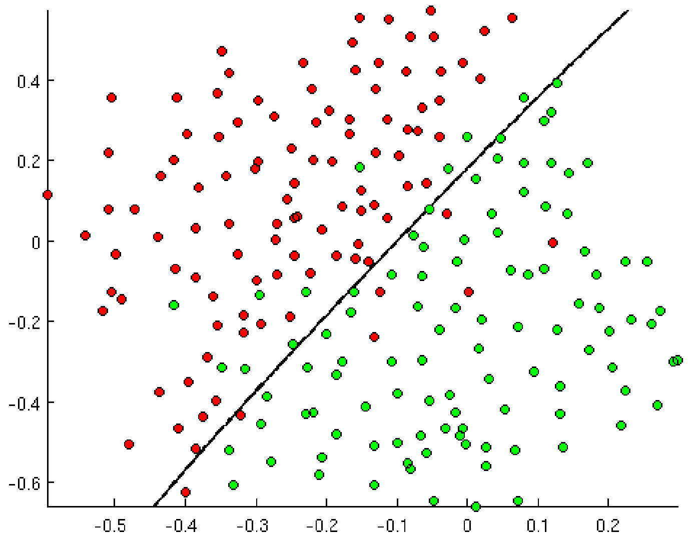
$$= \sum_{k=0}^{\infty} \sqrt{\frac{(2\alpha)^k}{k!}} \exp\left(-\alpha \mathrm{x}_n^2\right)(\mathrm{x}_\mathrm{n})^k \sqrt{\frac{(2\alpha)^k}{k!}} \exp\left(-\alpha \mathrm{x}_m^2\right)(\mathrm{x}_\mathrm{m})^k$$

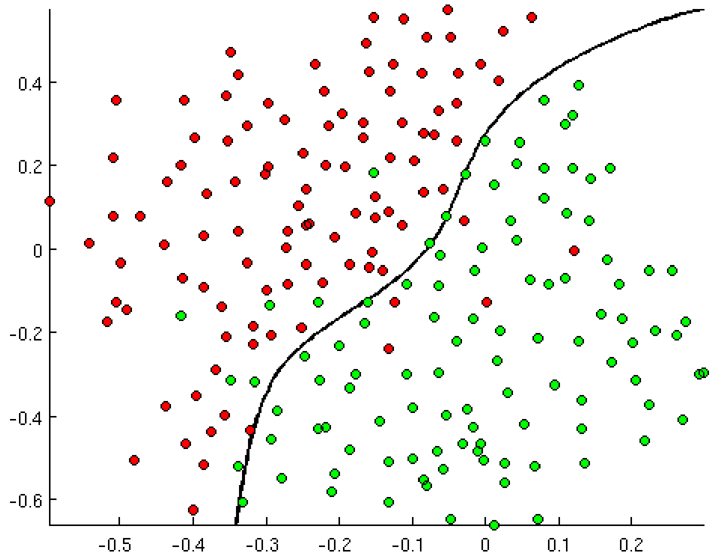
$$= \phi(\mathrm{x}_\mathrm{n})\phi(\mathrm{x}_\mathrm{m})$$

Mapping to infinite-dimension !

Gaussian Kernel

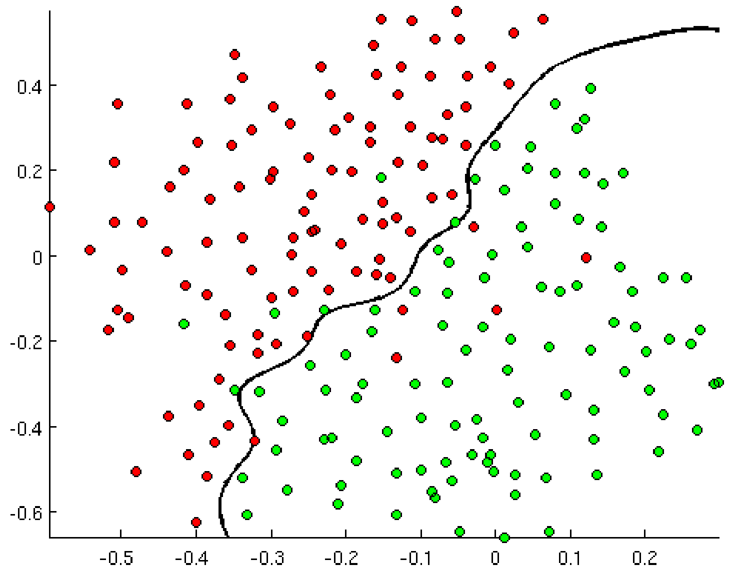
$\alpha = 1$

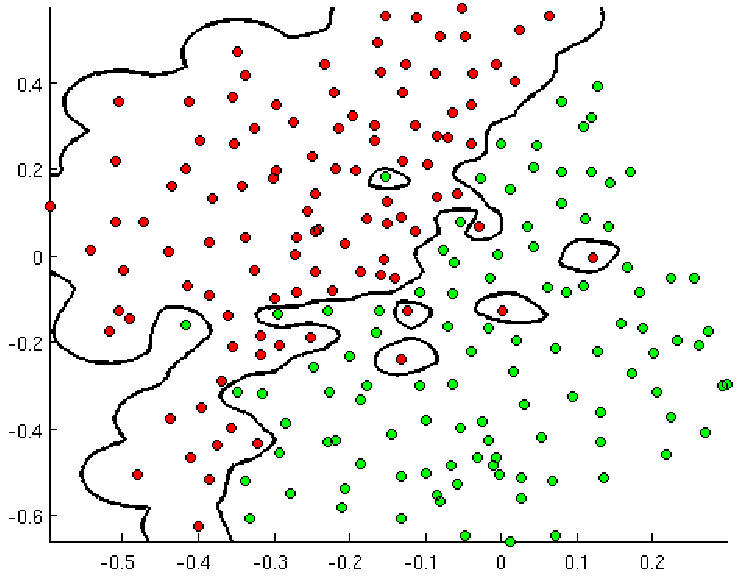




$\alpha = 10$

$\alpha = 100$





$\alpha = 1000$

http://openclassroom.stanford.edu/MainFolder/DocumentPage.php?course=MachineLearning&doc=exercises/ex8/ex8.html

Hand-on Experience

Colab: Google Colaboratory

- A web base free google cloud service
  - Jupyter Notebook with Google Drive

- You can even use GPU for free!
  - Good but it provides the best effort service
    - You must save your things in your google drive or somewhere else.

- Resource check
  - !cat /proc/meminfo
  - !cat /proc/cpuinfo
  - !df -h

Colab: Principal Component Analysis (PCA)

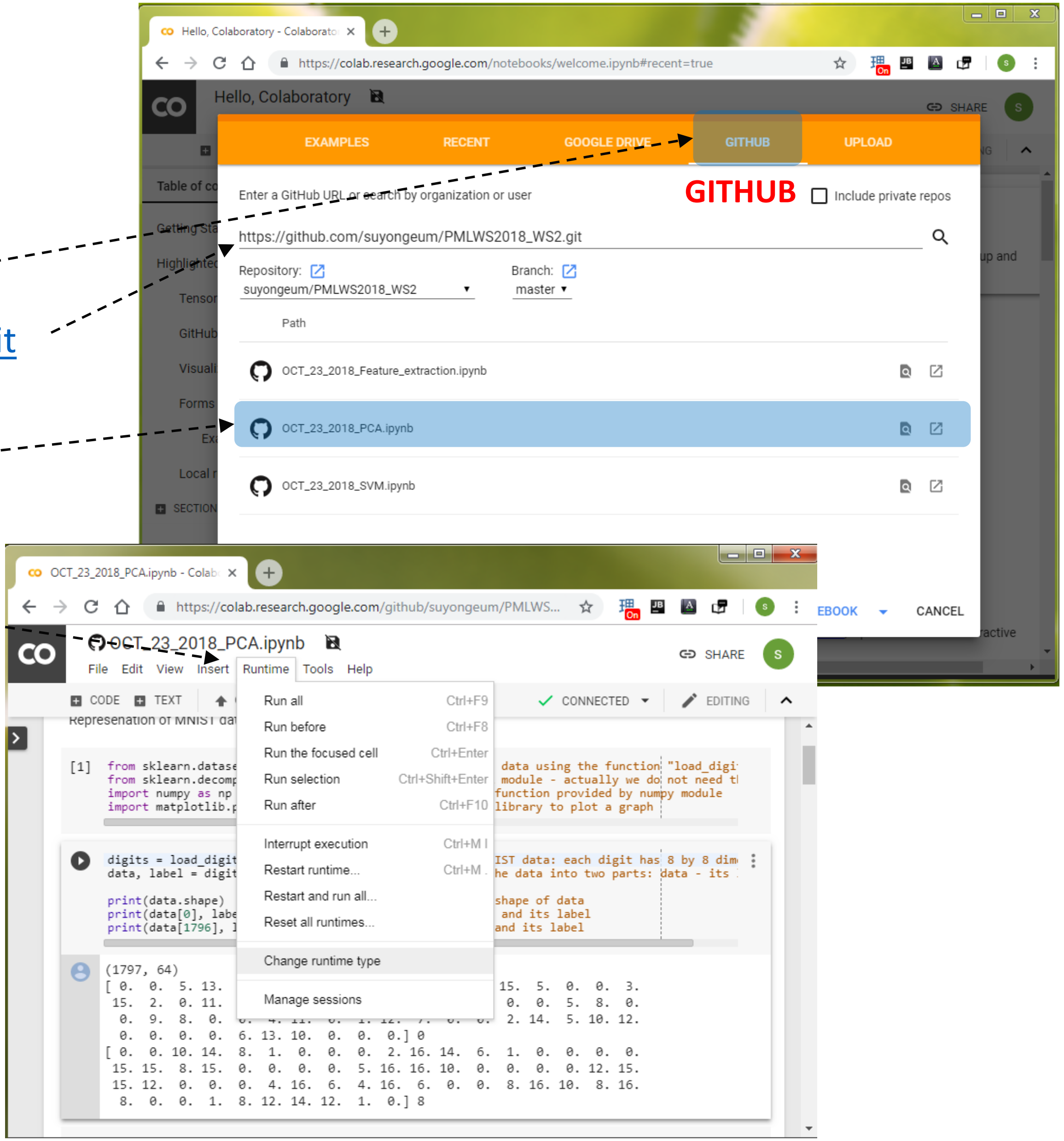
1) Go to the Colab
   - https://colab.research.google.com

2) Select "GITHUB" and copy the link below into
   - https://github.com/suyongeum/PMLWS2018_WS2.git

3) Select the notebook in the list
   - OCT_23_2018_PCA.ipynb

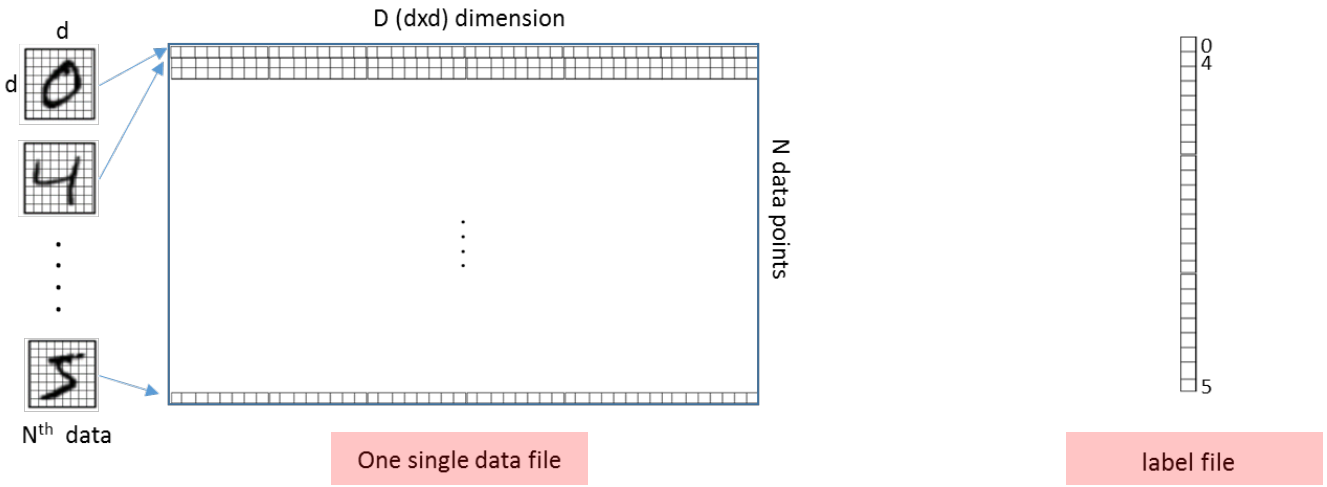
4) Go to "Runtime" – "Change runtime type"
   - Python 3
   - GPU

5) Save it into your gdrive
   - "File" - "Save a copy in Drive …"



Data loading: MNIST

- MNIST data set
  - http://yann.lecun.com/exdb/mnist/
  - Training data
    - One single file (45M) which includes 60,000 hand digit images for training,
    - One single file (59K) which includes corresponding labels.
  - Testing data
    - One single file (7.5M) which includes 10,000 hand digit images for testing,
    - One single file (9.8K) which includes corresponding labels.



Making your gdrive as a working directory

- Defining a root directory where you mount your gdrive

```
from google.colab import drive
drive.mount('/gdrive/')
```

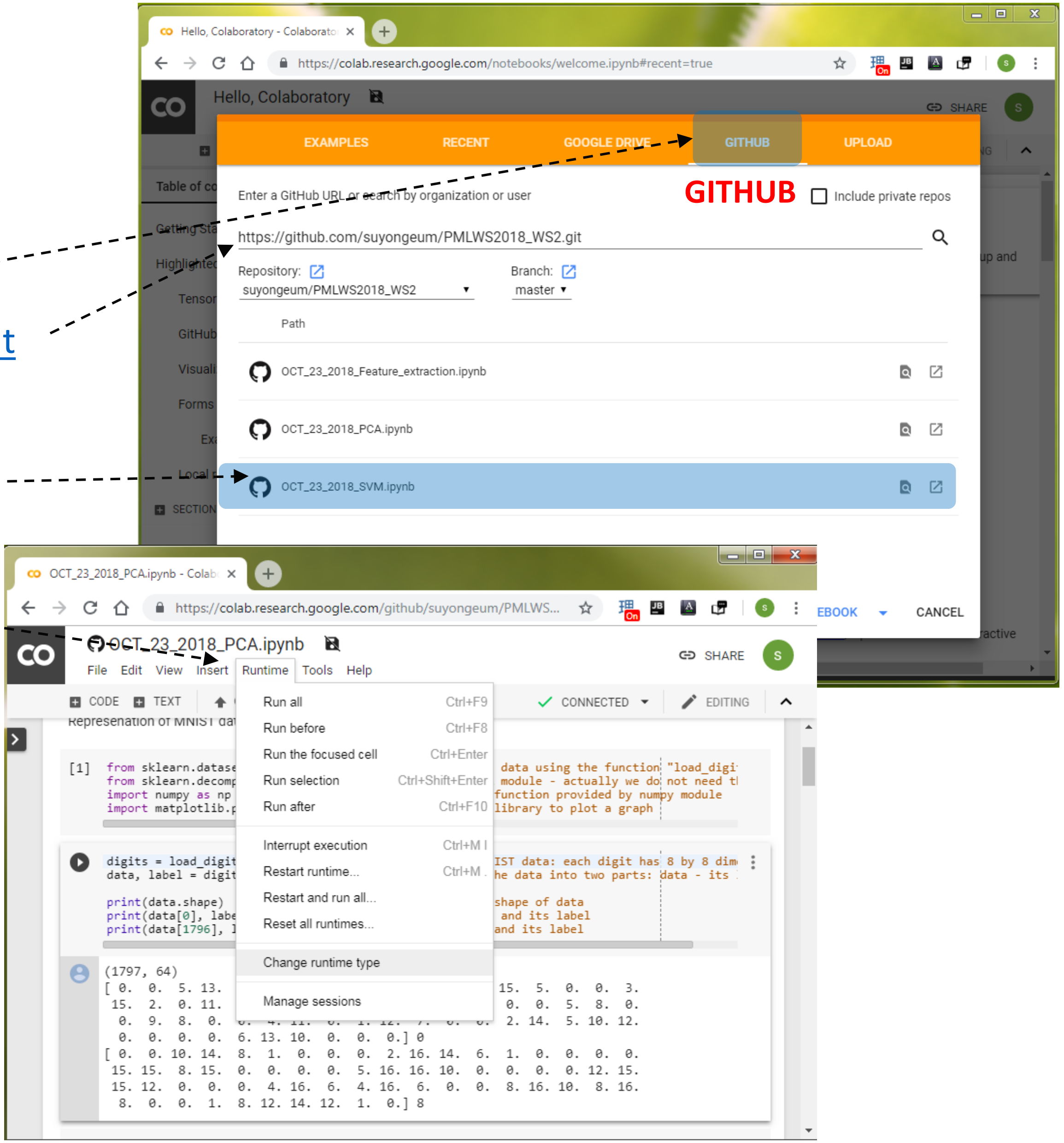
  - /gdrive/My Drive/Colab Notebooks/

- Running time measurement

```
import datetime
before = datetime.datetime.now().timestamp()

…

after = datetime.datetime.now().timestamp()
print( "Time taken:", after – before)
```

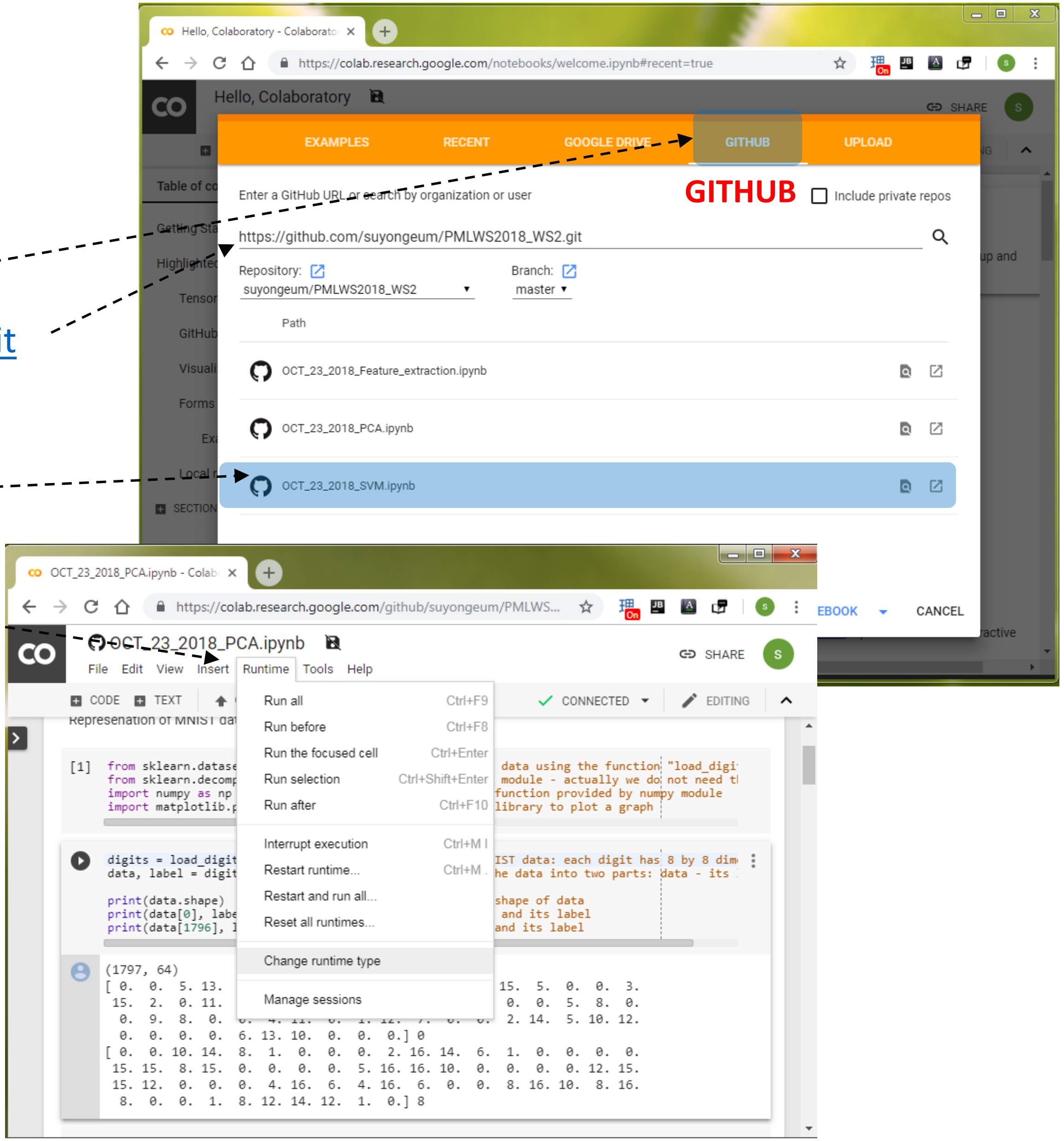
Colab: Support Vector Machine (SVM)

1) Go to the Colab
   - https://colab.research.google.com
2) Select "GITHUB" and copy the link below into
   - https://github.com/suyongeum/PMLWS2018_WS2.git
3) Select the notebook in the list
   - OCT_23_2018_SVM.ipynb
4) Go to "Runtime" – "Change runtime type"
   - Python 3
   - GPU
5) Save it into your gdrive
   - "File" - "Save a copy in Drive …"



Colab: Feature extraction

1) Go to the Colab
   - https://colab.research.google.com
2) Select "GITHUB" and copy the link below into
   - https://github.com/suyongeum/PMLWS2018_WS2.git
3) Select the notebook in the list
   - OCT_23_2018_Feature_extraction.ipynb
4) Go to "Runtime" – "Change runtime type"
   - Python 3
   - GPU
5) Save it into your gdrive
   - "File" - "Save a copy in Drive …"



Backup Slides

Finally finally…

$$\min_{\lambda} \; L(\lambda) = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} t_n t_m \lambda_n \lambda_m \mathrm{K}(\mathrm{x}_n^T \mathrm{x}_m) - \sum_{n=1}^{N} \lambda_n$$

$$s.t. \quad \lambda \geq 0, \qquad t^T \lambda = 0$$

$$\mathrm{w} = \sum_{z_n \in SV} \lambda_n t_n \mathrm{z}_n \qquad w_0 = t_n - \sum_{z_n \in SV} \lambda_n t_n z_n z_n = t_n - \sum_{z_n \in SV} \lambda_n t_n K(\mathrm{x}_n, \mathrm{x}_n)$$

$$\mathrm{sign}(\mathrm{w}^{\mathrm{T}} \mathrm{z} + w_0)$$

$$\mathrm{sign}\left(\sum \lambda_n t_n \mathrm{z}_n \mathrm{z} + t_n - \sum_{z_n \in SV} \lambda_n t_n K(\mathrm{x}_n, \mathrm{x}_n) \right)$$

$$\mathrm{sign}\left(\sum \lambda_n t_n K(x_n, x) + t_n - \sum \lambda_n t_n K(x_n, x_n) \right)$$

- Now you have a function, which classifies a data point in z space without mapping the data point to z space at all.
- Do you see why it is called a trick?