国際融合科学論/先端融合科学論

LECTURE 02
Machine Learning I: conventional machine learning

Dr. Suyong Eum

OSAKA UNIVERSITY

Lecture Outline

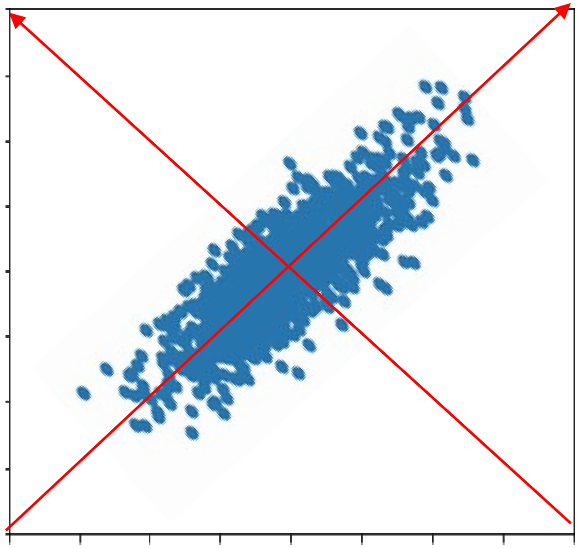
1) Principal Component Analysis (PCA)
   - Feature selections
   - Dimension reduction
2) Support Vector Machine (SVM)
   - Hard margin SVM: linear classification
   - Kernel trick: nonlinear classification

Principal Component Analysis (PCA)

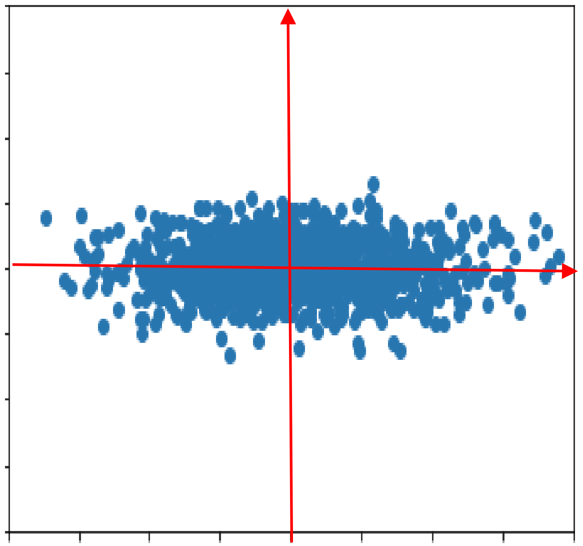
Principal Component Analysis (PCA): definition

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:

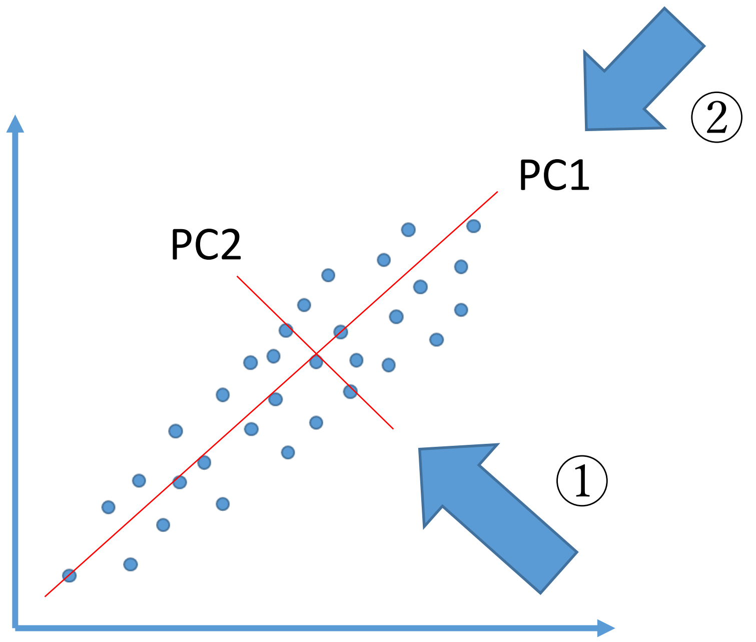


$$\begin{bmatrix} 8 & 1 \\ 1 & 8 \end{bmatrix}$$

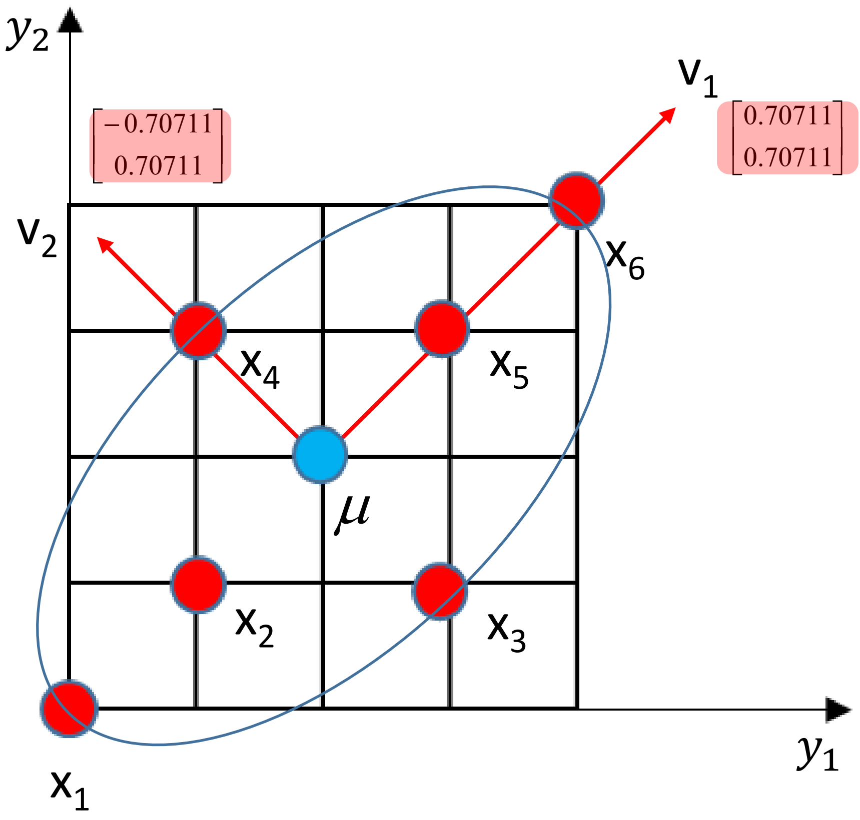


$$\begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}$$

- How to select principal components?
  - One that captures the largest variance of the data points
  - Intuitively speaking, you can observe more data from the direction ① than any other direction, and then from the direction ②, you can observe the data with the least redundancy compared to the direction 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"

$$= [2\sqrt{2},\ \sqrt{2},\ 0,\ 0,\ \sqrt{2},\ 2\sqrt{2}]$$ Variance = 4

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

$$= [0,\ 0,\ \sqrt{2},\ \sqrt{2}, 0, 0]$$ Variance = 0.8

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

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

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

  -0.70711   0.70711
   0.70711   0.70711
```

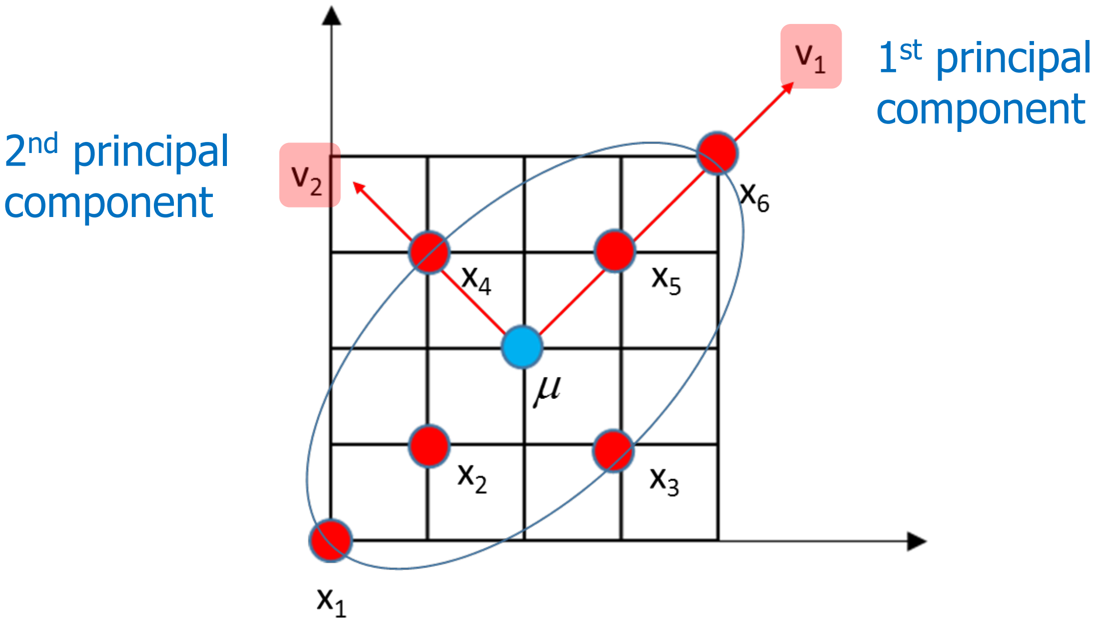
V

```
val =

Diagonal Matrix

   0.80000         0
         0   4.00000
```

Λ

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 value 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: $(-0.70711, 0.70711)$ — eigen value 0.80000

1st principal component: $(0.70711, 0.70711)$ — eigen value 4.00000

How to find the principal components showing the largest variance?

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

Eigen decomposition

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

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
```

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

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

$$= \boxed{V\Sigma^2 V^T}$$

$$\Lambda = \Sigma^2$$

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

Eigen value

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

Singular value

How to find the principal components showing the largest variance?

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

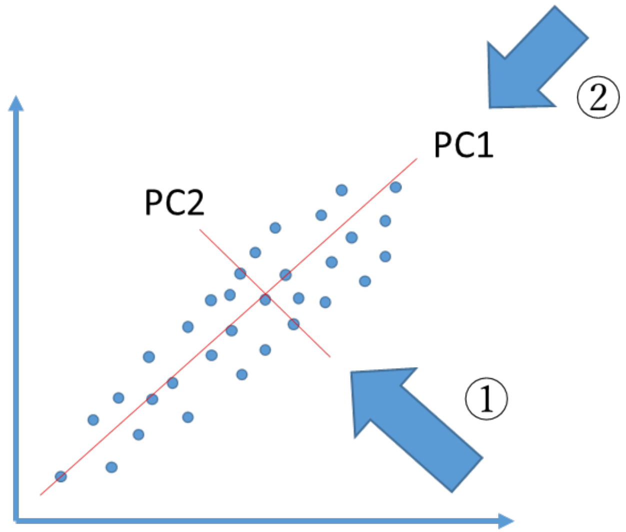
$$\Lambda = \Sigma^2$$

$$4.4721^2 = 20$$

```
>> [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

- Principal Component Analysis (PCA) is nothing but finding principal components of a given data set,
  - Principal components are the directions where you look at the data set, which provides the most information of the data set.
  - They're equivalent to eigen vectors which can be found by SVD or EVD.
  - The eigen value corresponding to each eigen vector represents how widely the data set is spread along the direction which is perpendicular to the eigen vector.

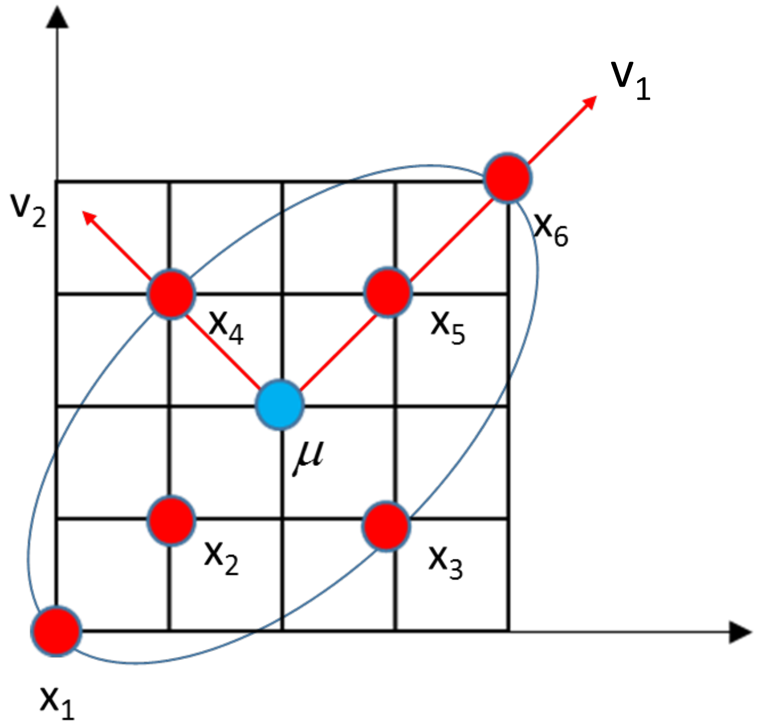


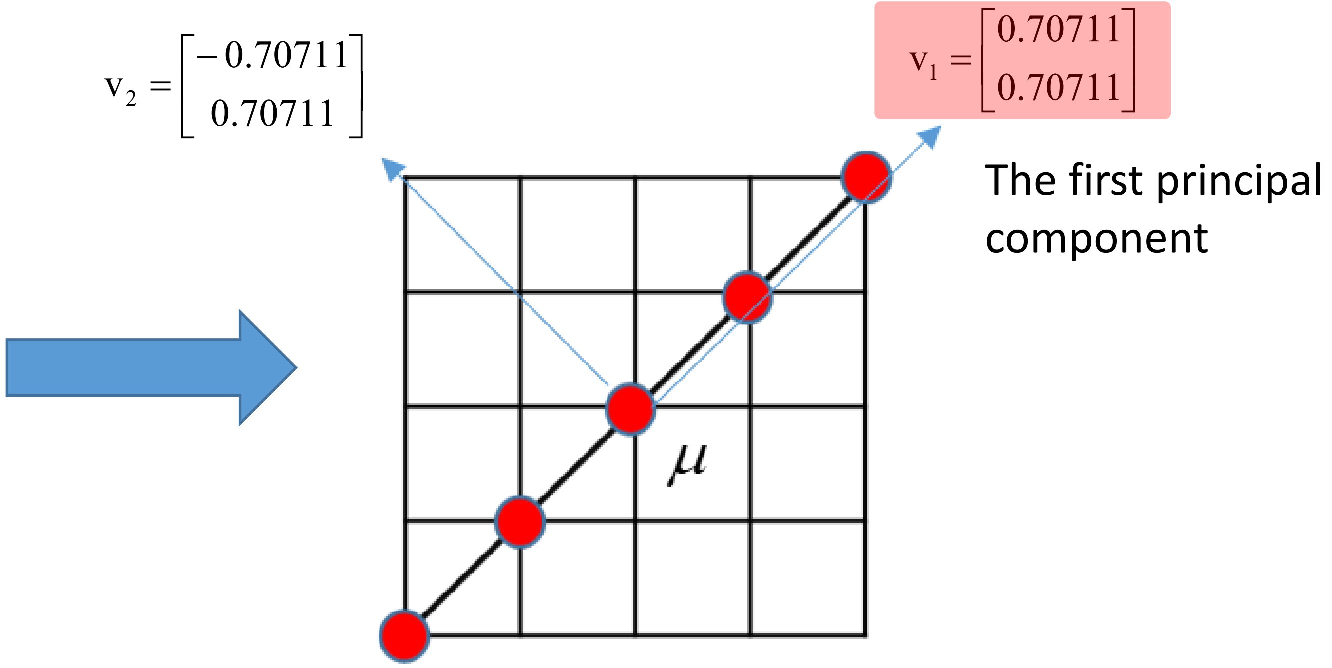
Dimension Reduction

- A data point is defined by several, let’s say, features,
- The number of features to define a data point is called the dimension of the data,
- High dimension data implies that it contains much information,
- Sometimes, we reduce its dimension, e.g., to visualize the data or to efficiently analyze them,
- PCA can reduce the dimension without losing relatively less information of the data.

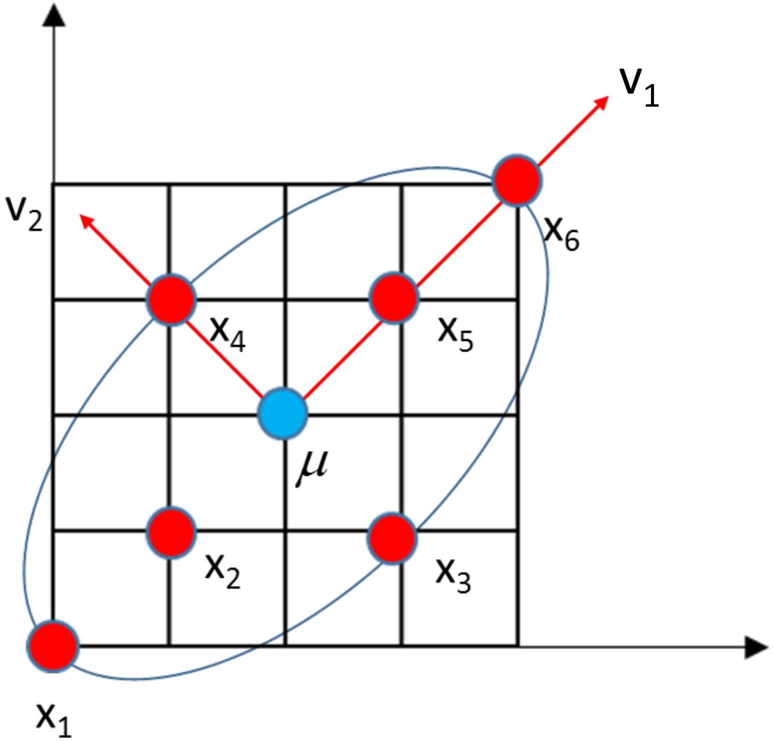
Dimension Reduction

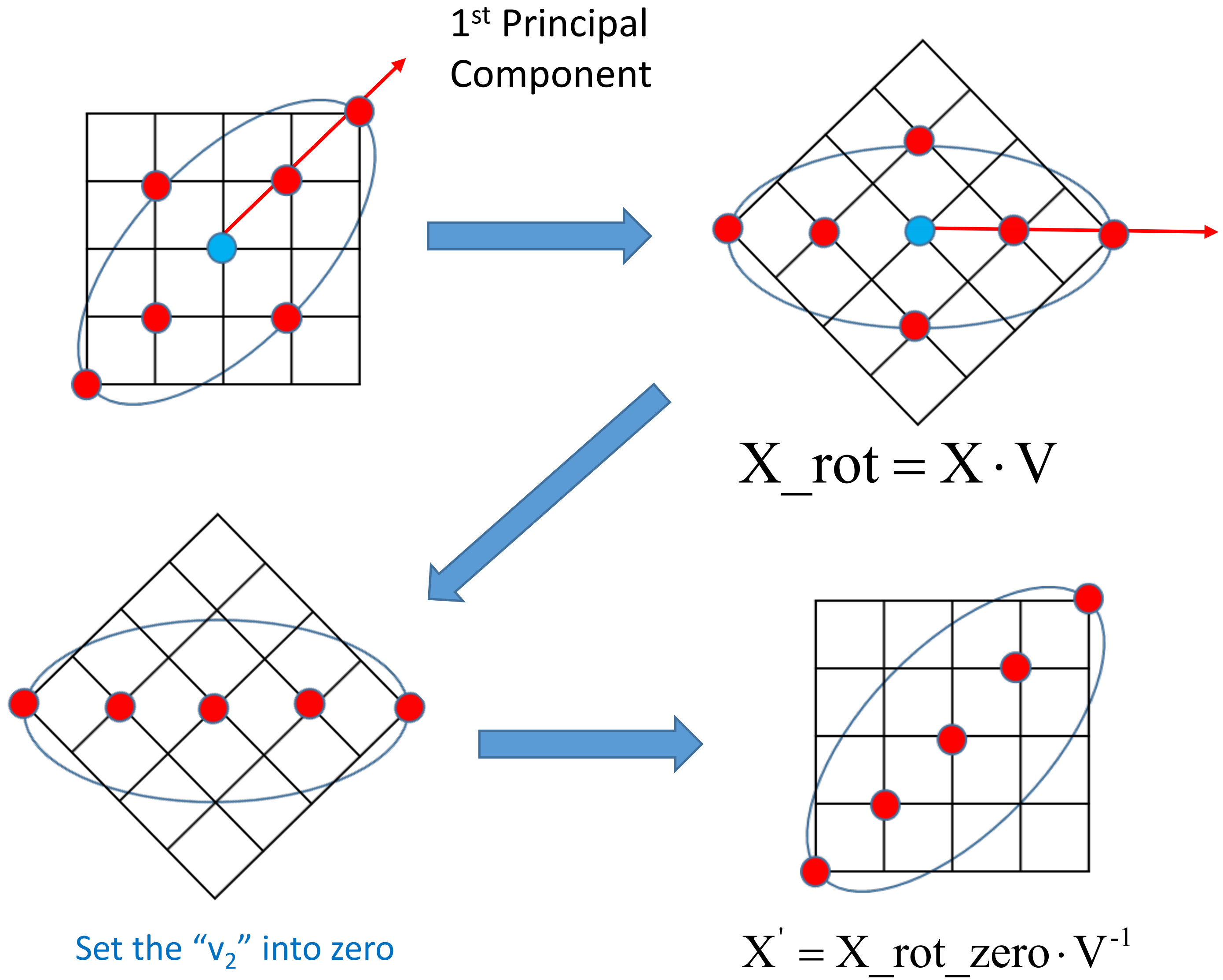
- The previous example shows the case of two-dimensional data
- How can we reduce the two-dimensional data to one dimension?
- Yes, just project the data points onto the eigenvector space!





Dimension Reduction

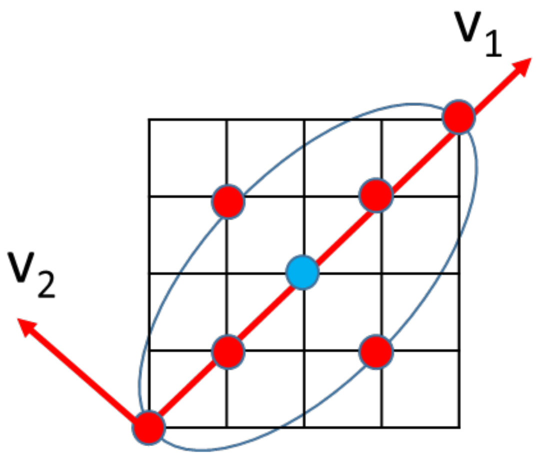




$$X\_rot = X \cdot V$$

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

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

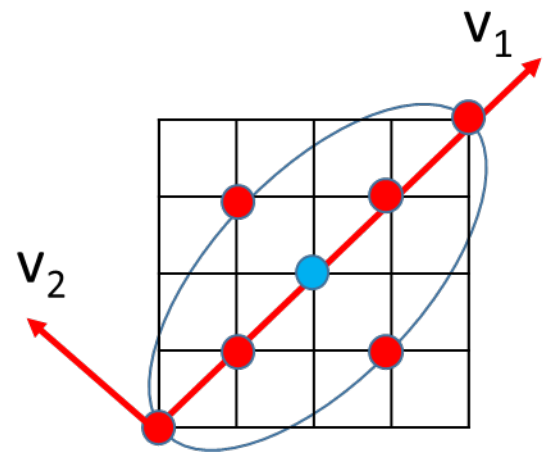


$$X = \begin{bmatrix} -2 & -2 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}$$

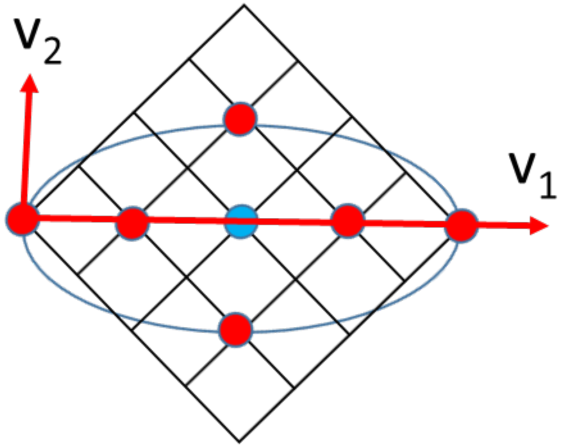
$$X = U\Sigma V^T$$

```
>> [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
```

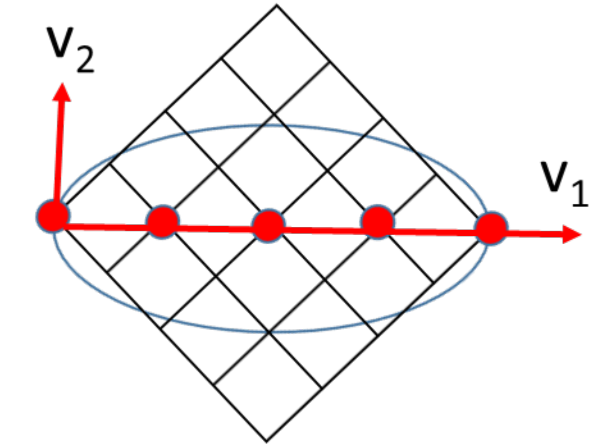
Dimension Reduction



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



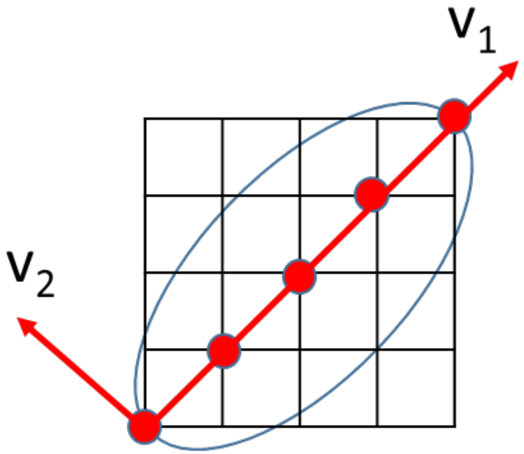
$$\begin{array}{ccc} \text{X\_rot} & & \text{X} \quad\quad \text{V} \\ \begin{bmatrix} -2\sqrt{2} & 0 \\ -\sqrt{2} & 0 \\ 0 & -\sqrt{2} \\ 0 & \sqrt{2} \\ \sqrt{2} & 0 \\ 2\sqrt{2} & 0 \end{bmatrix} & = & \begin{bmatrix} -2 & -2 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix} \begin{bmatrix}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



Set the "v_2" elements into zero

X_rot_zero

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

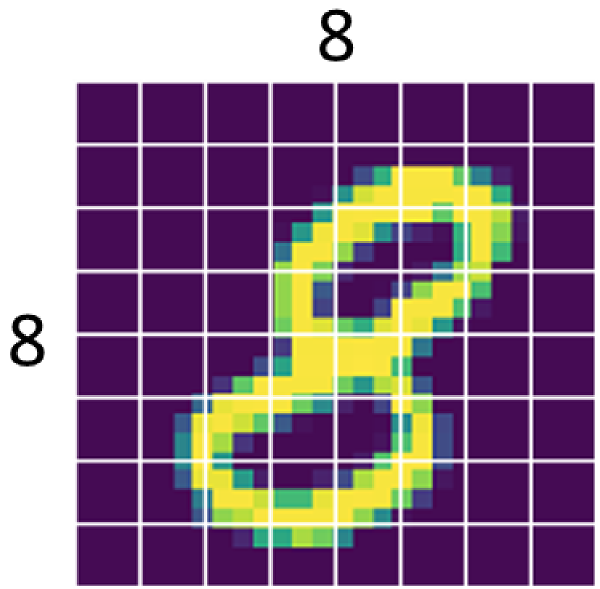


$$\begin{bmatrix} -2 & -2 \\ -1 & -1 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}$$

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

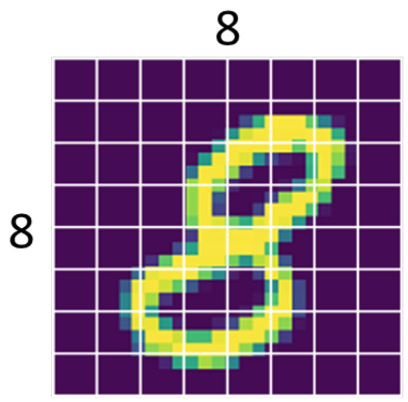
Dimension Reduction: an example

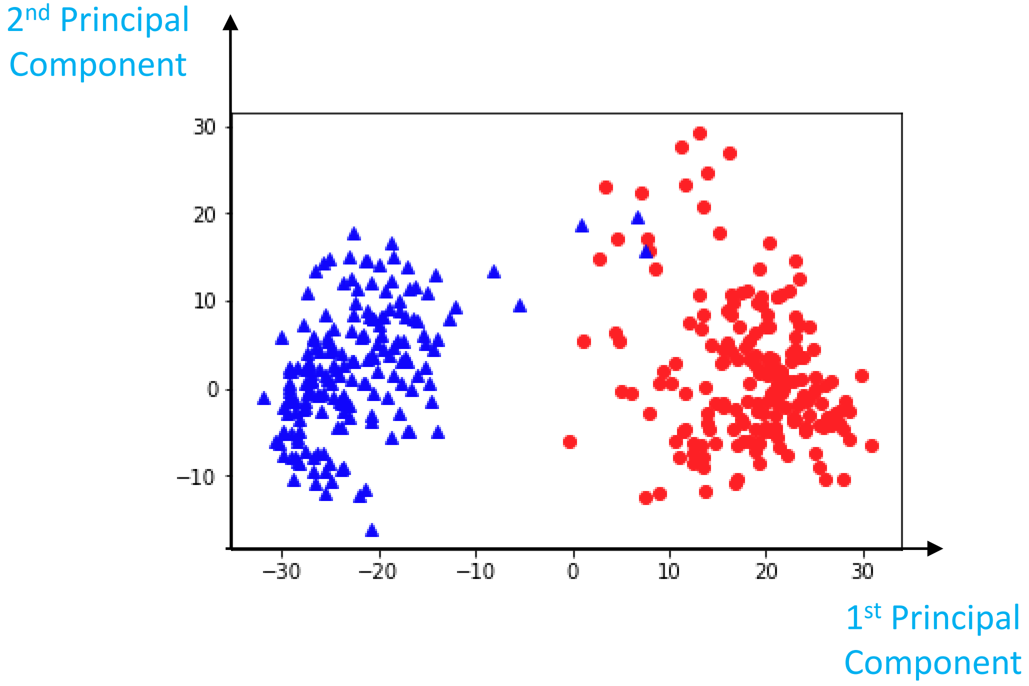
- Let's say, we have one image representing one data point as shown below,
- Then, we decide to present the data by all pixels which are 64 in this case, in other words, it is 64-dimensional data,
- What happens if we reduce its dimension to 2 dimension?



Dimension Reduction: an example

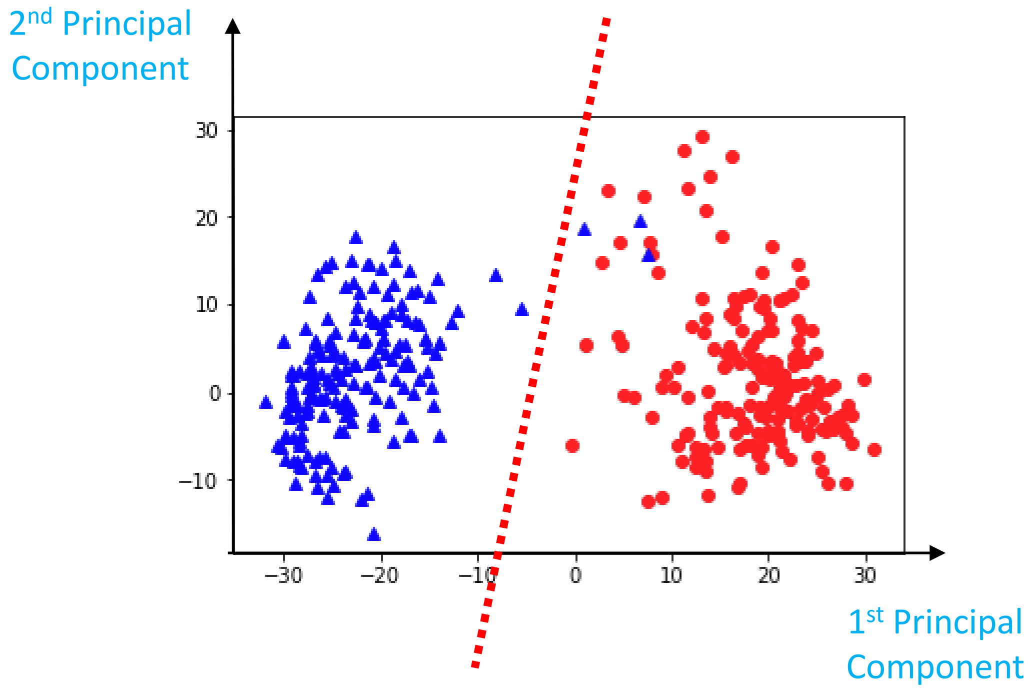
- Let's say, we have one image representing one data point as shown below,
- Then, we decide to present the data by all pixels which are 64 in this case, in other words, it is 64-dimensional data,
- What happens if we reduce its dimension to 2 dimension?





Dimension Reduction: an example

- Well, now we have a new set of data which have two dimension, so they can be presented in the two-dimensional space. Data visualization!
- Also, we may be able to classify those data by drawing a line???



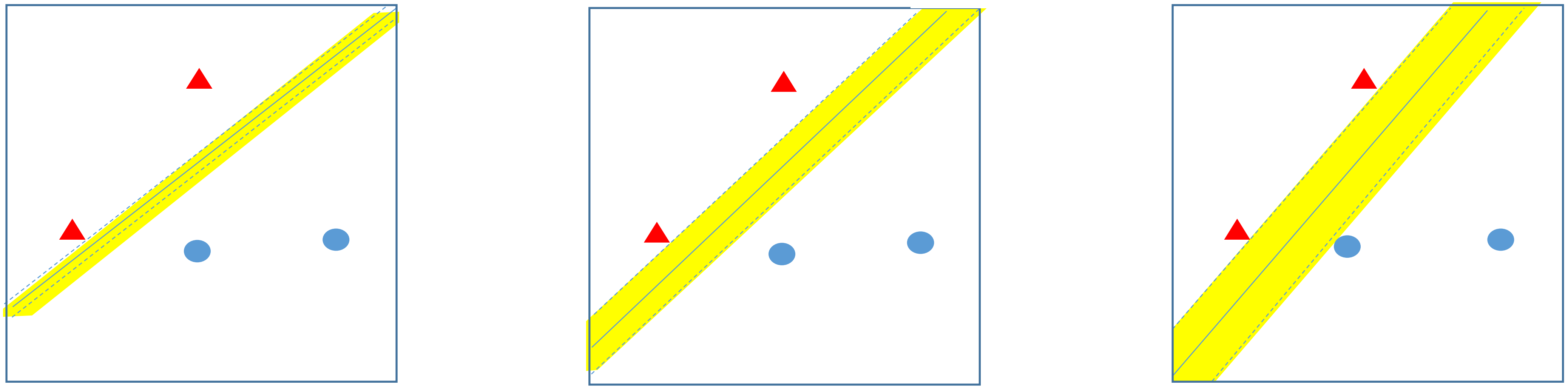
Support Vector Machine (SVM)

Why Support Vector Machine?

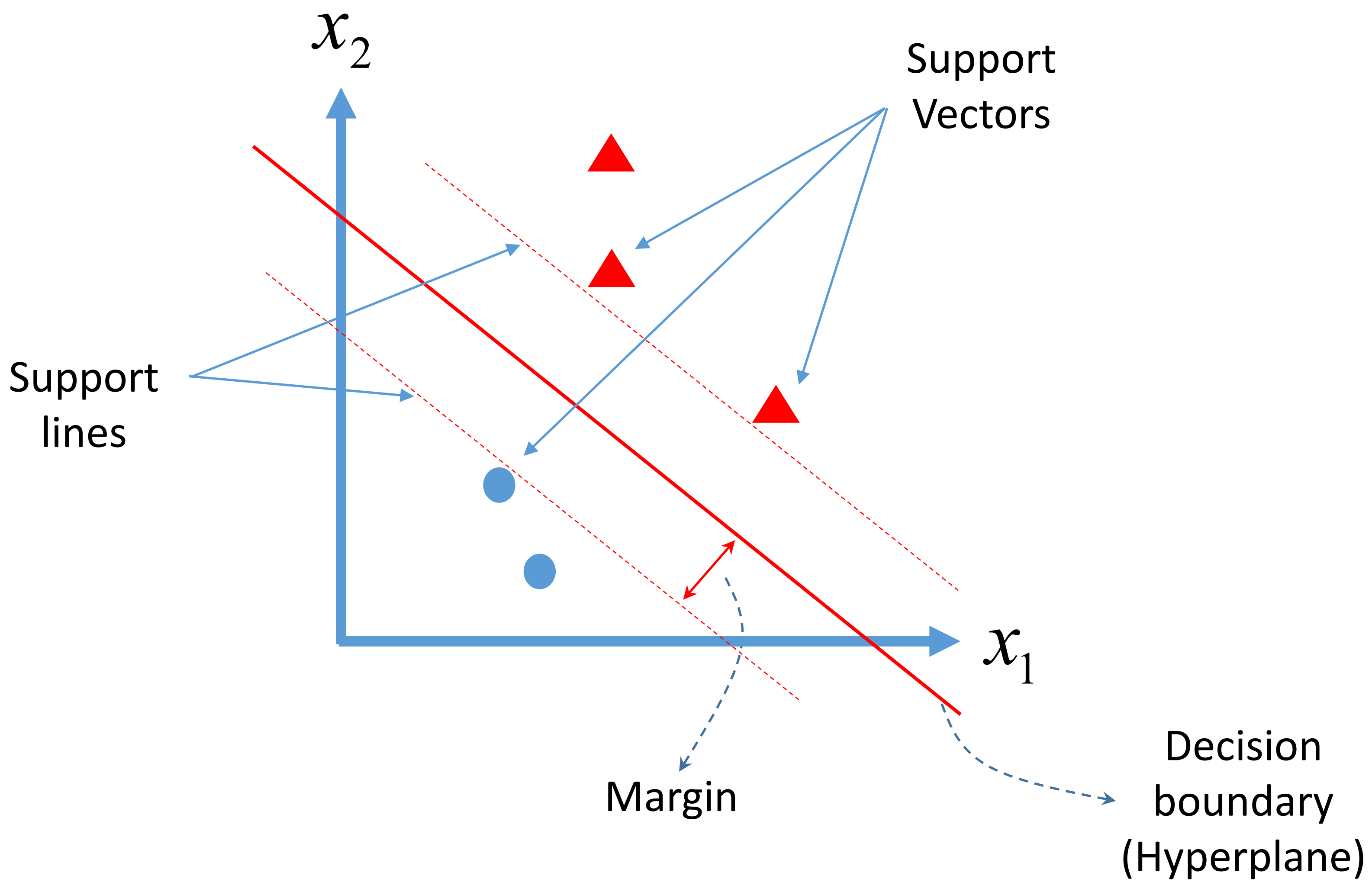
- Most widely used classification approach (practical)
  - Linearly separable data set
  - Non-linearly separable data set

- Supported by well defined mathematical theories
  - Geometry
  - Optimization

Which line is better to split two data sets?



Terminology used in SVM



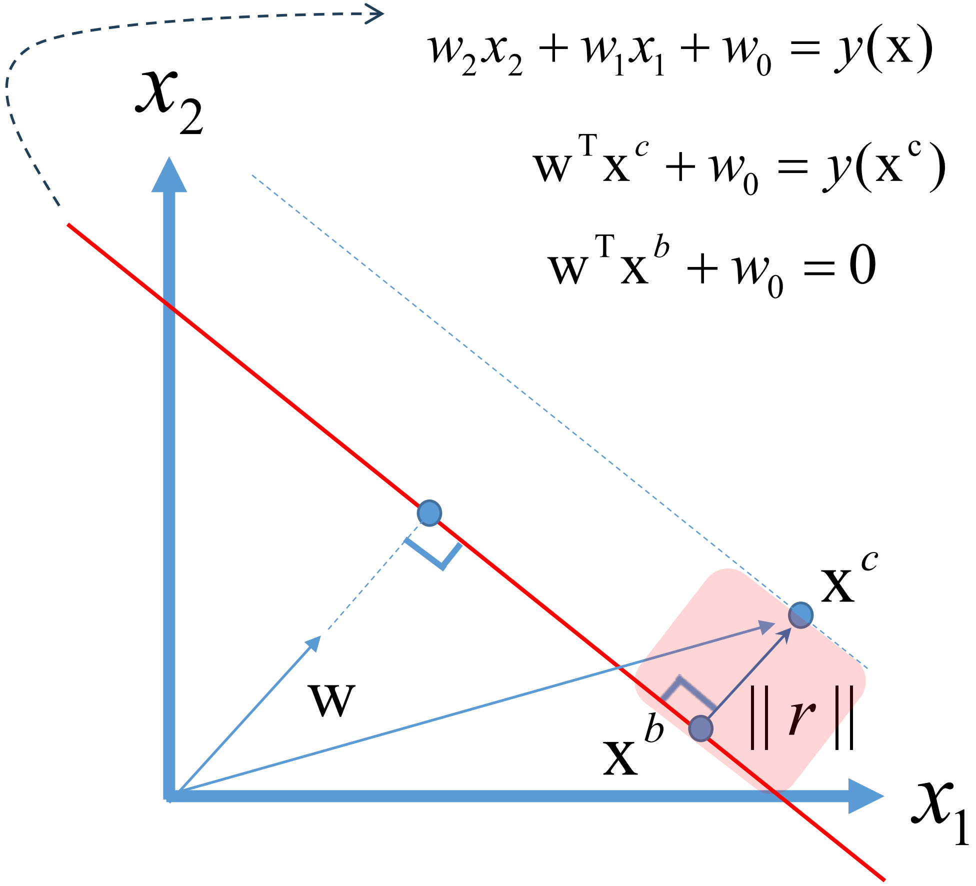
Margin distance

Size of the vector

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

Margin

Unit vector showing the direction only



Margin distance

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

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

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

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

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

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

Let's say

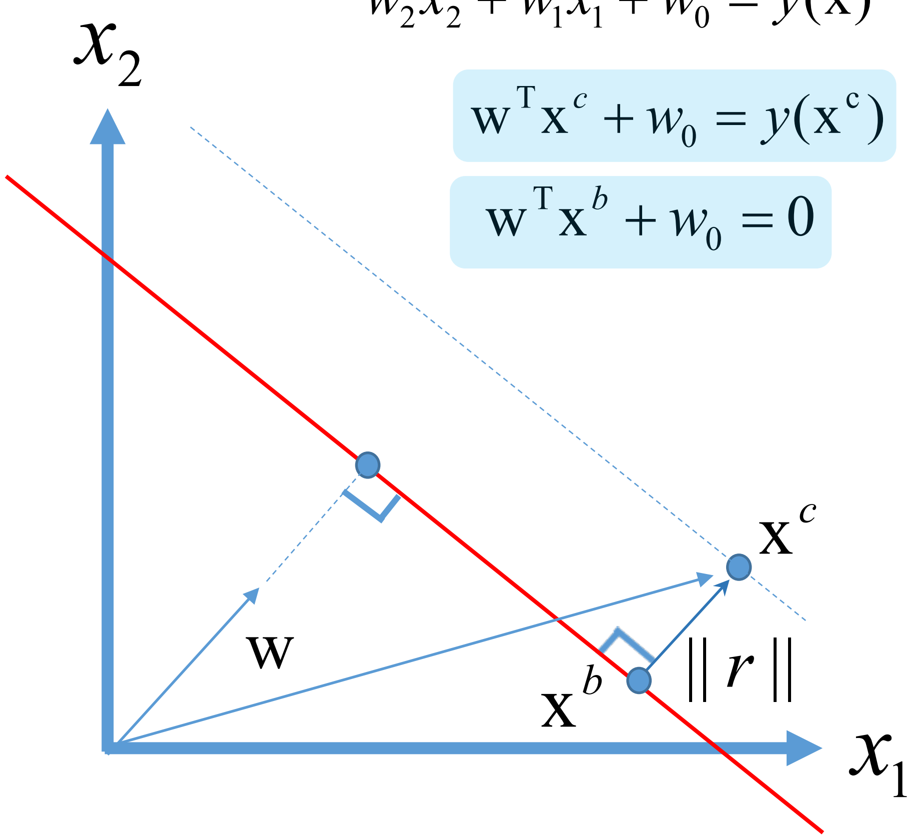
$$|\,\mathrm{y}(\mathrm{x}^c)\,| = 1$$

We use it later …

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

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

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



Problem formulation

- Finding a decision boundary which maximizes the margin.

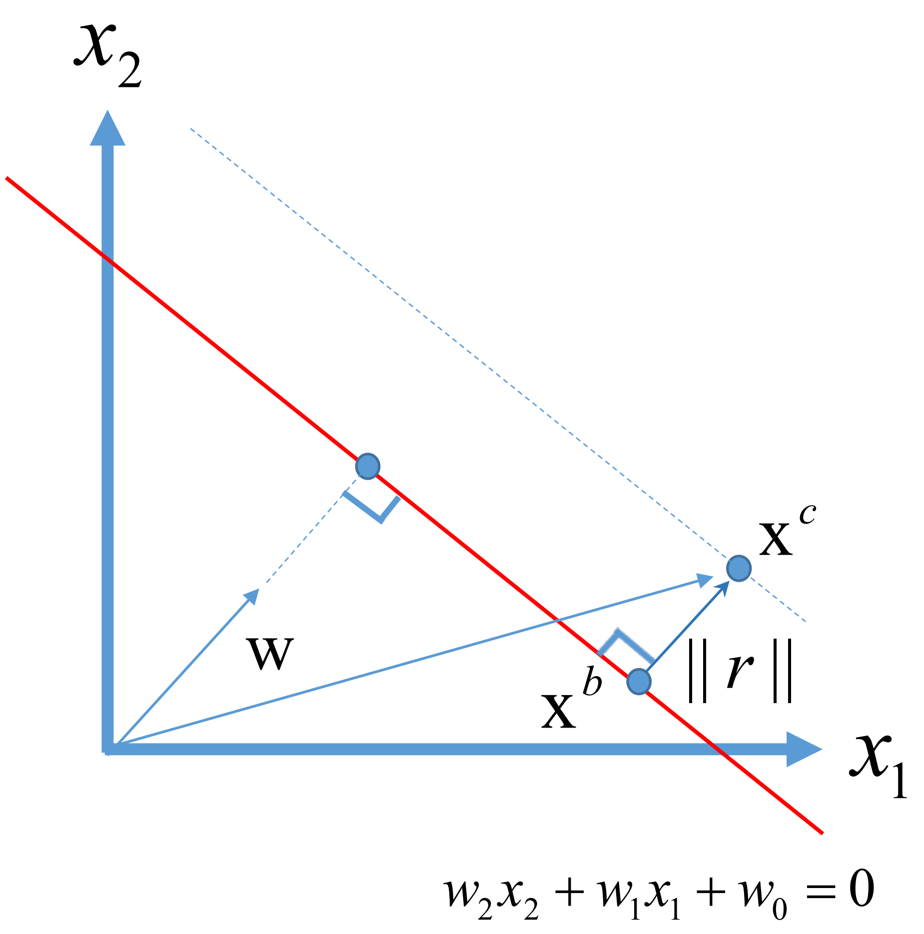
$$\max \| r \| = \frac{1}{\| \mathrm{w} \|}$$

s.t.

$$t_n y(\mathrm{x}_n) > 0$$

Every data points are classified correctly.

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



Problem formulation

- Let's modify the optimization problem a bit.

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

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

- Do you remember?

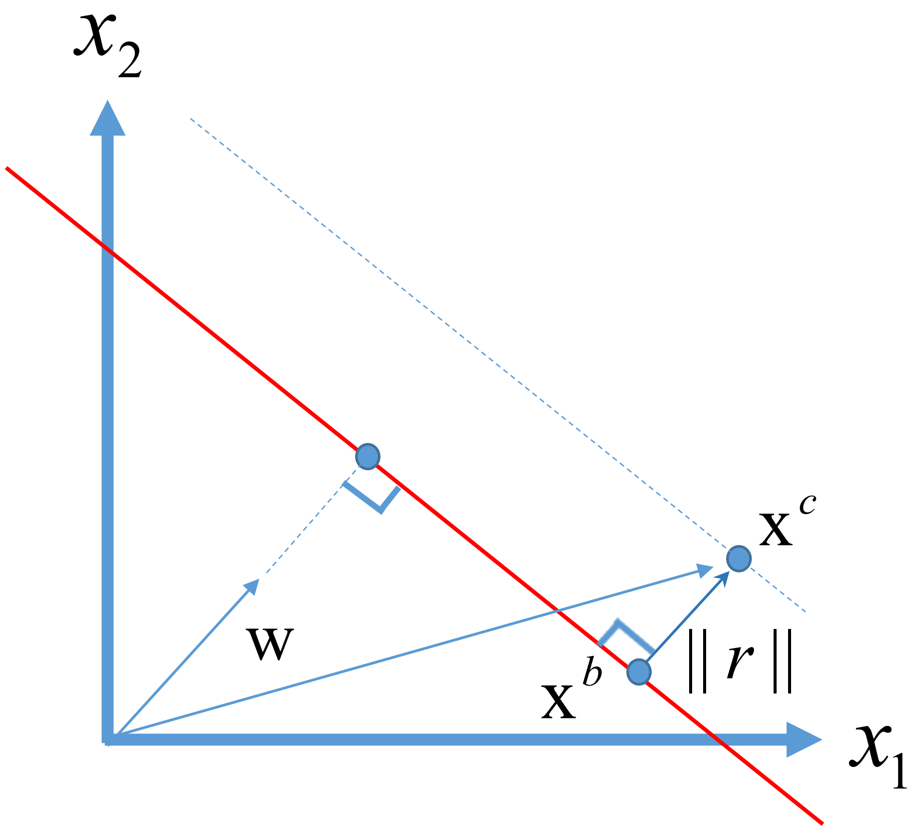
$$\max \frac{1}{\|\mathrm{w}\|}$$

$$s.t. \quad t_n \mathrm{y}(\mathrm{x}_n) \geq 1, \quad \forall n$$

Let's say

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

meaning that any data point is away from the decision boundary at least 1



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

- Finally

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

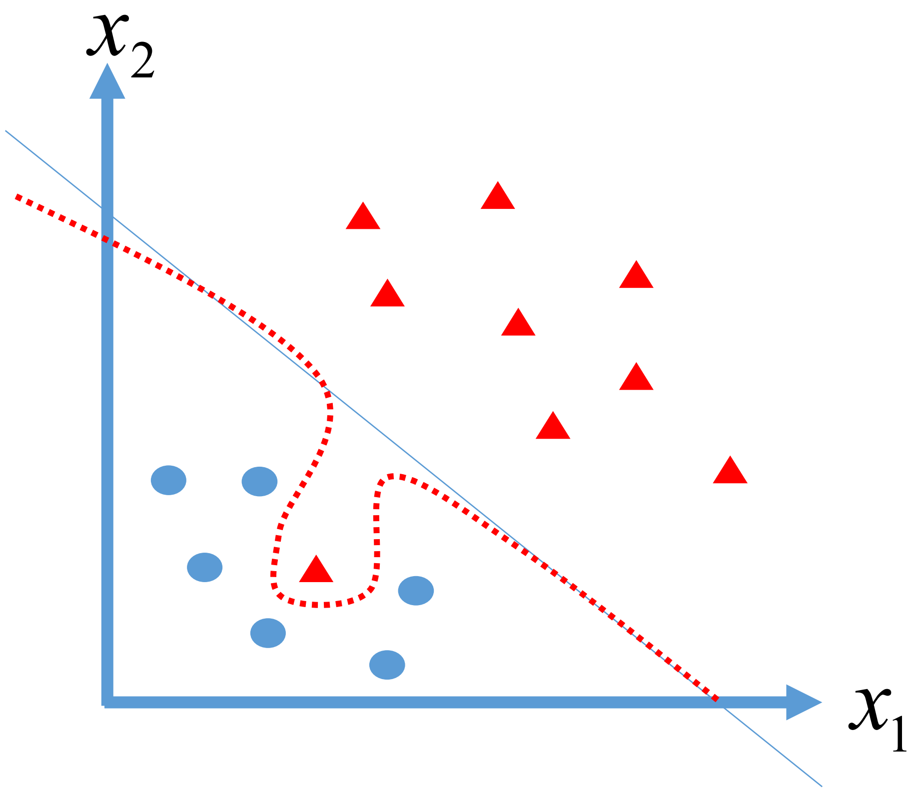
$$s.t. \quad t_n(\mathrm{w}^{\mathrm{T}}\mathrm{x}_n + w_0) \geq 1, \quad \forall n$$

<u>Quadratic programming</u>

How about non-linearly separable case?

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

$$s.t. \quad t(\mathrm{x}_n)(\mathrm{W}\cdot\mathrm{x}_n + w_0) \geq 1, \forall n$$



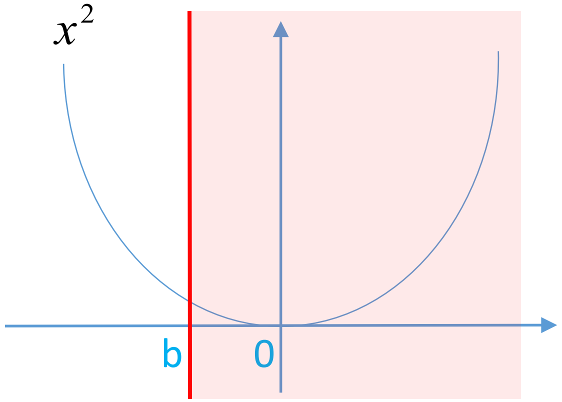
Kernel Trick

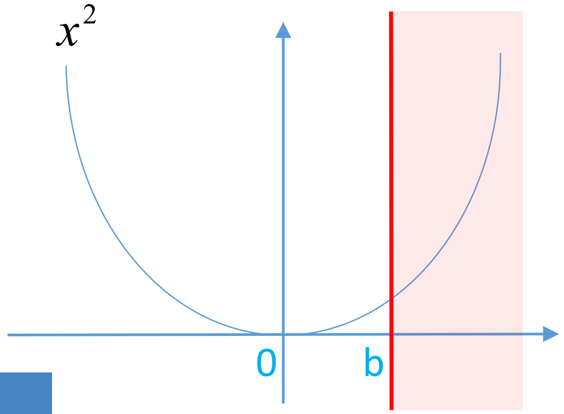
Lagrange method for an optimization problem with inequality constraints

$$\min x^2 \quad s.t. \quad x \geq b$$

? =

$$\min_{x} \max_{\lambda} x^2 - \lambda(x-b) \quad s.t. \quad \lambda \geq 0$$





- If $b \leq 0$, the minima is 0 … so $\lambda=0$
- If $b > 0$, the minima is b^2 … so $x=b$
- So, either λ or $(x-b)$ becomes zero, in other words,
  - $\lambda(x-b) = 0$ (complementary slackness)
- Since $x \geq b$, maximizing λ minimizes the objective value
  - $\lambda \geq 0$

Convert the quadratic problem in SVM to Lagrange optimization problem

$$\min \frac{1}{2}\mathrm{w}^T\mathrm{w}$$
$$s.t. \quad t_n(\mathrm{w}^{\mathrm{T}}x_n + w_0) \geq 1$$

$$\min_{\mathrm{w}} \max_{\lambda} \frac{1}{2}\mathrm{w}^T\mathrm{w} - \sum_{n=1}^{n} \lambda_n (t_n(\mathrm{w}^{\mathrm{T}}x_n + w_0) - 1)$$
$$s.t. \quad \lambda_n \geq 0$$

Proof begins

Proof begins

Convert the quadratic problem in SVM to Lagrange optimization problem

$$\min \frac{1}{2}\mathrm{w}^T\mathrm{w}$$
$$s.t. \quad t_n(\mathrm{w}^{\mathrm{T}}x_n + w_0) \geq 1$$

$$\min_{\mathrm{w}} \max_{\lambda} \frac{1}{2}\mathrm{w}^T\mathrm{w} - \sum_{n=1}^{n} \lambda_n (t_n(\mathrm{w}^{\mathrm{T}}x_n + w_0) - 1)$$
$$s.t. \quad \lambda_n \geq 0$$

- We would like to convert again the optimization problem above into another form, which provides same results.
  - Because we want to solve the optimization problem in term of "lagrange multiplier (λ_n)".

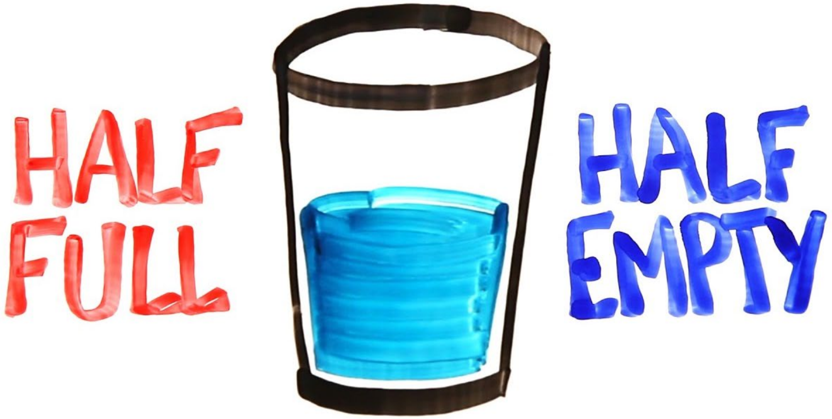
$$\max_{\lambda} \min_{\mathrm{w}} \frac{1}{2}\mathrm{w}^T\mathrm{w} - \sum_{n=1}^{n} \lambda_n (t_n(\mathrm{w}^{\mathrm{T}}x_n + w_0) - 1)$$
$$s.t. \quad \lambda_n \geq 0$$

Convert the quadratic problem in SVM to Lagrange optimization problem

Primal problem

$$\min_{\mathrm{w}} \max_{\lambda} \frac{1}{2}\mathrm{w}^T\mathrm{w} - \sum_{n=1}^{n} \lambda_n (t_n(\mathrm{w}^{\mathrm{T}} x_n + w_0) - 1)$$

$$s.t. \quad \lambda_n \geq 0$$



- We would like to convert again the optimization problem above into another form, which provides same results.
  - Because we want to solve the optimization problem in term of "lagrange multiplier (λ_n)".

Dual problem

$$\max_{\lambda} \min_{\mathrm{w}} \frac{1}{2}\mathrm{w}^T\mathrm{w} - \sum_{n=1}^{n} \lambda_n (t_n(\mathrm{w}^{\mathrm{T}} x_n + w_0) - 1)$$

$$s.t. \quad \lambda_n \geq 0$$

Convert the quadratic problem in SVM to Lagrange optimization problem

Karush–Kuhn–Tucker conditions

KKT conditions

1) Stationarity condition

$$\frac{\partial}{\partial \mathrm{w}}\frac{1}{2}\mathrm{w}^T\mathrm{w}-\frac{\partial}{\partial \mathrm{w}}\sum_{n=1}^{n}\lambda_n(t_n(\mathrm{w}^{\mathrm{T}}x_n+w_0)-1)=0$$

2) Complementary slackness condition

$$\lambda_n(t_n(\mathrm{w}^{\mathrm{T}}x_n+w_0)-1)=0$$

3) Duality feasibility condition

$$\lambda_n \geq 0$$

Primal problem

$$\min_{\mathrm{w}}\max_{\lambda}\frac{1}{2}\mathrm{w}^T\mathrm{w}-\sum_{n=1}^{n}\lambda_n(t_n(\mathrm{w}^{\mathrm{T}}x_n+w_0)-1)$$

$$s.t. \quad \lambda_n \geq 0$$

- We would like to convert again the optimization problem above into another form, which provides same results.
  - Because we want to solve the optimization problem in term of "lagrange multiplier (λ_n)".

Dual problem

$$\max_{\lambda}\min_{\mathrm{w}}\frac{1}{2}\mathrm{w}^T\mathrm{w}-\sum_{n=1}^{n}\lambda_n(t_n(\mathrm{w}^{\mathrm{T}}x_n+w_0)-1)$$

$$s.t. \quad \lambda_n \geq 0$$

Dual problem of the quadratic problem: applying stationarity condition

$$\max_{\lambda} \min_{w,w_0} \quad L(\mathrm{w}, \mathrm{w}_0, \lambda) = \frac{1}{2}\mathrm{w}^{\mathrm{T}}\mathrm{w} \; - \sum_{n=1}^{N} \lambda_n \left(t_n (\mathrm{w}^{\mathrm{T}} x_n + w_0) - 1 \right)$$

$$\boxed{\mathrm{w} = \sum_{n=1}^{N} \lambda_n t_n x_n}$$

$$\frac{\partial L}{\partial w} = \mathrm{w} - \sum_{n=1}^{N} \lambda_n t_n x_n = 0$$

- The first one is called stationarity condition.
  - when we partial differentiate the problem with respect to its parameter "w", each of them should be zero.

Dual problem of the quadratic problem: applying stationarity condition

$$\max_{\lambda} \min_{w, w_0} \quad L(\mathrm{w}, \mathrm{w}_0, \lambda) = \frac{1}{2}\mathrm{w}^{\mathrm{T}}\mathrm{w} \; - \sum_{n=1}^{N} \lambda_n \left(t_n (\mathrm{w}^{\mathrm{T}} x_n + w_0) - 1 \right)$$

$$\boxed{\mathrm{w} = \sum_{n=1}^{N} \lambda_n t_n x_n} \qquad \boxed{\sum_{n=1}^{N} \lambda_n t_n = 0}$$

$$\frac{\partial L}{\partial w} = \mathrm{w} - \sum_{n=1}^{N} \lambda_n t_n x_n = 0 \qquad \frac{\partial L}{\partial w_0} = -\sum_{n=1}^{N} \lambda_n t_n = 0$$

- The first one is called stationarity condition.
  - Again, this time in terms of "w_0"

Dual problem of the quadratic problem: applying stationarity condition

$$\max_{\lambda} \min_{w, w_0} \quad L(\mathrm{w}, \mathrm{w}_0, \lambda) = \frac{1}{2}\mathrm{w}^{\mathrm{T}}\mathrm{w} \; - \sum_{n=1}^{N} \lambda_n \left(t_n (\mathrm{w}^{\mathrm{T}} x_n + w_0) - 1 \right)$$

$$\mathrm{w} = \sum_{n=1}^{N} \lambda_n t_n x_n$$

$$\sum_{n=1}^{N} \lambda_n t_n = 0$$

$$L(\lambda) = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} t_n t_m \lambda_n \lambda_m \mathrm{x}_n^T \mathrm{x}_m - \sum_{n=1}^{N}\sum_{m=1}^{N} t_n t_m \lambda_n \lambda_m \mathrm{x}_n^T \mathrm{x}_m - \sum_{n=1}^{N} \lambda_n t_n w_0 + \sum_{n=1}^{N} \lambda_n$$

Dual problem of the quadratic problem: applying stationarity condition

$$\max_{\lambda} \min_{w,w_0} \quad L(\mathrm{w}, \mathrm{w}_0, \lambda) = \frac{1}{2}\mathrm{w}^{\mathrm{T}}\mathrm{w} - \sum_{n=1}^{N} \lambda_n \left(t_n (\mathrm{w}^{\mathrm{T}} x_n + w_0) - 1 \right)$$

$$\mathrm{w} = \sum_{n=1}^{N} \lambda_n t_n x_n$$

$$\sum_{n=1}^{N} \lambda_n t_n = 0$$

$$L(\lambda) = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} t_n t_m \lambda_n \lambda_m \mathrm{x}_n^T \mathrm{x}_m - \sum_{n=1}^{N}\sum_{m=1}^{N} t_n t_m \lambda_n \lambda_m \mathrm{x}_n^T \mathrm{x}_m - \boxed{\cancel{\sum_{n=1}^{N} \lambda_n t_n w_0}} + \sum_{n=1}^{N} \lambda_n$$

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

Dual problem of the quadratic problem: applying stationarity condition

$$\max_{\lambda} \min_{w,w_0} L(\mathrm{w}, \mathrm{w}_0, \lambda) = \frac{1}{2}\mathrm{w}^{\mathrm{T}}\mathrm{w} - \sum_{n=1}^{N} \lambda_n \left(t_n (\mathrm{w}^{\mathrm{T}} x_n + w_0) - 1 \right)$$

$$\boxed{\mathrm{w} = \sum_{n=1}^{N} \lambda_n t_n x_n}$$

$$\boxed{\sum_{n=1}^{N} \lambda_n t_n = 0}$$

$$L(\lambda) = \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} t_n t_m \lambda_n \lambda_m \mathrm{x}_n^T \mathrm{x}_m - \sum_{n=1}^{N} \sum_{m=1}^{N} t_n t_m \lambda_n \lambda_m \mathrm{x}_n^T \mathrm{x}_m - \boxed{\sum_{n=1}^{N} \cancel{\lambda_n t_n} w_0} + \sum_{n=1}^{N} \lambda_n$$

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

$$\boxed{\lambda_n \geq 0} \quad \boxed{\sum_{n=1}^{N} \lambda_n t_n = 0} \quad \boxed{\mathrm{w} = \cancel{\sum_{n=1}^{N}} \lambda_n t_n x_n}$$

"w" does not appear in the equation, and so we do not use this constraint anymore

Dual problem of the quadratic problem: applying stationarity condition

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

$$s.t. \quad \lambda_n \geq 0, \quad \sum_{n=1}^{N} \lambda_n t_n = 0$$

- ❑ Let’s change it to a quadratic programming again.
- ❑ As mentioned previously, a quadratic programming problem needs to be minimized

Dual problem of the quadratic problem: applying stationarity condition

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

$$s.t. \quad \lambda_n \geq 0, \quad \sum_{n=1}^{N} \lambda_n t_n = 0$$

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

$$s.t. \quad \lambda_n \geq 0, \quad \sum_{n=1}^{N} \lambda_n t_n = 0$$

- Again, the optimization problem becomes a quadratic programming problem.

Let’s summarize

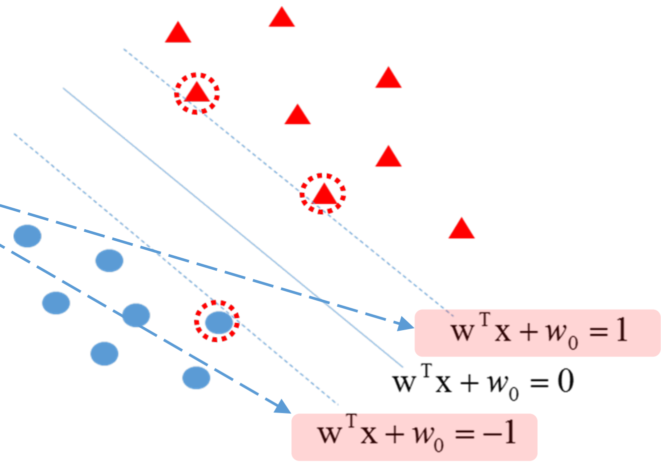
$$\min_{\lambda} \quad L(\lambda) = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} t_n t_m \lambda_n \lambda_m \mathrm{x}_n^T \mathrm{x}_m - \sum_{n=1}^{N} \lambda_n$$

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

- The solution from the quadratic programming is “lagrange multipliers”(λ_n)
- Many of the solutions (lagrange multipliers) are zero
- Complementary slackness (one of KKT conditions) should be satisfied.

$$\lambda_n(t_n(\mathrm{w}^{\mathrm{T}} x_n + w_0) - 1) = 0$$

- In other words, if λ_n are not zero, $(t_n(w_t x_n + w_0)-1)$ should be zero where corresponding data points should be support vectors.



Let's summarize

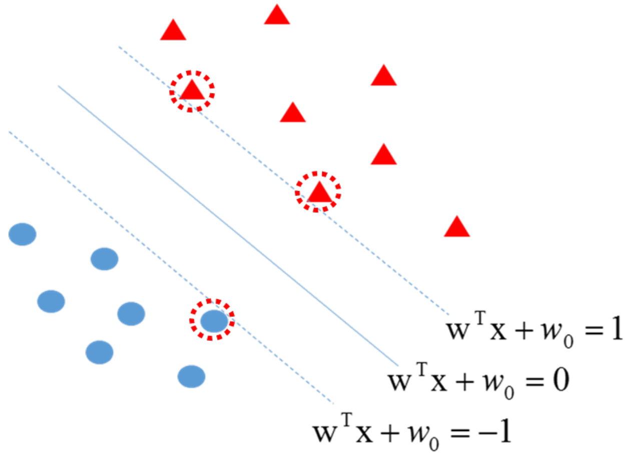
$$\min_{\lambda} \quad L(\lambda) = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} t_n t_m \lambda_n \lambda_m \mathrm{x}_n^T \mathrm{x}_m - \sum_{n=1}^{N} \lambda_n$$

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

- The solution from the quadratic programming is "lagrange multipliers"(λ_n)
- Many of the solutions (lagrange multipliers) are zero
- Complementary slackness (one of KKT conditions) should be satisfied.

$$\lambda_n (t_n (\mathrm{w}^{\mathrm{T}} x_n + w_0) - 1) = 0$$

- In other words, if λ_n are not zero, ($t_n(w_t x_n+w_0)$-1) should be zero where corresponding data points should be support vectors.
- With the non-zero λ_n, w and w_0 can be calculated using $t_n(w_t x_n+w_0)=1$



$$\mathrm{w} = \sum_{n=1}^{N} \lambda_n t_n x_n$$

$$w_0 = t_n - \sum_{n=1}^{N} \lambda_n t_n x_n x_n$$

Let's summarize

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

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

We obtained previously

- $t_n(w_t x_n + w_0) = 1$

$\Rightarrow (w_t x_n + w_0) = t_n$

$\Rightarrow w_0 = t_n - w_t x_n$

$$w_0 = t_n - \sum_{n=1}^{N} \lambda_n t_n x_n x_n$$

- The solution from the quadratic programming is "lagrange multipliers"(λ_n)
- Many of the solutions (lagrange multipliers) are zero
- Complementary slackness (one of KKT conditions) should be satisfied.

$$\lambda_n (t_n (\mathrm{w}^{\mathrm{T}} x_n + w_0) - 1) = 0$$

- In other words, if λ_n are not zero, $(t_n(w_t x_n + w_0) - 1)$ should be zero where corresponding data points should be support vectors.
- With the non-zero λ_n, w and w_0 can be calculated using $t_n(w_t x_n + w_0) = 1$

$$\mathrm{w} = \sum_{n=1}^{N} \lambda_n t_n x_n \qquad w_0 = t_n - \sum_{n=1}^{N} \lambda_n t_n x_n x_n$$

Proof ends

Proof ends

Kernel trick

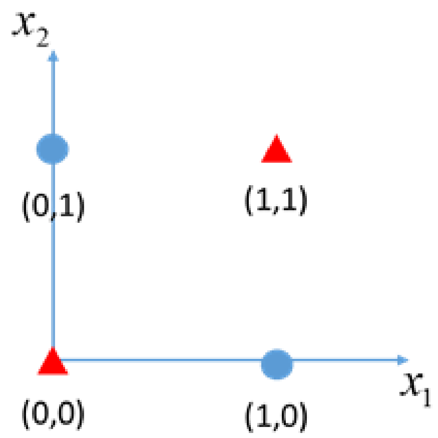
$$\min \frac{1}{2}\mathrm{w}^T\mathrm{w}$$

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

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

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

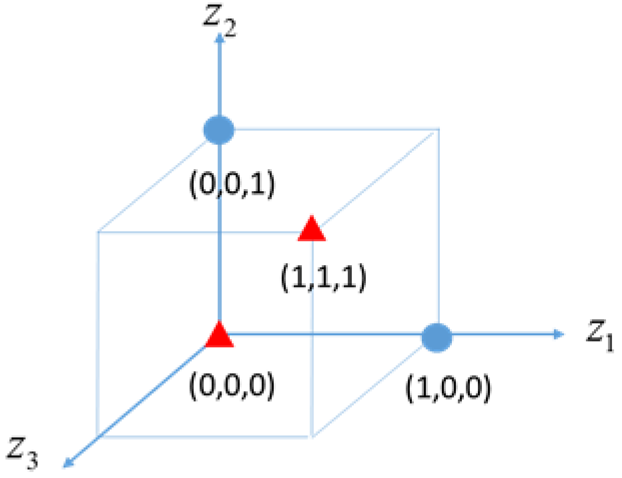
- If data x_n are not linearly separable, what should we do?



Space X

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

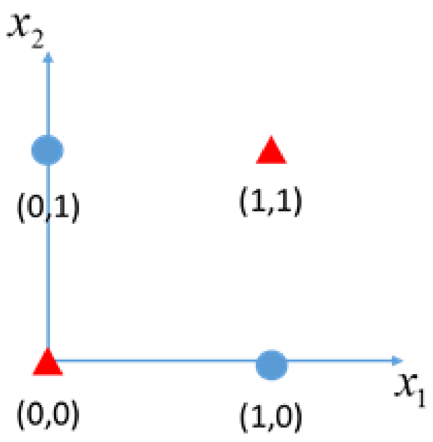
$$\phi(\mathrm{x})$$
$$X \to Z$$



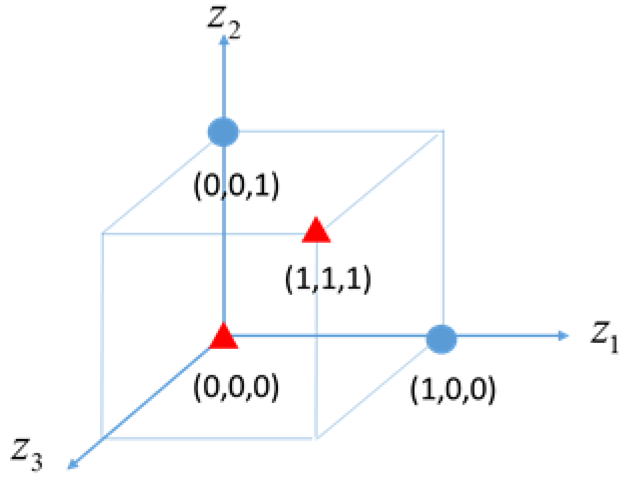
Space Z

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.



Space X



Space Z

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

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

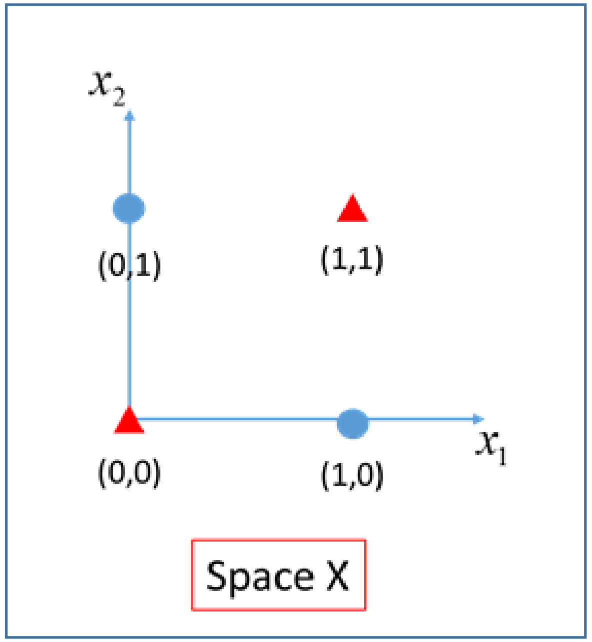
$$\min_{\lambda} \quad L(\lambda) = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} t_n t_m \lambda_n \lambda_m \mathbf{z}_n^T \mathbf{z}_m - \sum_{n=1}^{N} \lambda_n$$

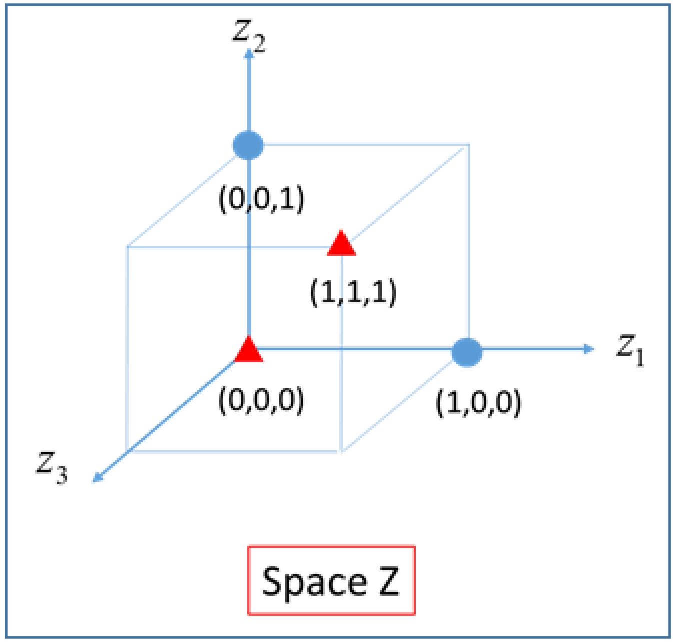
$$s.t. \quad \lambda \geq 0, \qquad t^T \lambda = 0$$

Kernel trick: Kernel function

- 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.

$$K(x_n^T, x_m) = z_n^T z_m$$





Kernel trick: Kernel function

- 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.

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

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

$\Rightarrow$

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

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

- 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.

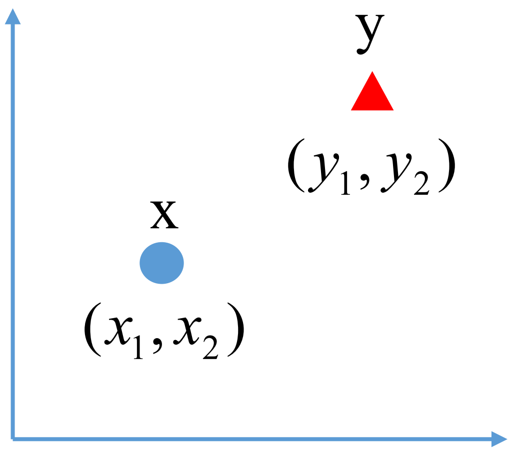
$$\min_{\lambda} \; L(\lambda) = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} t_n t_m \lambda_n \lambda_m \mathrm{z}_n^T \mathrm{z}_m - \sum_{n=1}^{N}\lambda_n$$
$$s.t. \quad \lambda \geq 0, \qquad t^T\lambda = 0$$

$$\Rightarrow$$

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

$$\min_{\lambda} \; L(\lambda) = \frac{1}{2}\lambda^T \begin{bmatrix} t_1 t_1 K(\mathrm{x}_1, \mathrm{x}_1) & t_1 t_2 K(\mathrm{x}_1^T, \mathrm{x}_2) & \cdots & t_1 t_N K(\mathrm{x}_1^T, \mathrm{x}_N) \\ t_2 t_1 K(\mathrm{x}_2, \mathrm{x}_1) & t_2 t_2 K(\mathrm{x}_2^T, \mathrm{x}_2) & \cdots & t_2 t_N K(\mathrm{x}_2^T, \mathrm{x}_N) \\ \cdots & \cdots & \cdots & \cdots \\ t_N t_1 K(\mathrm{x}_N \mathrm{x}_1) & t_N t_2 K(\mathrm{x}_N^T, \mathrm{x}_2) & \cdots & t_N t_N K(\mathrm{x}_N^T, \mathrm{x}_N) \end{bmatrix} \lambda + (-1^T)\lambda$$

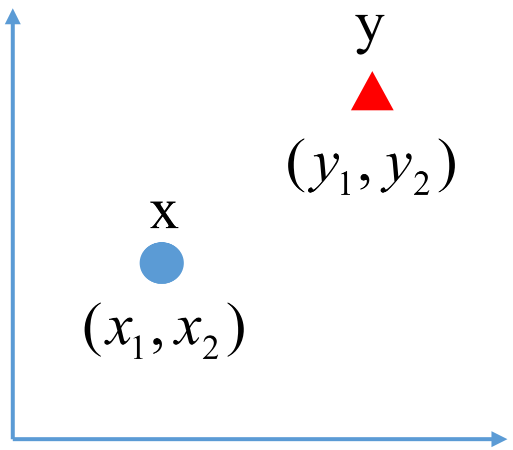
Polynomial kernel of degree 2



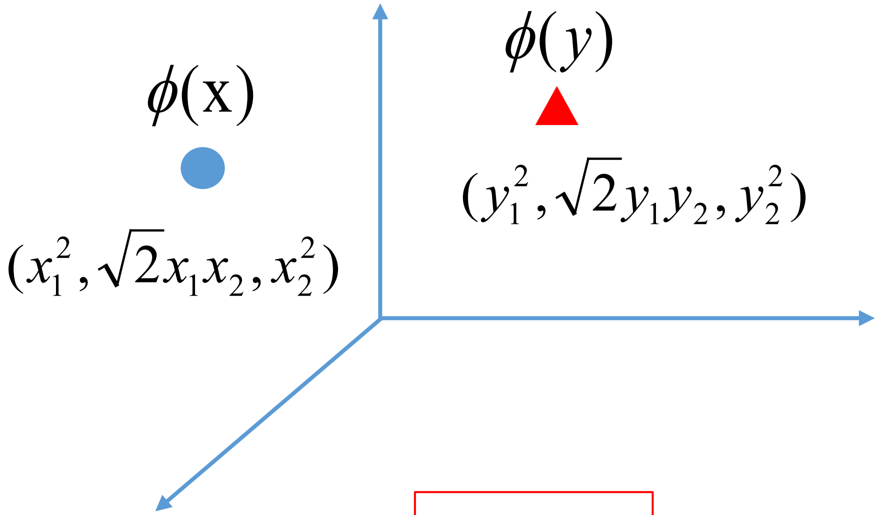
Space X

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

Polynomial kernel of degree 2



Space X



Space Z

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

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

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

Taylor series expansion of
an exponential function

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

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

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

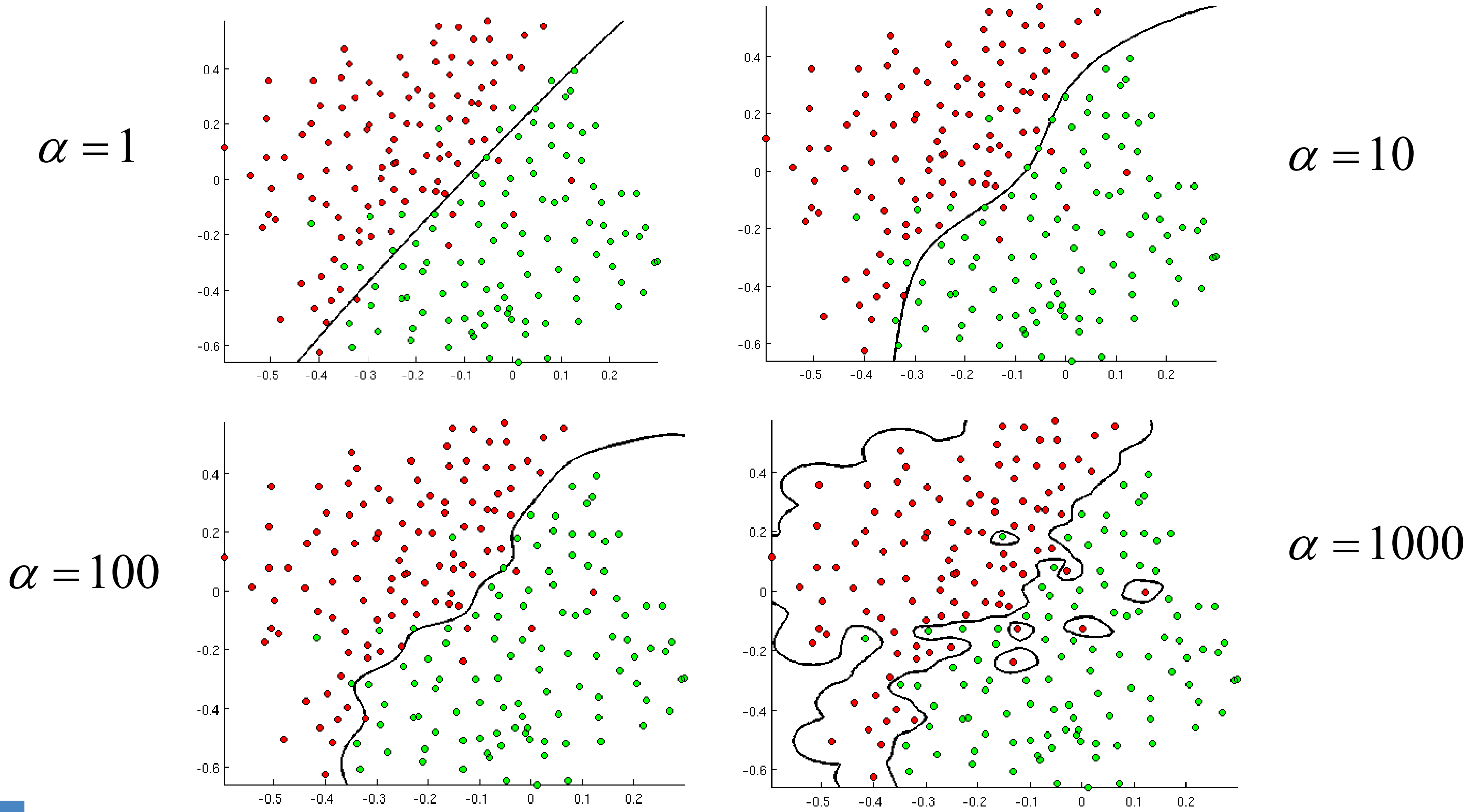
$$= \exp\left(-\alpha \mathrm{x}_n^2\right)\exp\left(-\alpha \mathrm{x}_m^2\right)\sum_{k=0}^{\infty} \frac{(2\alpha)^k (\mathrm{x}_\mathrm{n})^k (\mathrm{x}_\mathrm{m})^k}{\mathrm{k}!}$$

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

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

Mapping to infinite-dimension !

Gaussian Kernel: parameter alpha



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

Summary

- PCA and SVM are probably the most representative conventional machine learning algorithms.
- PCA helps you to manipulate a set of data in a way that
  - determining which features are important,
  - reducing its dimension, so that the data can be processed or visualized more efficiently.
- SVM is a classification method founded on well defined mathematical framework, which can handle linear or nonlinear classification problems.