Perceptron and Neural Networks

CISC3024 – Pattern Recognition

Prof. Yicong Zhou

yicongzhou@um.edu.mo

@Fall 2024

Outline

- Perceptron Algorithm
- XOR Problem
- Two-Layer Perceptron
- Three-Layer Perceptron
- Backpropagation Algorithm
- Conclusion

Perceptron Algorithm (1)

- Problem: consider a two-class task with ω_1 and ω_2
- Decision hyperplane

$$g(x) = w^T x + w_0 = 0$$

$$\Rightarrow \quad w_1 x_1 + w_2 x_2 + \cdots + w_l x_l + w_0 = 0$$

where $w = (w_1, w_2, \ldots, w_l)^T$ is *weight vector*, w_0 is *threshold*.

- Assume x_1 and x_2 on the decision hyperplane

$$w^T x_1 + w_0 = w^T x_2 + w_0 = 0$$

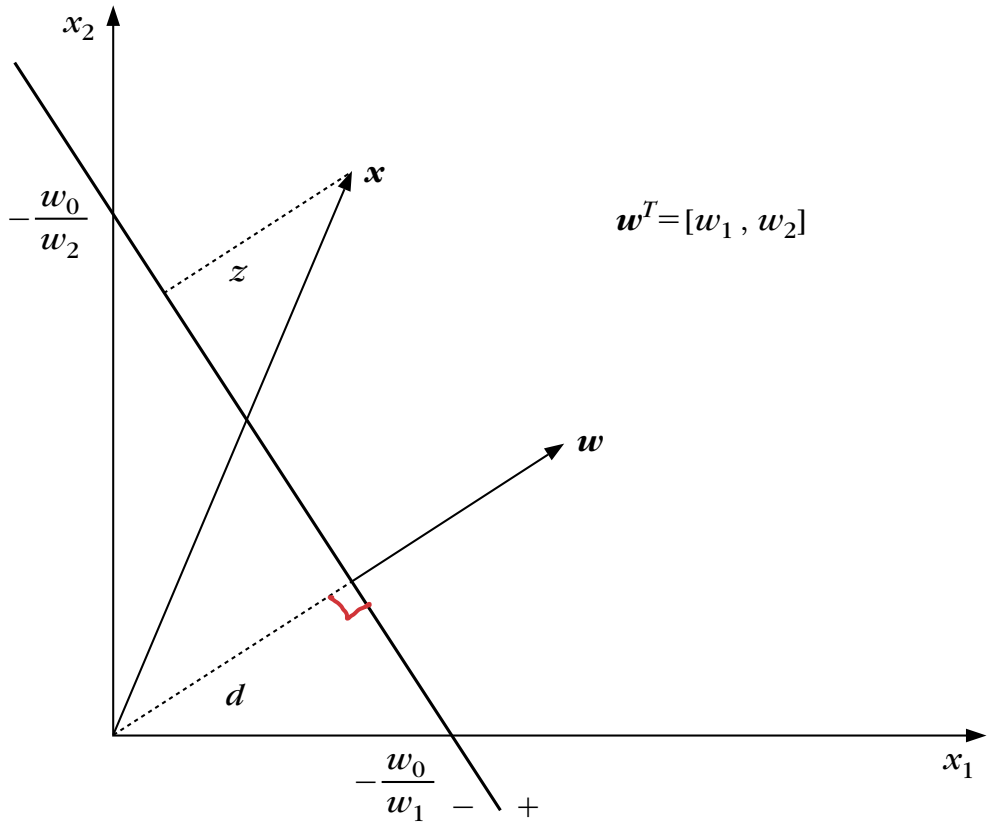
$$\Rightarrow \quad w^T (x_1 - x_2) = 0 \quad \forall x_1, x_2$$

Perceptron Algorithm (2)

- w is orthogonal to the decision hyperplane

$$d = \frac{|w_0|}{\sqrt{w_1^2 + w_2^2}}$$

$$z = \frac{|g(x)|}{\sqrt{w_1^2 + w_2^2}}$$



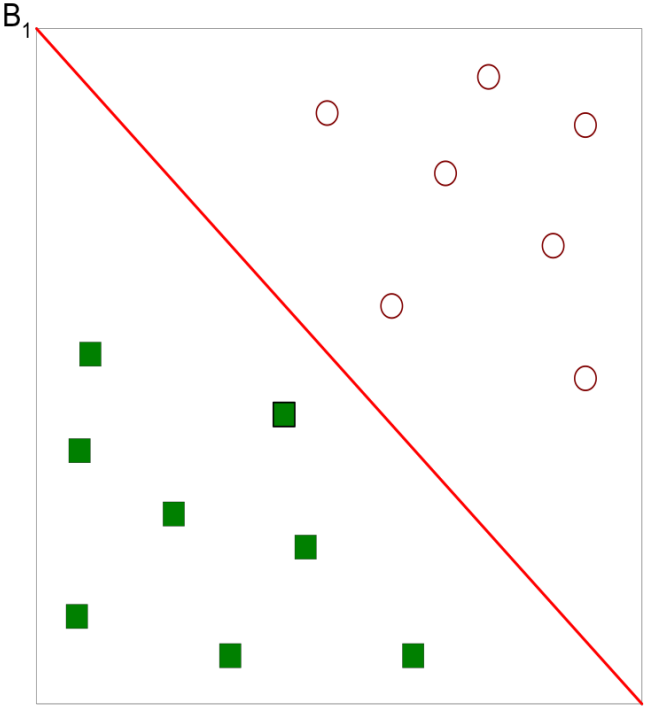
Perceptron Algorithm (3)

- Assume the linearly separable classes, i.e.,

$$\exists\, w^*: \quad w^{*T}x > 0 \quad \forall x \in \omega_1$$

$$w^{*T}x < 0 \quad \forall x \in \omega_2$$

exists such a hyper-plane



- The case $w^{*T}x + w_0^*$ falls under the above formulation, since
  - $w' \equiv (w^*, w_0^*)^T$, and $x' \equiv (x, 1)^T$
  - $w^{*T}x + w_0^* = w'^T x' = 0$

Perceptron Algorithm (4)

- Our goal: compute a solution, a hyperplane w, so that

$$w^T x + w_0 > 0 \ (or < 0), \qquad x \in \omega_1 \ (or \ \omega_2)$$

- Perceptron algorithm:
  - Defines a cost function
  - Chooses an algorithm to minimize the cost function
  - The minimum corresponds to a solution

Perceptron Algorithm (5)

- Perceptron cost function

Weights to be examined.

$$J(w) = \sum_{x \in Y} \delta_x w^T x$$

Incorrectly Classified.

  - Y is the subset of the training vectors wrongly classified by w
  - When $Y = \emptyset$, a solution is achieved and
  - $J(w) \geq 0$
  - $\delta_x = \begin{cases} -1 & \text{If } x \in \omega_1 \\ +1 & \text{If } x \in \omega_2 \end{cases}$

$J(w)$ is continuous and piecewise linear

Perceptron Algorithm (6)

partial derivative of the cost func. over the weights. $\vec{w}$.

- Perceptron Algorithm:

$$\frac{\partial J(w)}{\partial w} = \frac{\partial}{\partial w}\left(\sum_{x \in Y}(\delta_x w^T x)\right) = \sum_{x \in Y} \delta_x x$$

$$w(t+1) = w(t) - \rho_t \frac{\partial J(w)}{\partial w}\bigg|_{w = w(t)} \quad \Rightarrow \quad w(t+1) = w(t) - \rho_t \sum_{x \in Y} \delta_x x$$

- The perceptron algorithm converges in a finite number of iteration steps to a solution if

$$\lim_{t \to \infty} \sum_{k=0}^{t} \rho_k \to \infty \qquad \text{and} \qquad \lim_{t \to \infty} \sum_{k=0}^{t} \rho_k^2 < +\infty$$

Accumulation of LR is infinite.

=> Have a LR large enough at each step

Finite squared sum of LR

=> A LR that decays fast enough.

=> Prevent fluctuation & not converge

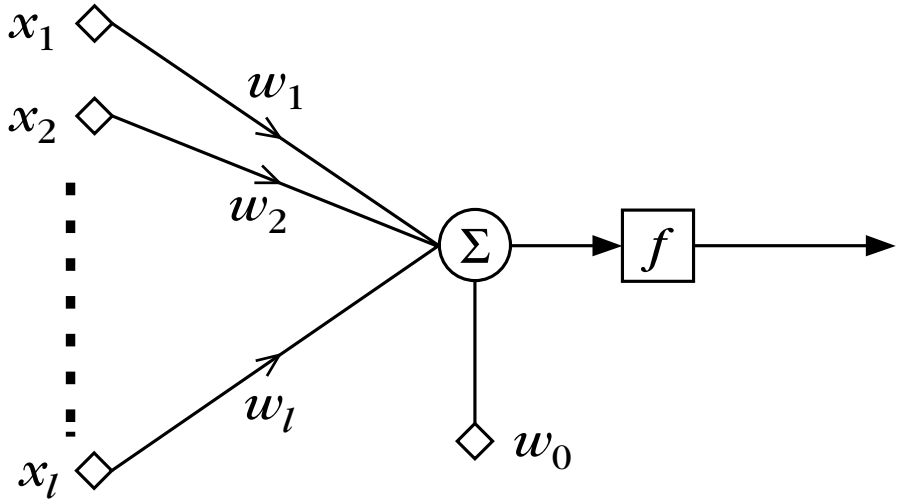
Perceptron Algorithm (7)

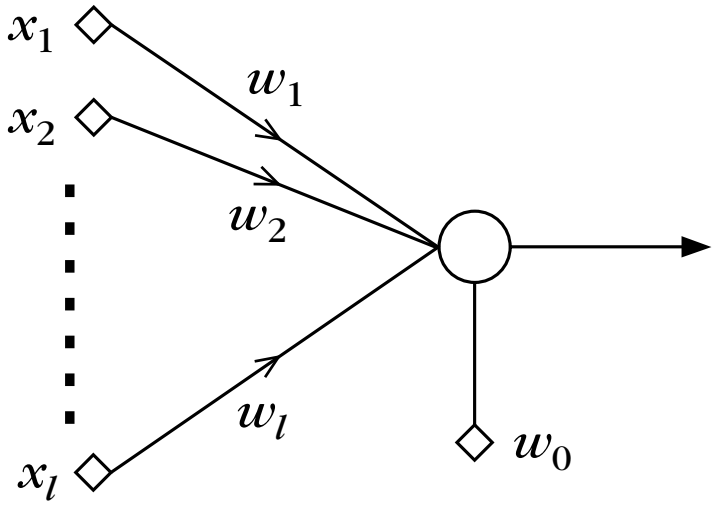
- The perceptron algorithm is a reward and punishment scheme

- Perceptron (or neuron)

If $w^T x + w_0 > 0$, assign $x \in \omega_1$

If $w^T x + w_0 < 0$, assign $x \in \omega_2$





Perceptron Algorithm (8)

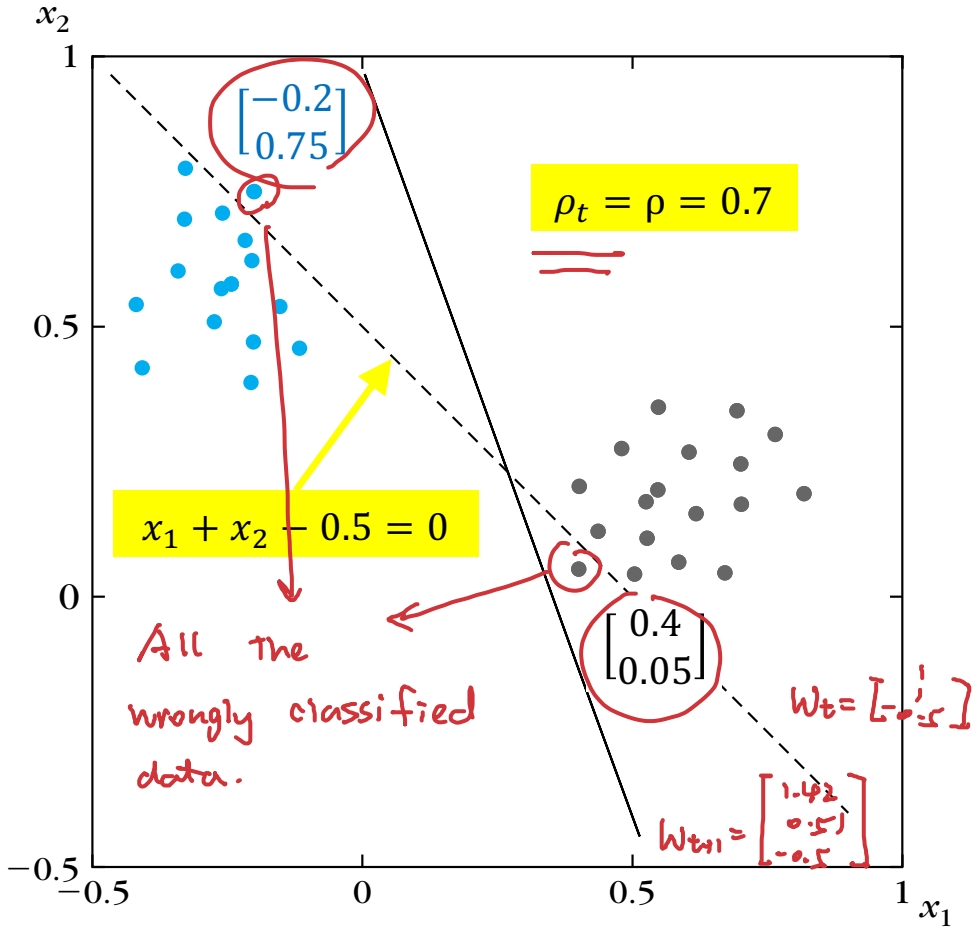
- Example:

$$w = [w_1, w_2, w_0]^T = [1, 1, -0.5]^T$$

The next weight vector will be

$$w(t+1) = \begin{bmatrix} 1 \\ 1 \\ -0.5 \end{bmatrix} - 0.7(-1)\begin{bmatrix} 0.4 \\ 0.05 \\ 1 \end{bmatrix} - 0.7(+1)\begin{bmatrix} -0.2 \\ 0.75 \\ 1 \end{bmatrix}$$

$$\Rightarrow \quad w(t+1) = \begin{bmatrix} 1.42 \\ 0.51 \\ -0.5 \end{bmatrix}$$



$$W_{t+1} = W_t + \rho_t \frac{\partial J(w)}{\partial w}\Big|_{w=w_t} = \begin{bmatrix} 1 \\ 1 \\ -0.5 \end{bmatrix} + 0.7 \times \sum_{i=1}^{t} \delta_x x = \begin{bmatrix} 1 \\ 1 \\ -0.5 \end{bmatrix} + 0.7 \times \left[(-1) \times \begin{bmatrix} 0.4 \\ 0.05 \\ 1 \end{bmatrix} + (+1) \times \begin{bmatrix} -0.2 \\ 0.75 \\ 1 \end{bmatrix} \right]$$

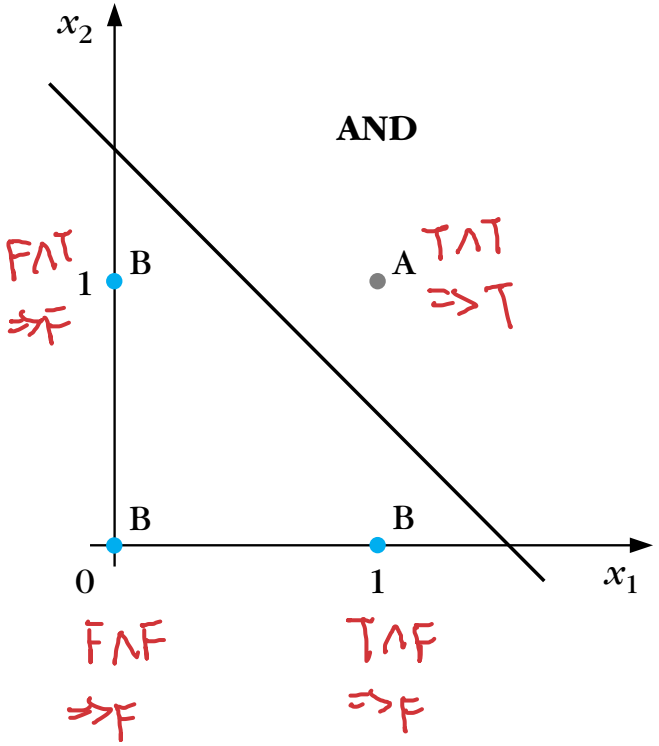
Outline

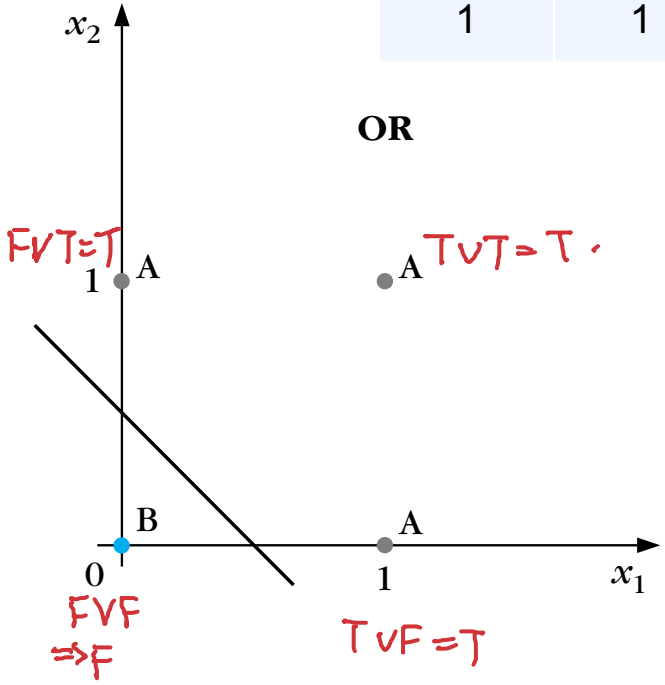
- Perceptron Algorithm
- XOR Problem
- Two-Layer Perceptron
- Three-Layer Perceptron
- Backpropagation Algorithm
- Conclusion

XOR problem (1)

- AND and OR operations are linearly separable

x_1	x_2	AND	Class	OR	Class
0	0	0	B	0	B
0	1	0	B	1	A
1	0	0	B	1	A
1	1	1	A	1	A

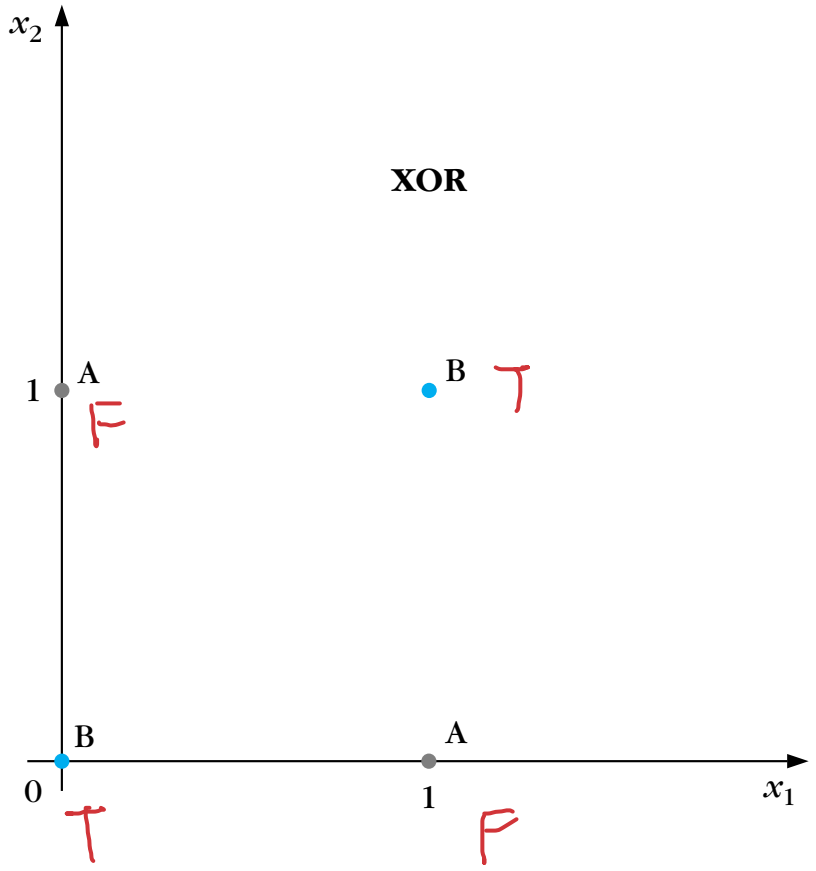




XOR problem (2)

x_1	x_2	XOR	Class
0	0	0	B
0	1	1	A
1	0	1	A
1	1	0	B

A line (hyperplane) cannot separate class A from class B

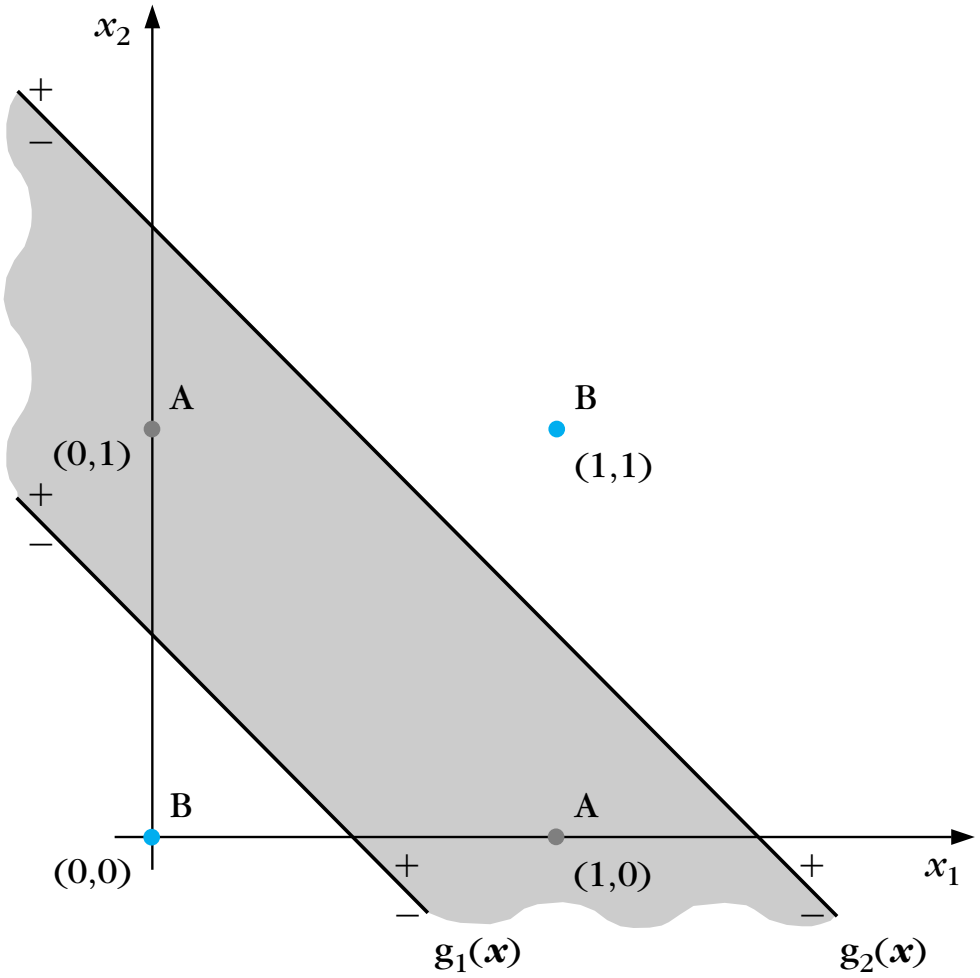


Two-Layer Perceptron (1)

- For XOR problem, draw two-lines

Class B is outside the shaded area and class A inside

This is a two-phase design.



Two-Layer Perceptron (2)

- Phase 1: draw two-lines (hyperplanes)

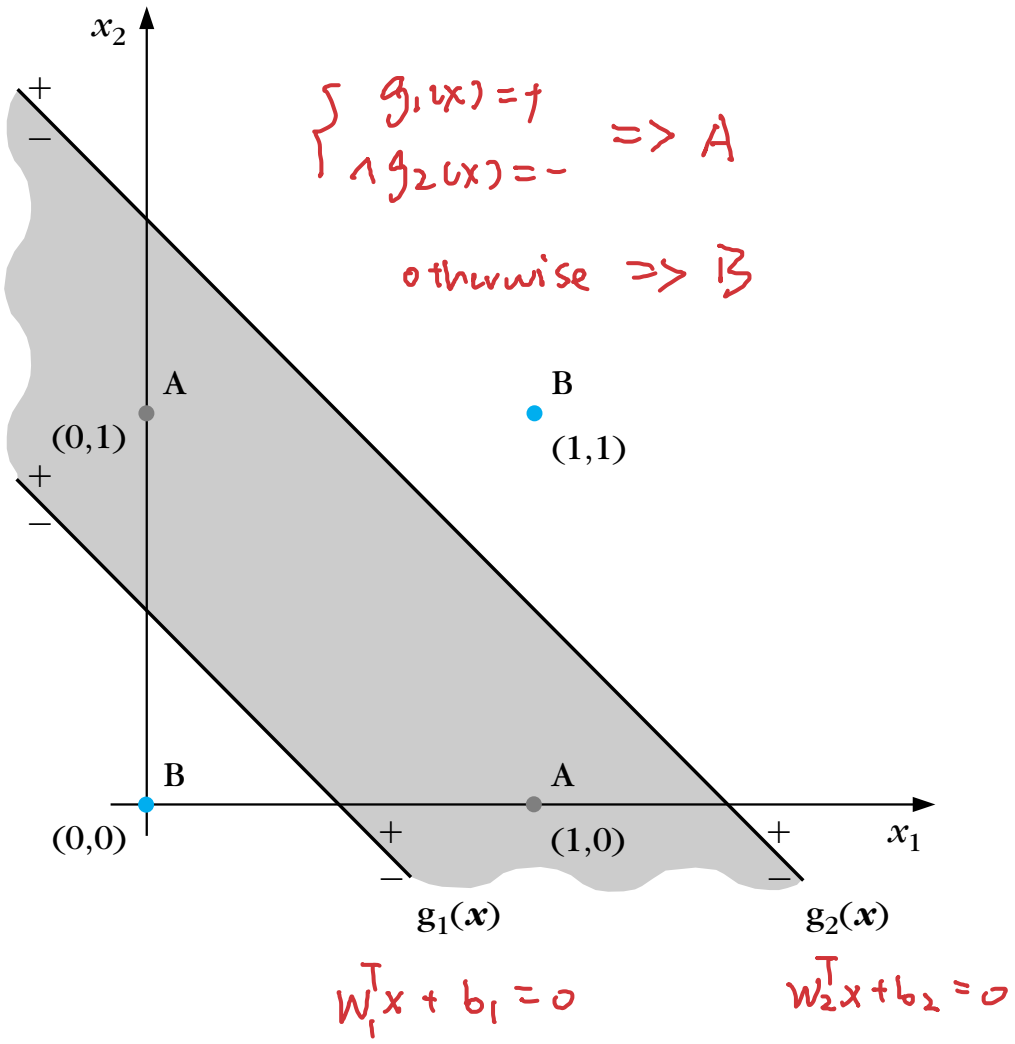
$$g_1(x) = g_2(x) = 0$$

  - Each one is realized by a perceptron
  - Depending on the position of x, the outputs of perceptrons are

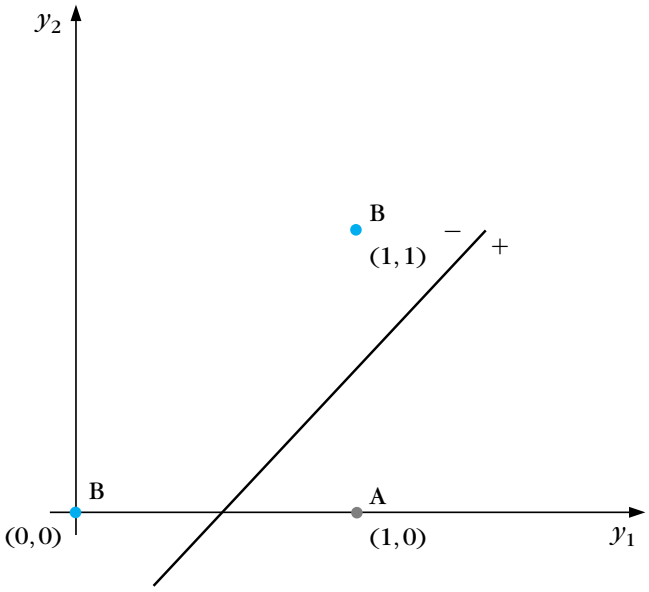
$$y_i = f\big(g_i(x)\big) = \begin{cases} 0 \\ 1 \end{cases}$$

- Phase 2: find the position of x $w.r.t$ both lines

Two-Layer Perceptron (3)



1st phase				2nd phase
x_1	x_2	y_1	y_2	
0	0	0 (-)	0 (-)	B (0)
0	1	1 (+)	0 (-)	A (1)
1	0	1 (+)	0 (-)	A (1)
1	1	1 (+)	1 (+)	B (0)

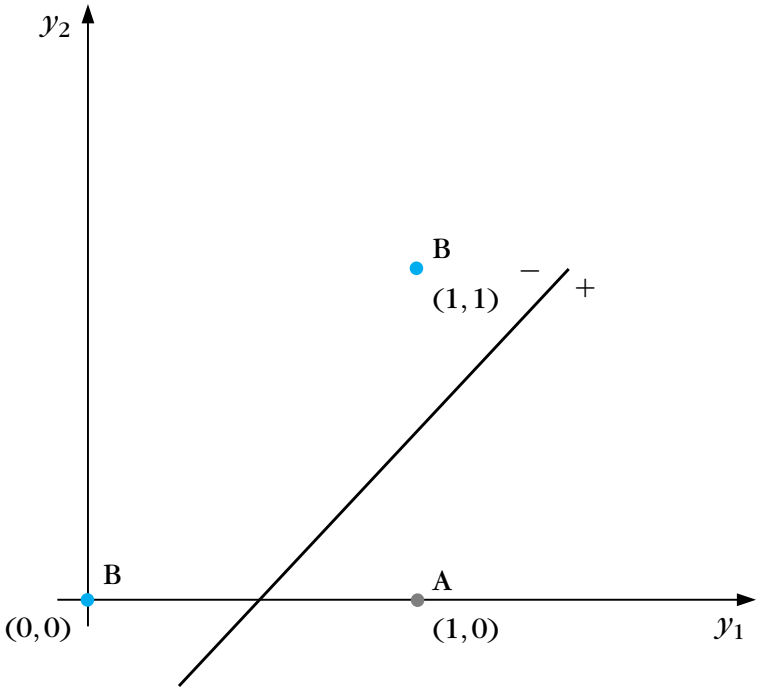


Two-Layer Perceptron (4)

- The computations of the first phase equivalently perform a mapping

$$x \longrightarrow y = [y_1, y_2]^T$$

- The decision is now performed on the transformed y data



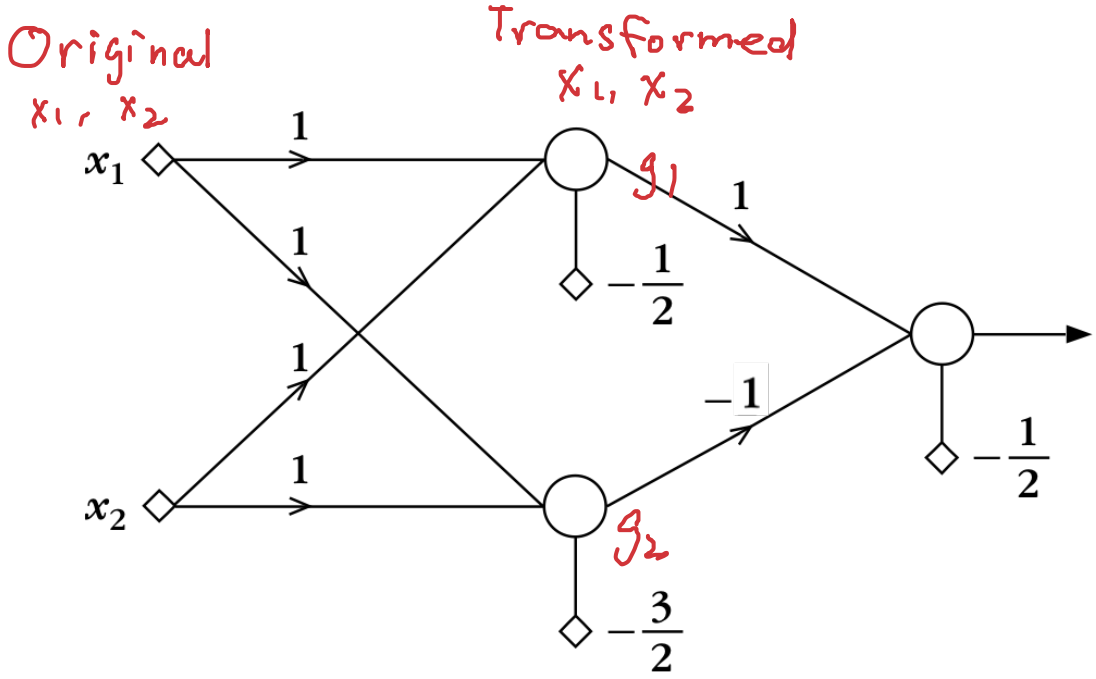
Classification can be performed using a second line.

The second line can be realized by a perceptron.

- Add layer => Do more pre-process on original data.
- Add dimension

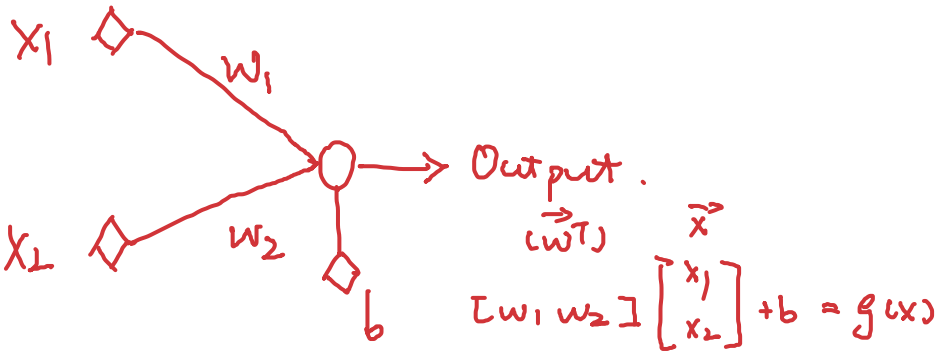
Two-Layer Perceptron (5)

- Computations of the first phase perform a mapping
  - Transforms the nonlinearly separable problem to linearly separable one



- Two-layer perceptron
  - One hidden layer
  - One output layer

Bare Model



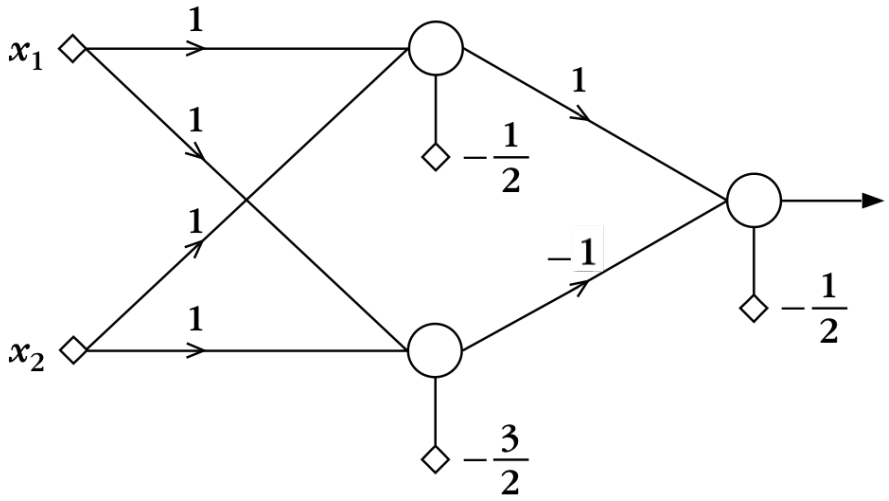
Two-Layer Perceptron (6)

- Neurons (nodes) realize the following lines (hyperplanes)

$$g_1(x) = x_1 + x_2 - \frac{1}{2} = 0$$

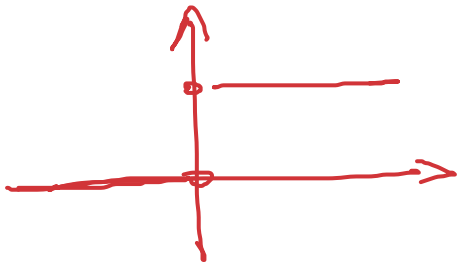
$$g_2(x) = x_1 + x_2 - \frac{3}{2} = 0$$

$$g(y) = y_1 - y_2 - \frac{1}{2} = 0$$



- activation function

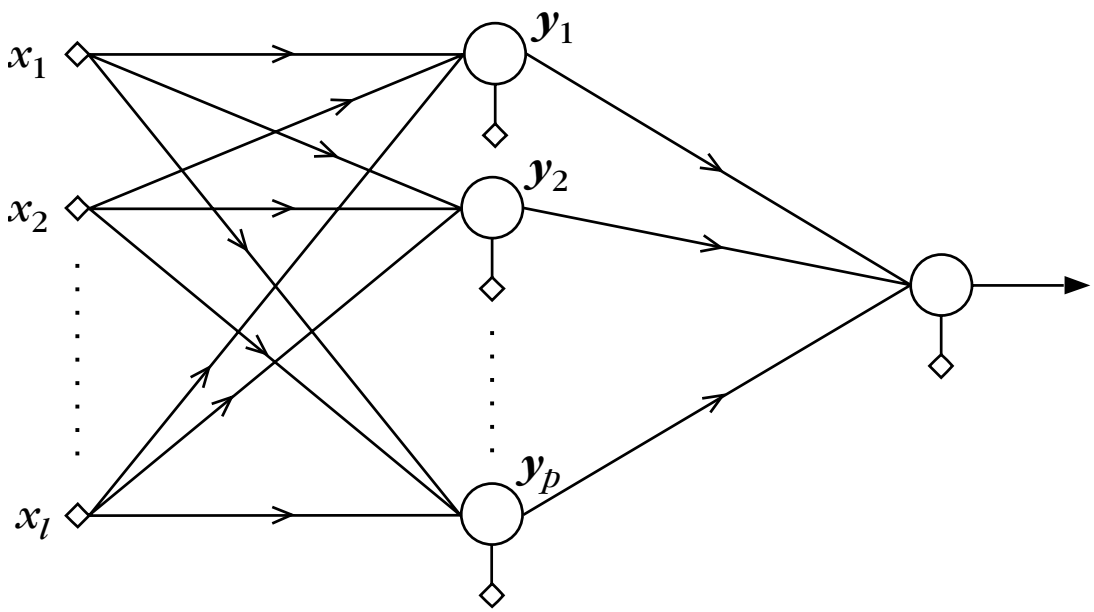
$$f(\cdot) = \begin{cases} 0 \\ 1 \end{cases}$$



Two-Layer Perceptron (7)

- General case: $x \in R^l$

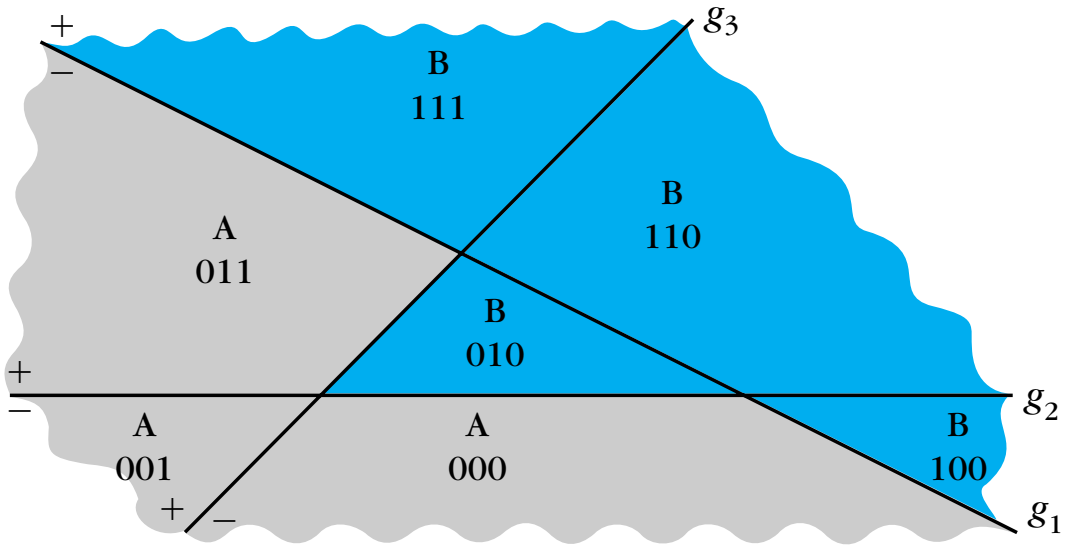
$$x \longrightarrow y = \left[y_1, y_2, \ldots, y_p\right]^T, \quad y_i \in \{0,1\} \quad i = 1, 2, \ldots, p$$

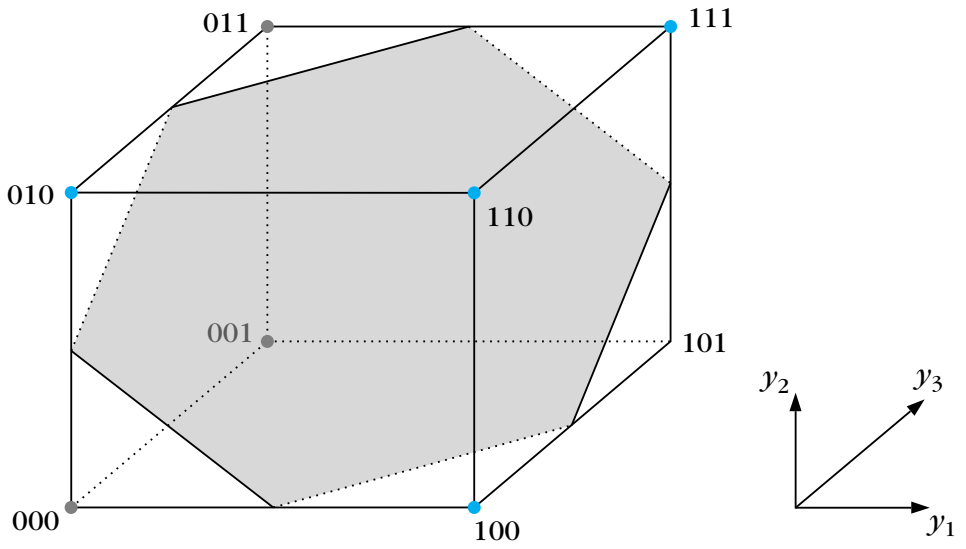


Two-Layer Perceptron (8)

The ability of 2-layer perceptron is still limited.

- Intersections of hyperplanes
  - form regions in the l-dimensional space
  - Each region corresponds to a vertex of the H_p unit hypercube





Two-Layer Perceptron (9)

- Two-layer perceptron
  - The output neuron realizes a hyperplane in the transformed space
  - The hyperplane separates some vertices from the others
  - It has the capability to classify vectors into classes that consist of unions of polyhedral regions
  - However, not any union, It depends on the relative position of the corresponding vertices

Outline

- Perceptron Algorithm
- XOR Problem
- Two-Layer Perceptron
- Three-Layer Perceptron
- Backpropagation Algorithm
- Conclusion

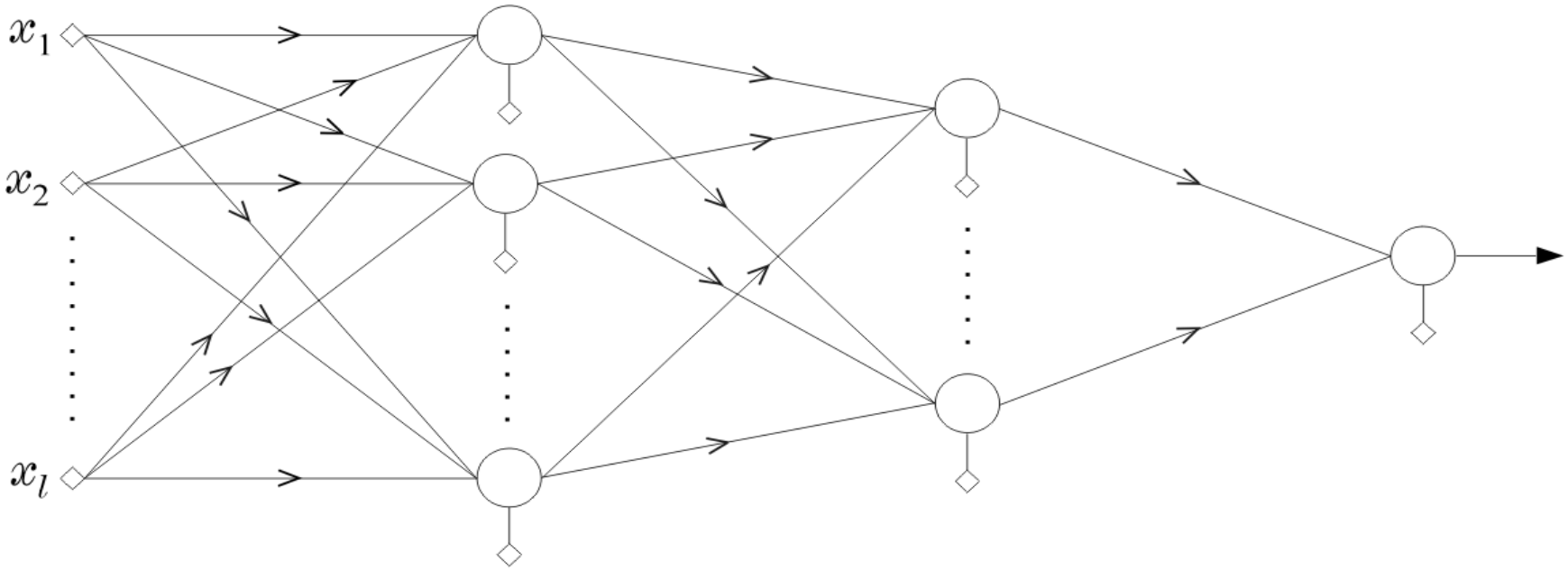
Three-Layer Perceptron (1)

Do projection 2 times.

- Three-layer perception
  - It can classify vectors into classes consisting of ANY union of polyhedral regions
  - The idea is like the XOR problem
  - It realizes more than one planes in the $y \in R^p$ space.

Three-Layer Perceptron (2)

- Structure:



Three-Layer Perceptron (3)

- Three-layer perceptron
  - 1st layer forms the hyperplanes
  - 2nd layer forms the regions
  - output neuron forms the classes
- Output neuron realizes an OR gate

Three-Layer Perceptron (4)

- For each vertex with class A, construct a hyperplane to leaves it on one side (+) and ALL others to the other side (-)

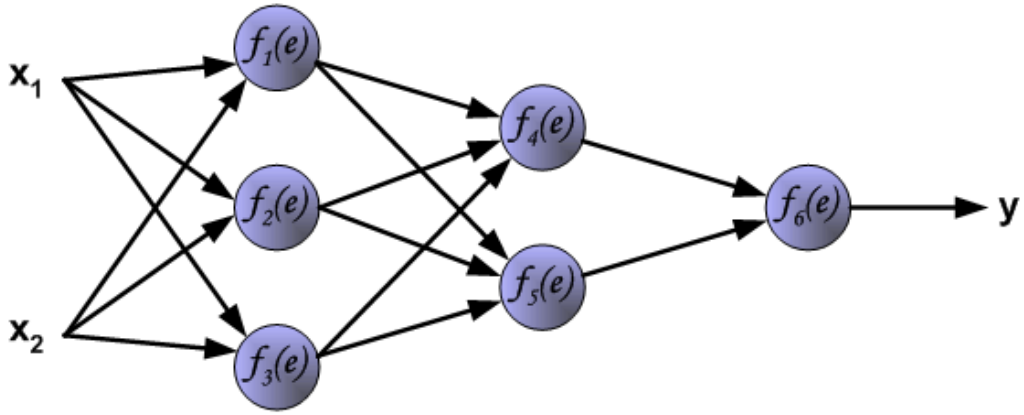
- Ways to design multilayer perceptrons are developing a structure
  - to classify correctly all the training patterns
  - OR to compute the synaptic weights to optimize a cost function

Outline

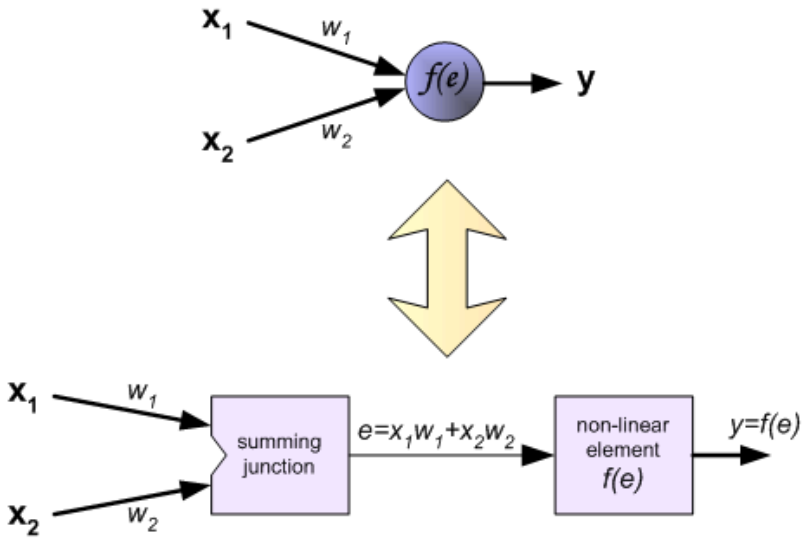
- Perceptron Algorithm
- XOR Problem
- Two-Layer Perceptron
- Three-Layer Perceptron
- Backpropagation Algorithm
- Conclusion

Backpropagation Algorithm (1)

- Three-layer neural network (Training Example)

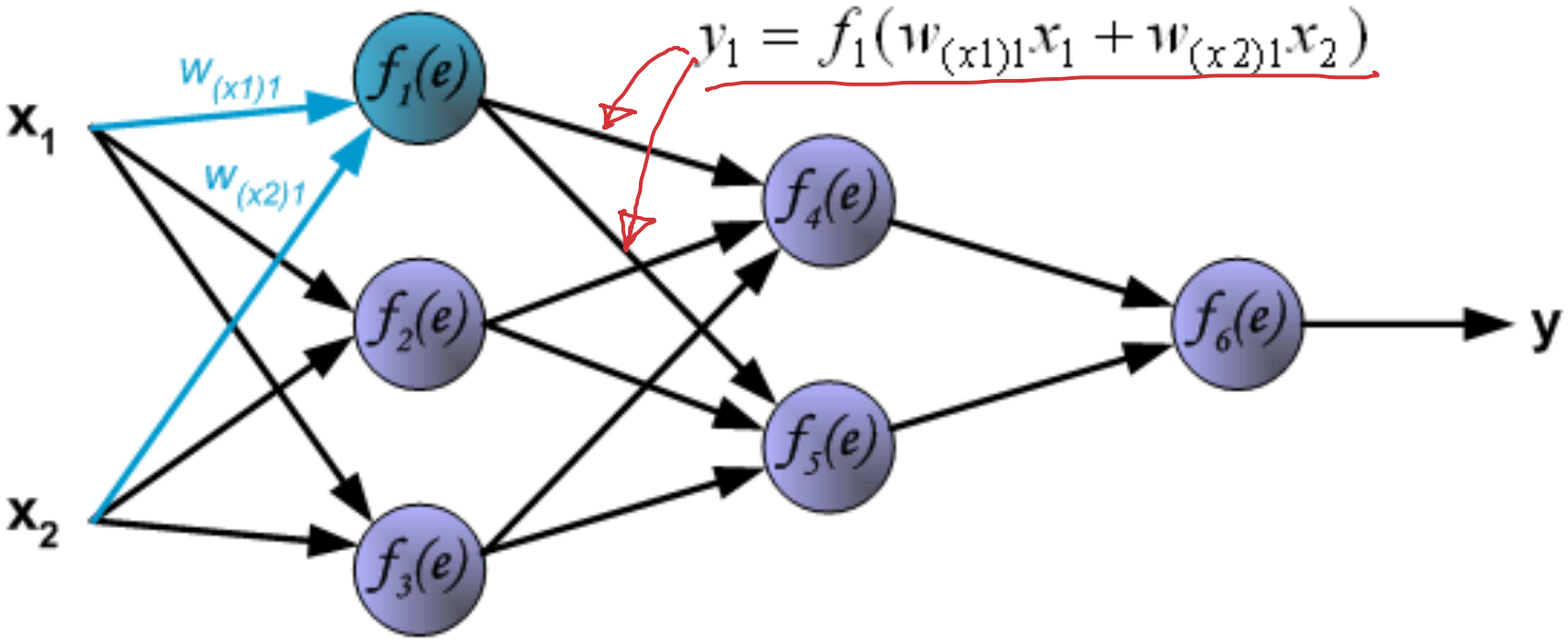


- Each neuron:
  - weighted input signals
  - activation function

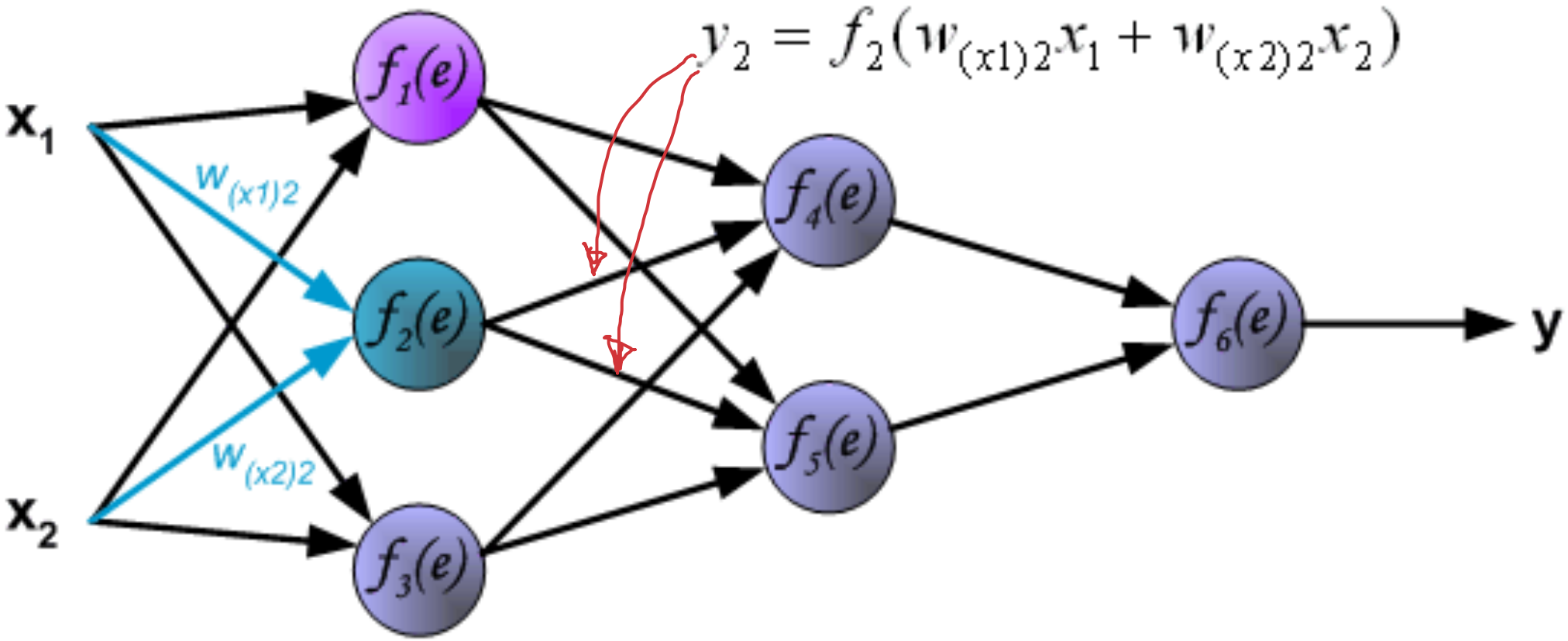


Backpropagation Algorithm (2)

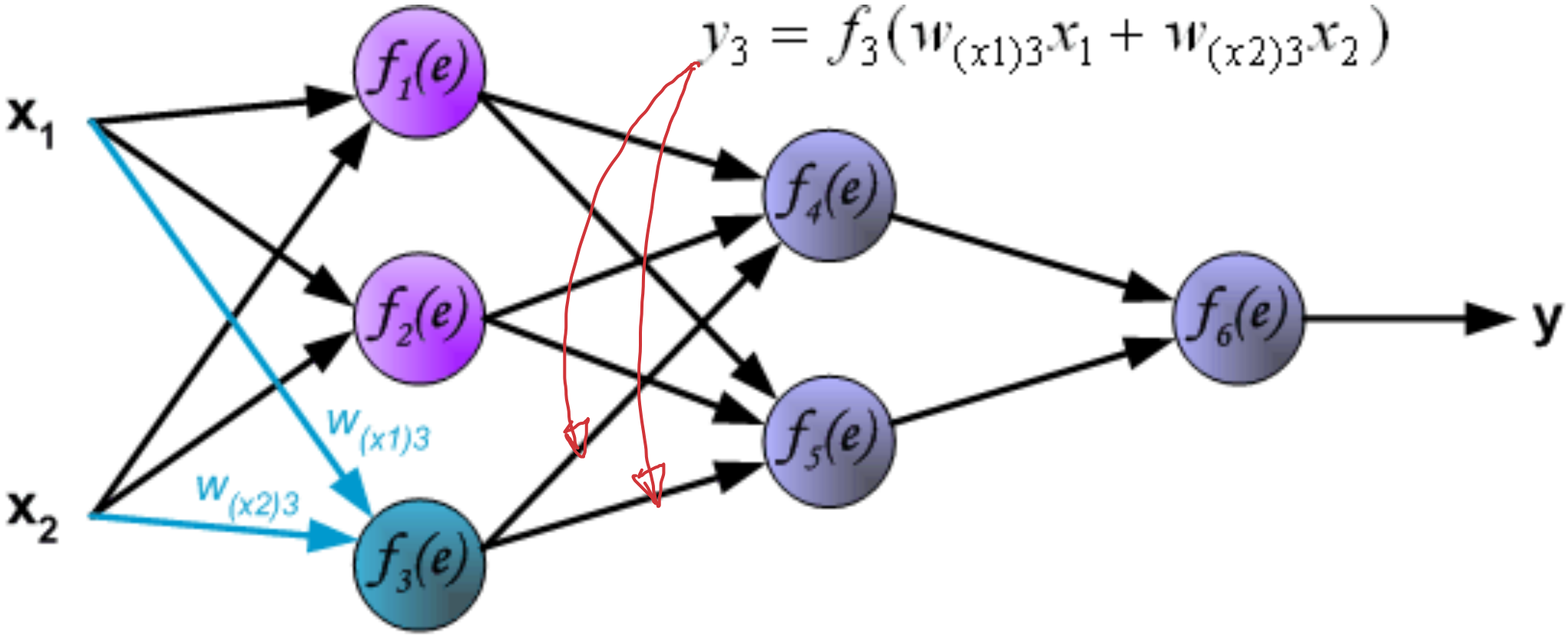
- Network training is an iterative process



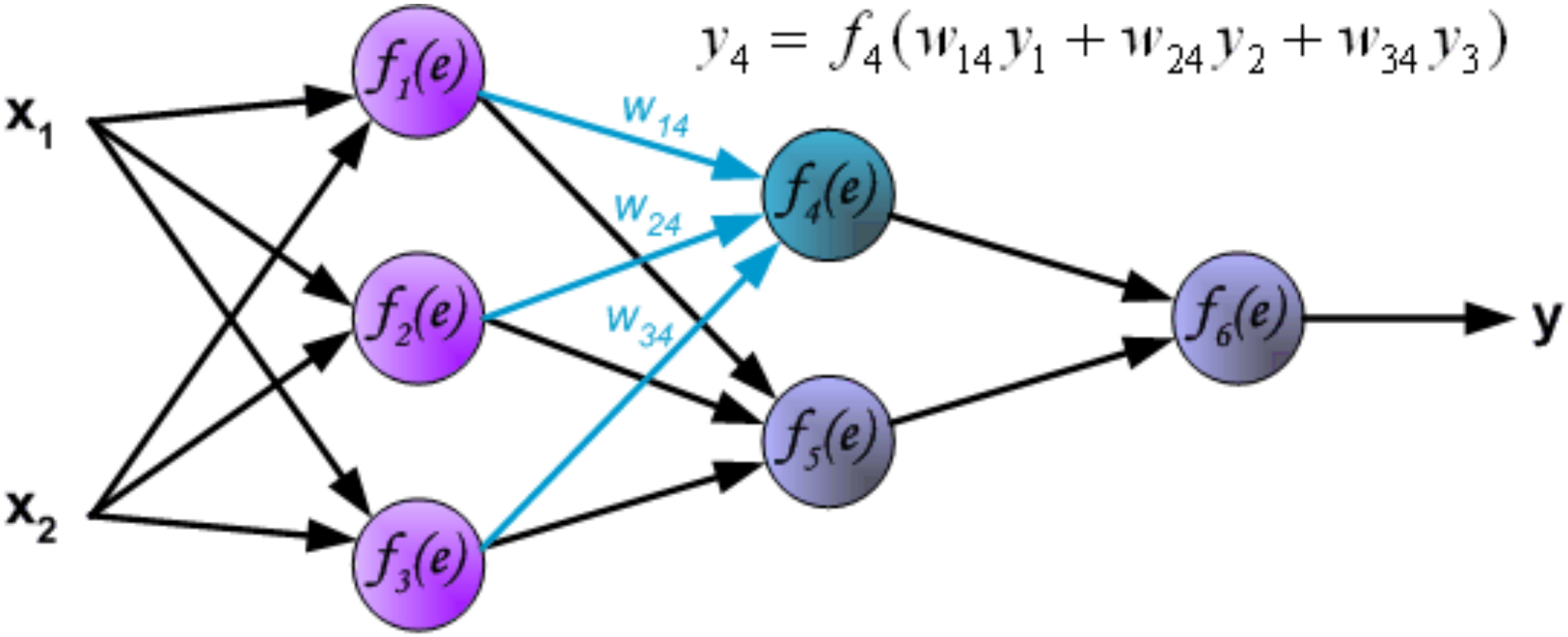
Backpropagation Algorithm (3)



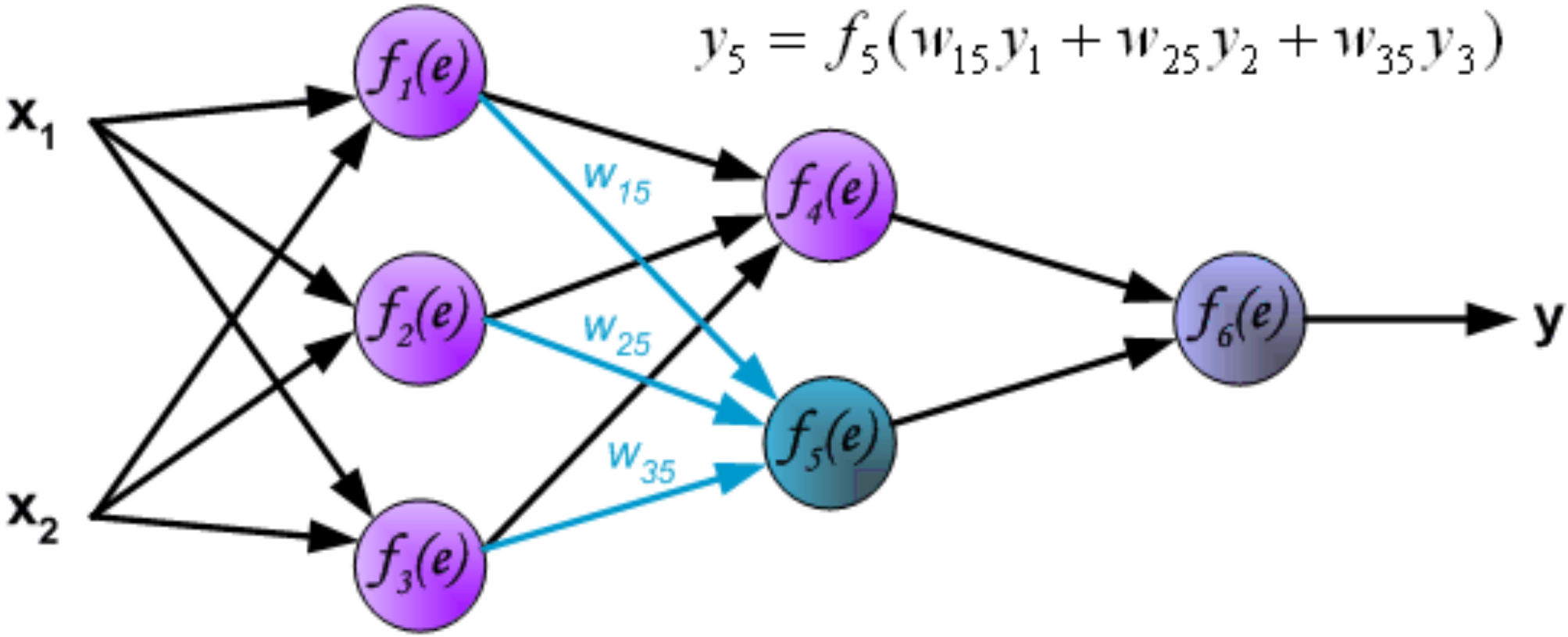
Backpropagation Algorithm (4)



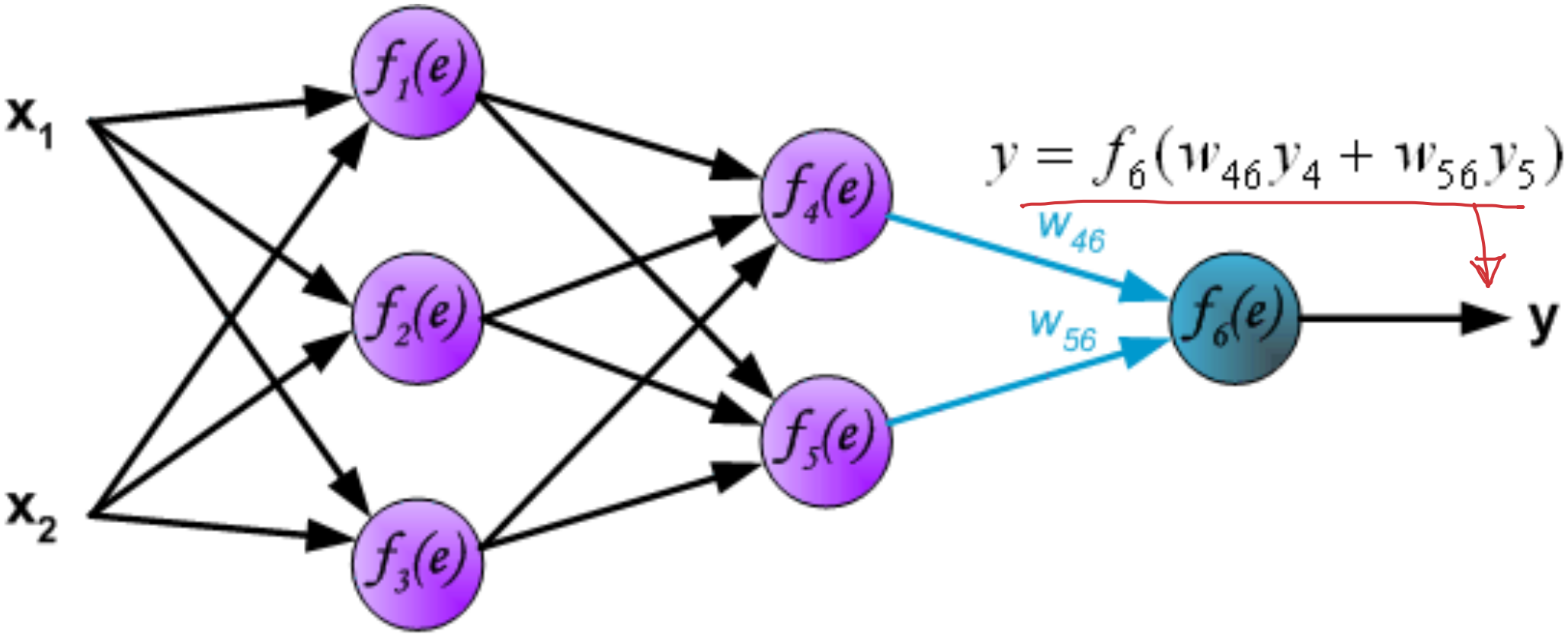
Backpropagation Algorithm (5)



Backpropagation Algorithm (6)

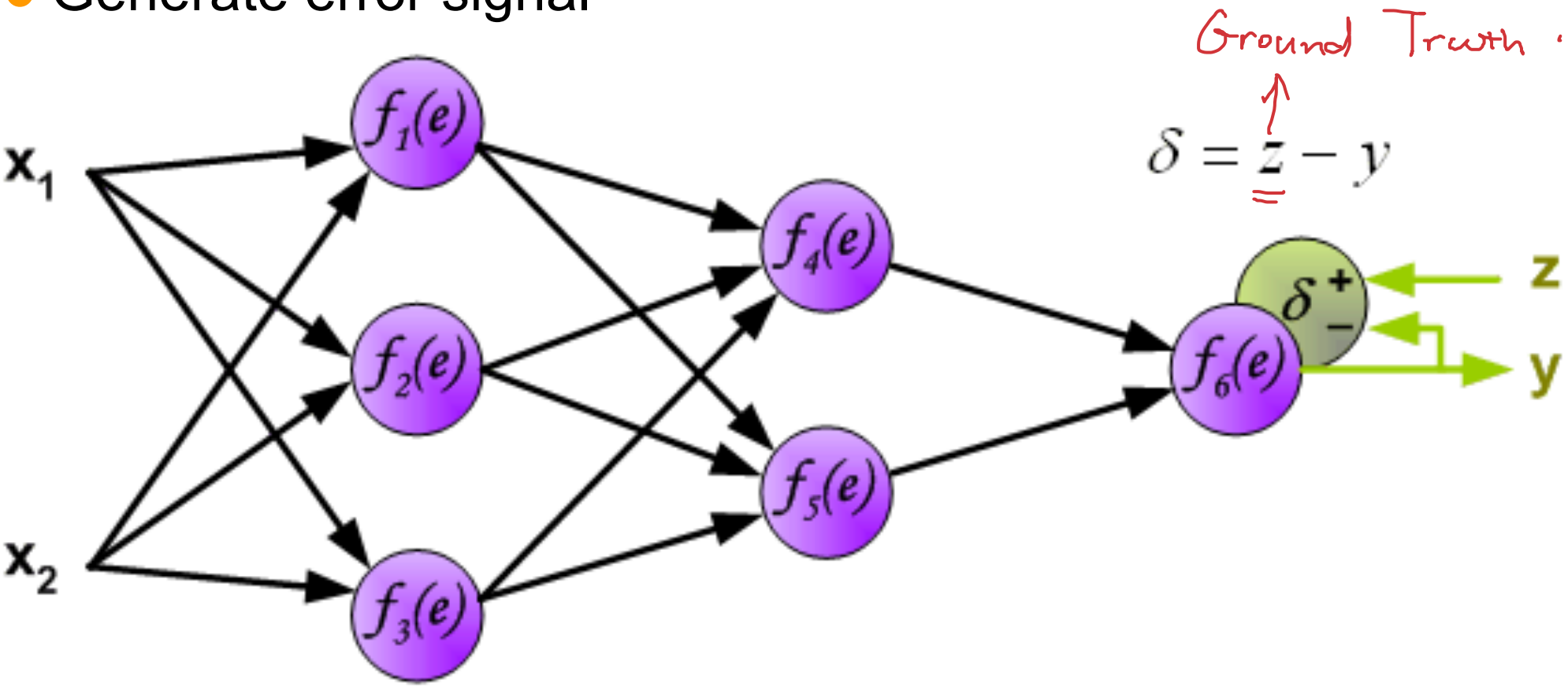


Backpropagation Algorithm (7)



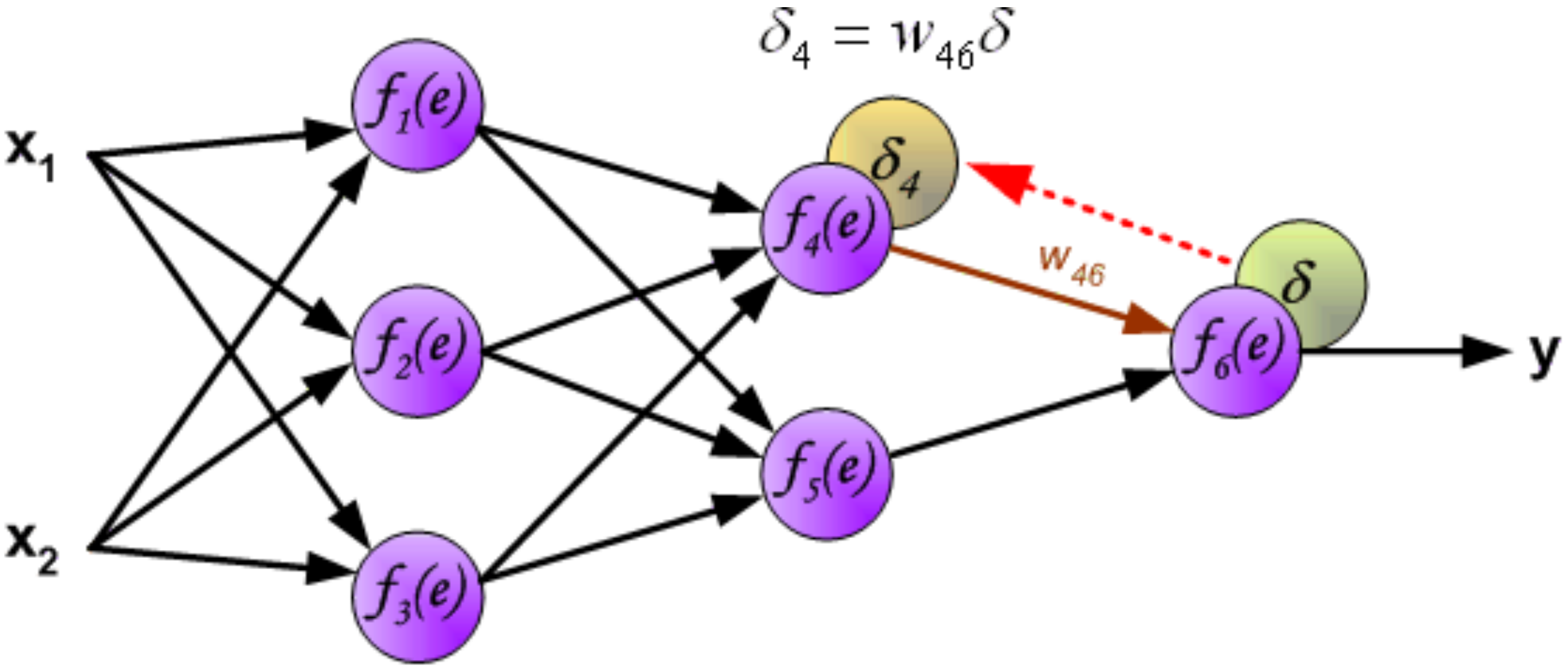
Backpropagation Algorithm (8)

- Generate error signal

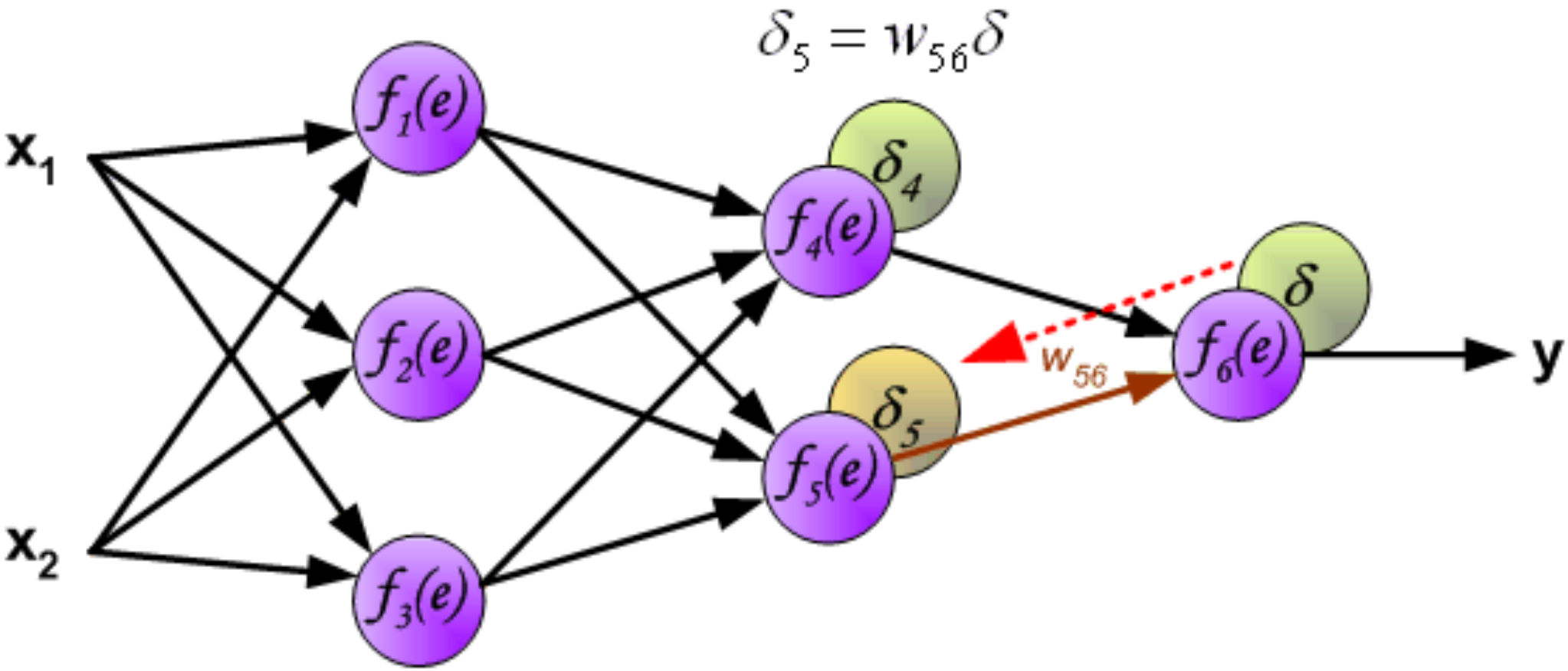


Backpropagation Algorithm (9)

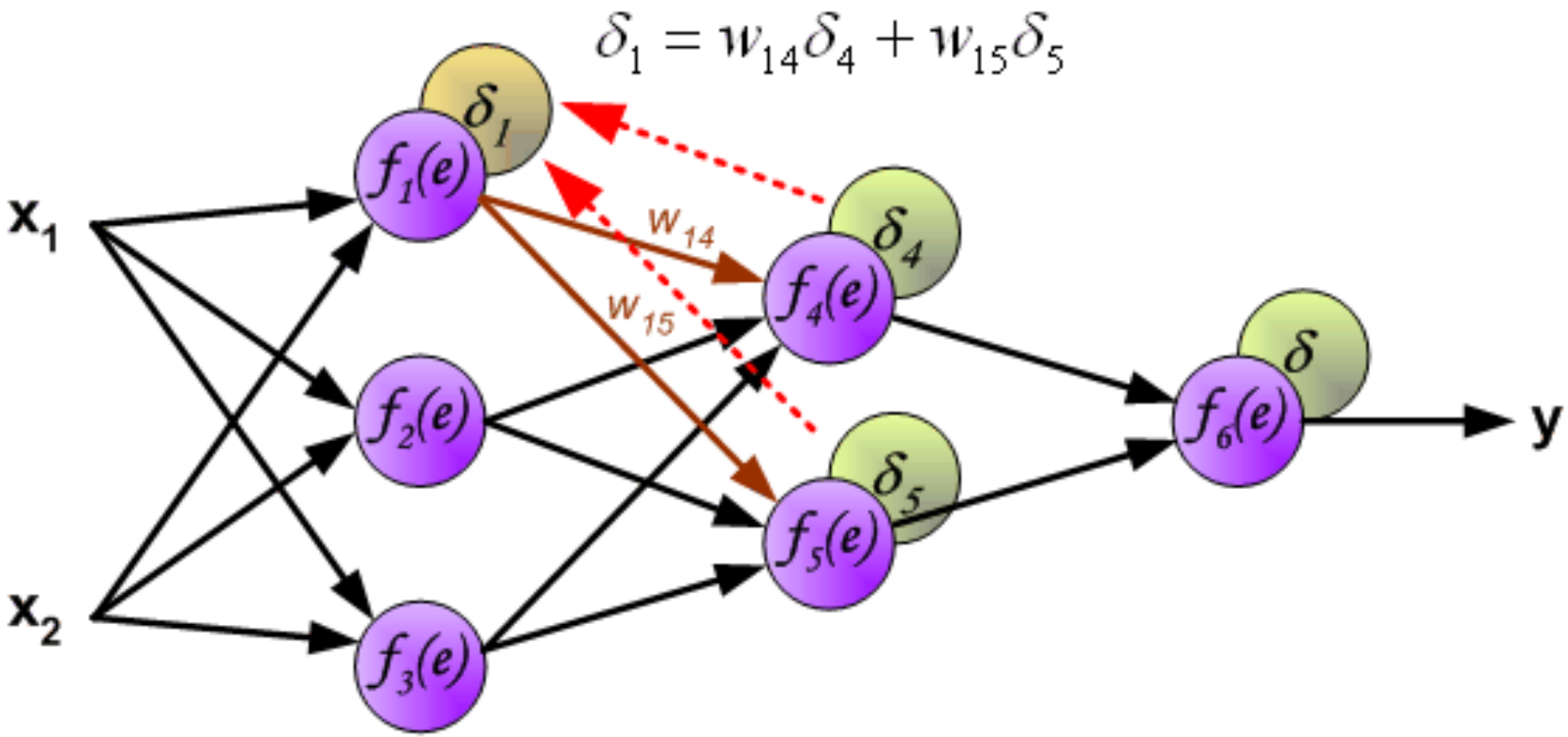
- propagate error signal back to all neurons



Backpropagation Algorithm (10)



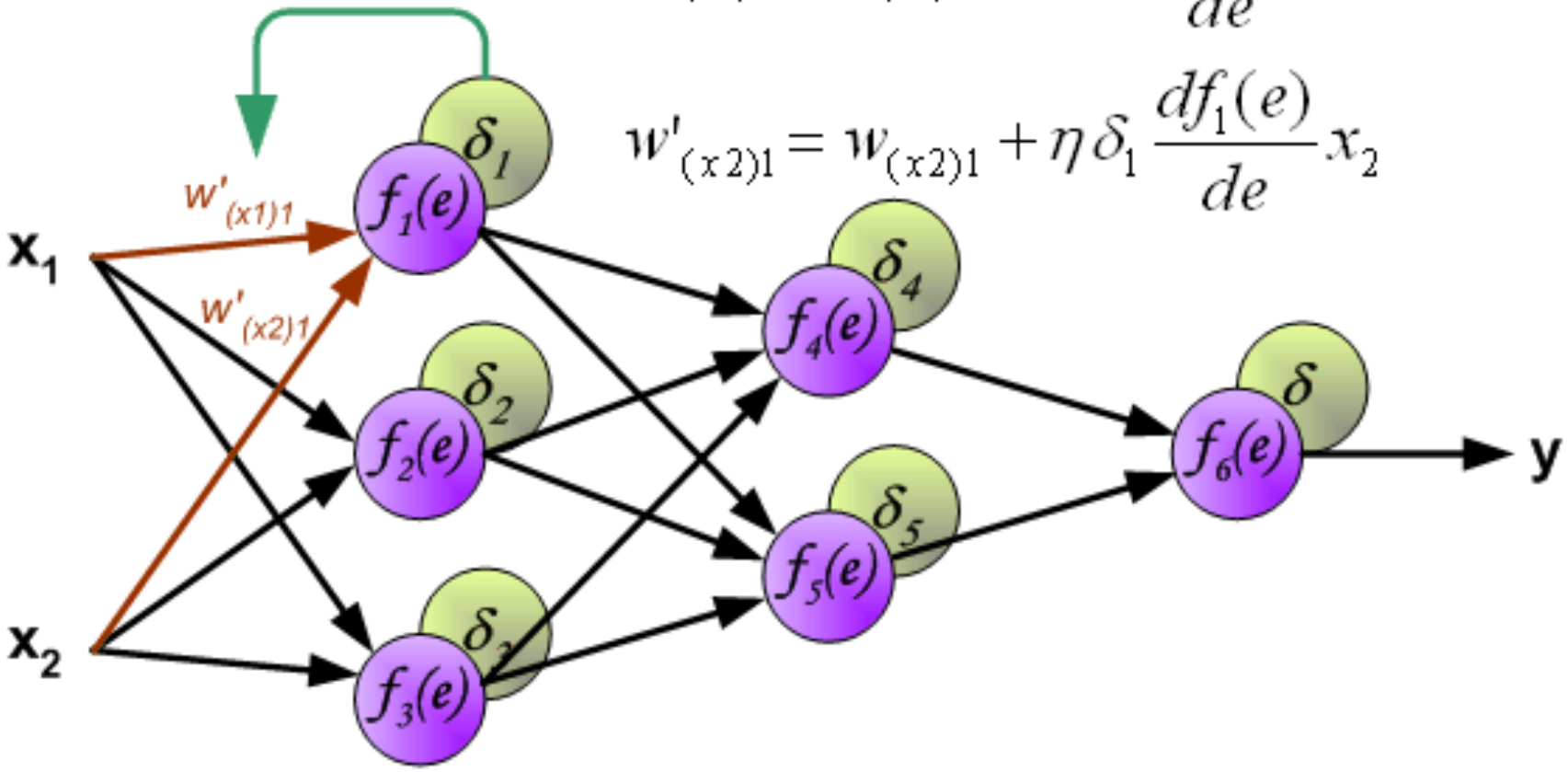
Backpropagation Algorithm (11)



Backpropagation Algorithm (12)

$$w'_{(x1)1} = w_{(x1)1} + \eta \delta_1 \frac{df_1(e)}{de} x_1$$

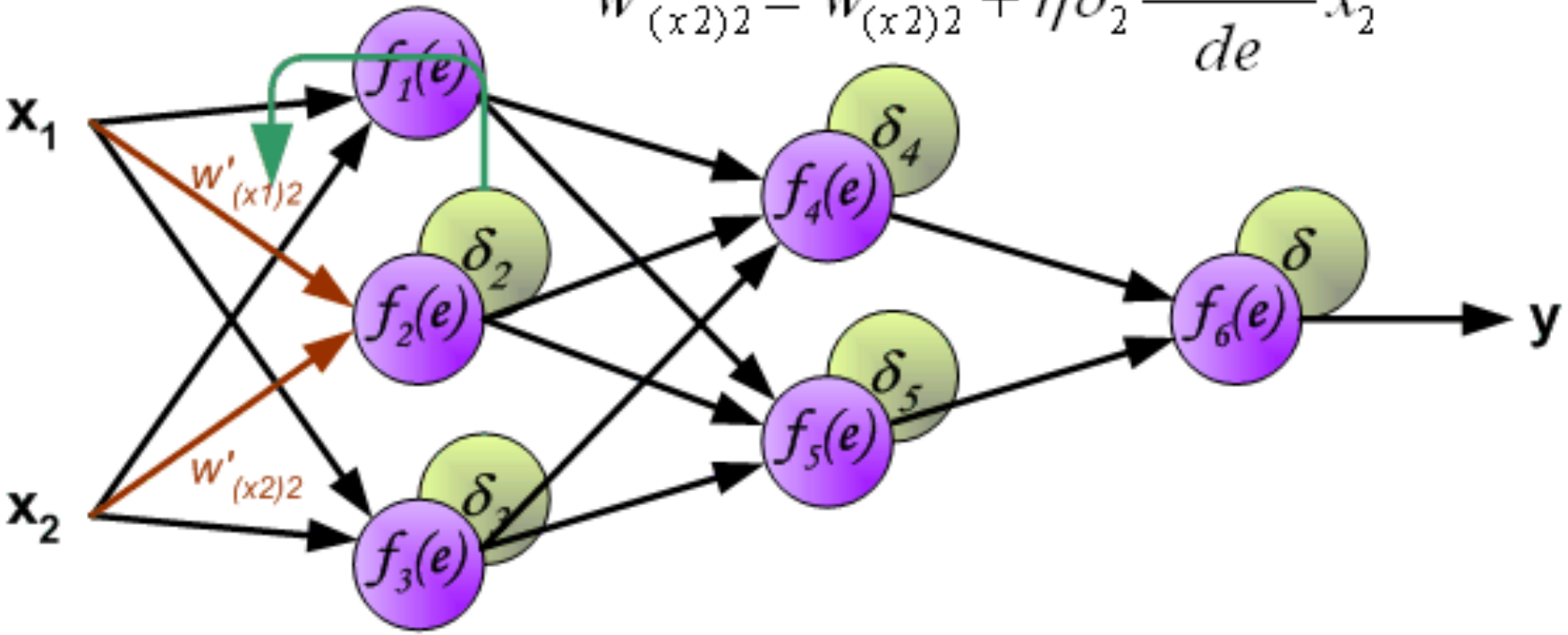
$$w'_{(x2)1} = w_{(x2)1} + \eta \delta_1 \frac{df_1(e)}{de} x_2$$



Backpropagation Algorithm (13)

$$w'_{(x1)2} = w_{(x1)2} + \eta \delta_2 \frac{df_2(e)}{de} x_1$$

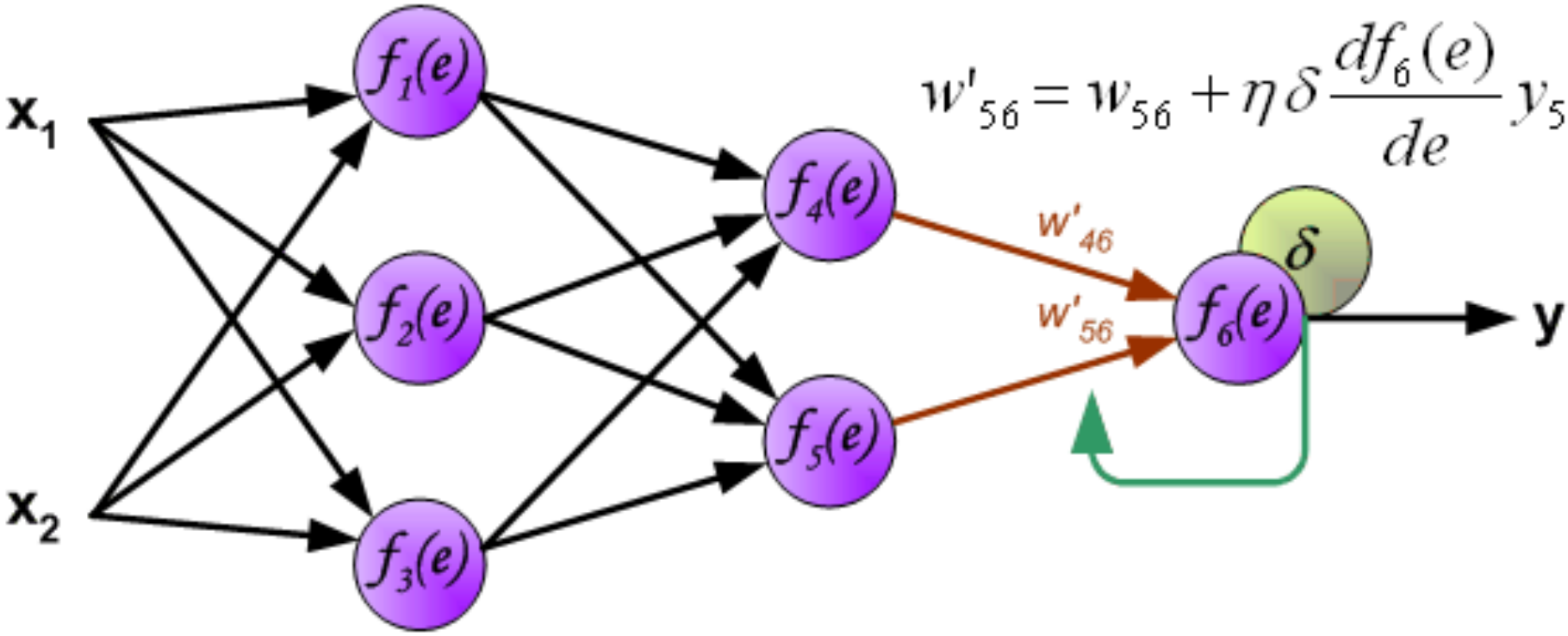
$$w'_{(x2)2} = w_{(x2)2} + \eta \delta_2 \frac{df_2(e)}{de} x_2$$



Backpropagation Algorithm (14)

$$w'_{46} = w_{46} + \eta \delta \frac{df_6(e)}{de} y_4$$

$$w'_{56} = w_{56} + \eta \delta \frac{df_6(e)}{de} y_5$$



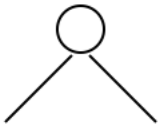
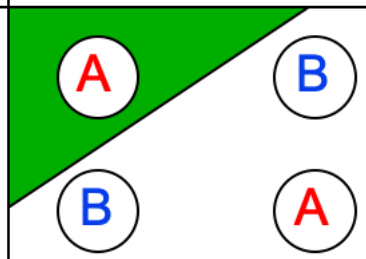
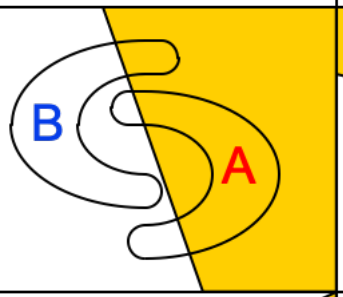
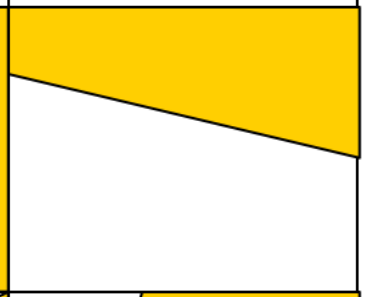
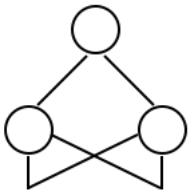
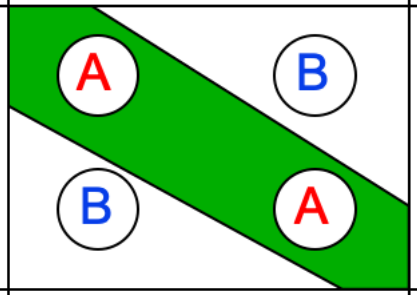
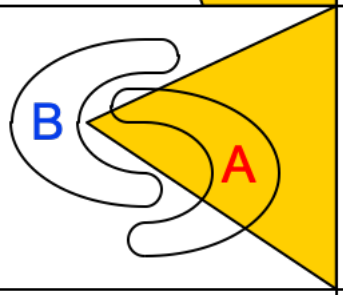
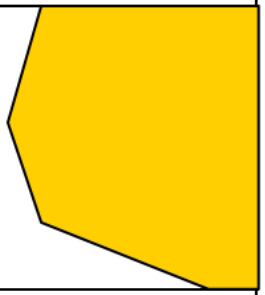
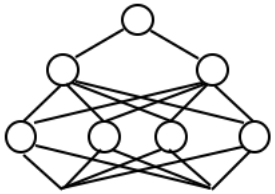
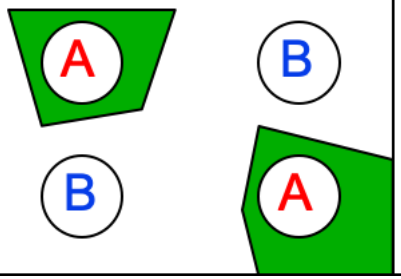
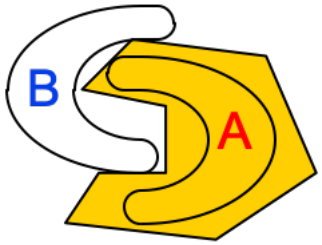
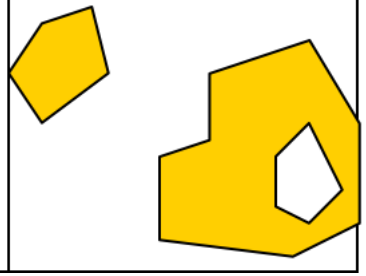
Conclusion (1)

- Perceptron algorithm
  - deals with linearly separable classes
  - converges to a solution in a finite number of iteration steps
  - is a reward and punishment algorithm
- XOR problem
  - cannot be separated by a linear hyperplane
  - two-phase design

Conclusion (2)

- Two-layer perceptron transforms the nonlinearly separable problem into linearly separable one
- Three-layer perceptron can classify vectors into classes consisting of ANY union of polyhedral regions
- Backpropagation algorithm is to propagate error signal back to all neurons.

Conclusion (3)

Structure	Types of Decision Regions	Exclusive-OR Problem	Classes with Meshed regions	Most General Region Shapes
Single-Layer 	Half Plane Bounded By Hyperplane			
Two-Layer 	Convex Open Or Closed Regions			
Three-Layer 	Arbitrary (Complexity Limited by No. of Nodes)			

Classification using Bayes Decision Theory

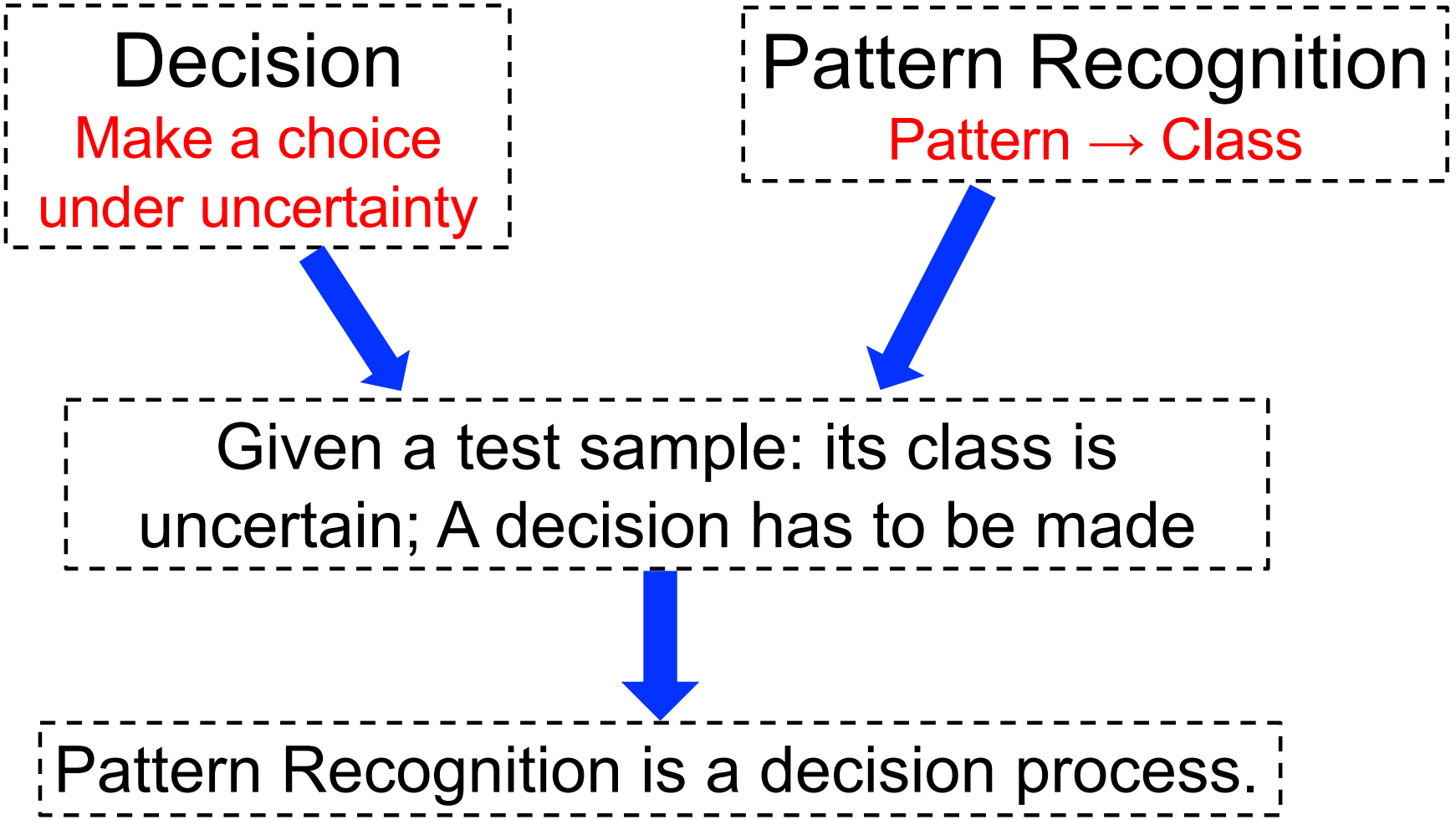
CISC3024 – Pattern Recognition

Prof. Yicong Zhou

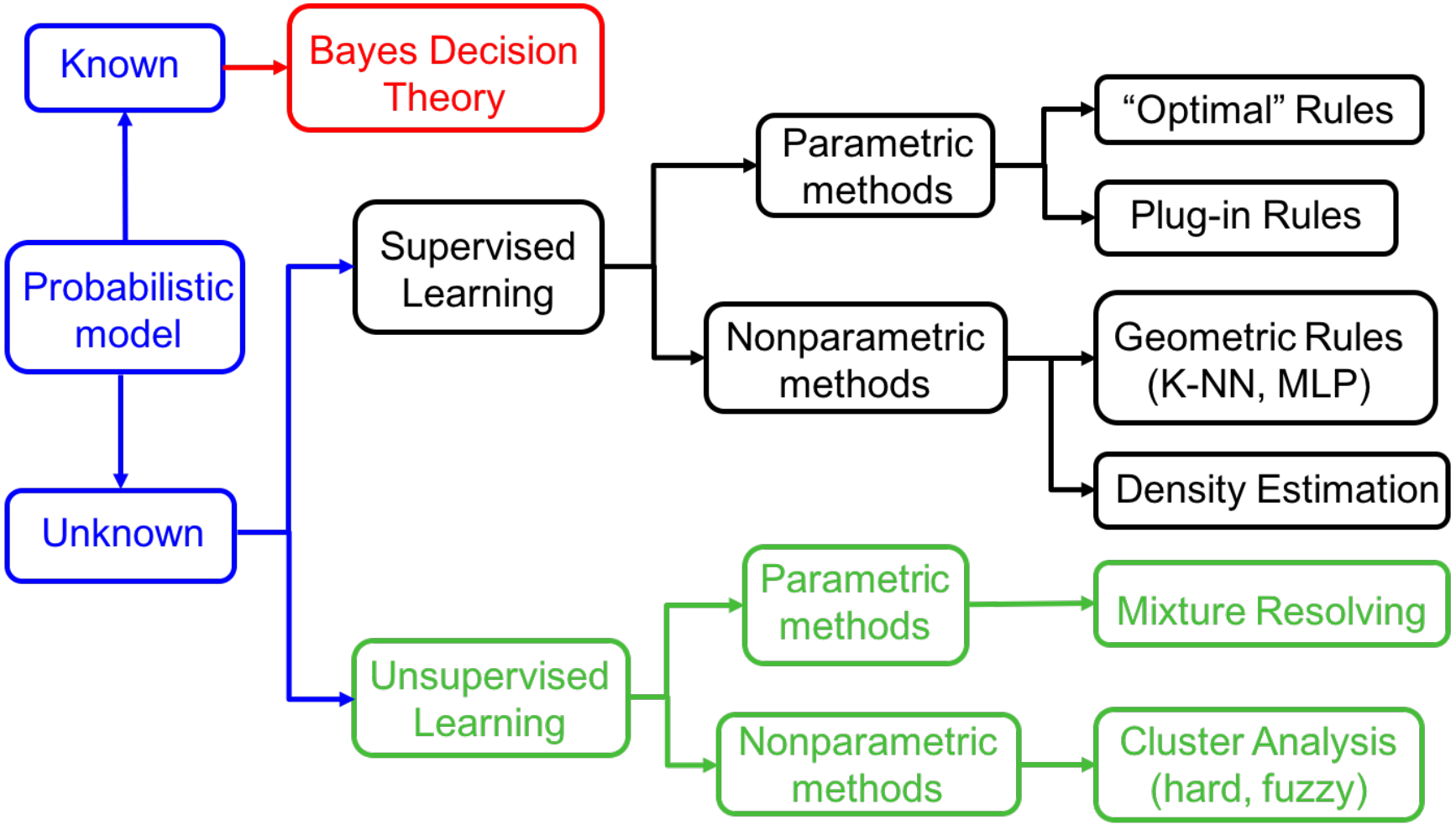
yicongzhou@um.edu.mo

@Fall 2024

Decision vs. Pattern Recognition



Statistic Methods



Outline

- Bayes Decision Theory
- Classification Examples
- Loss Functions
- Discriminant Functions
- Bayesian Classification for Normal Distributions
- Conclusions

Bayes Decision Theory (1)

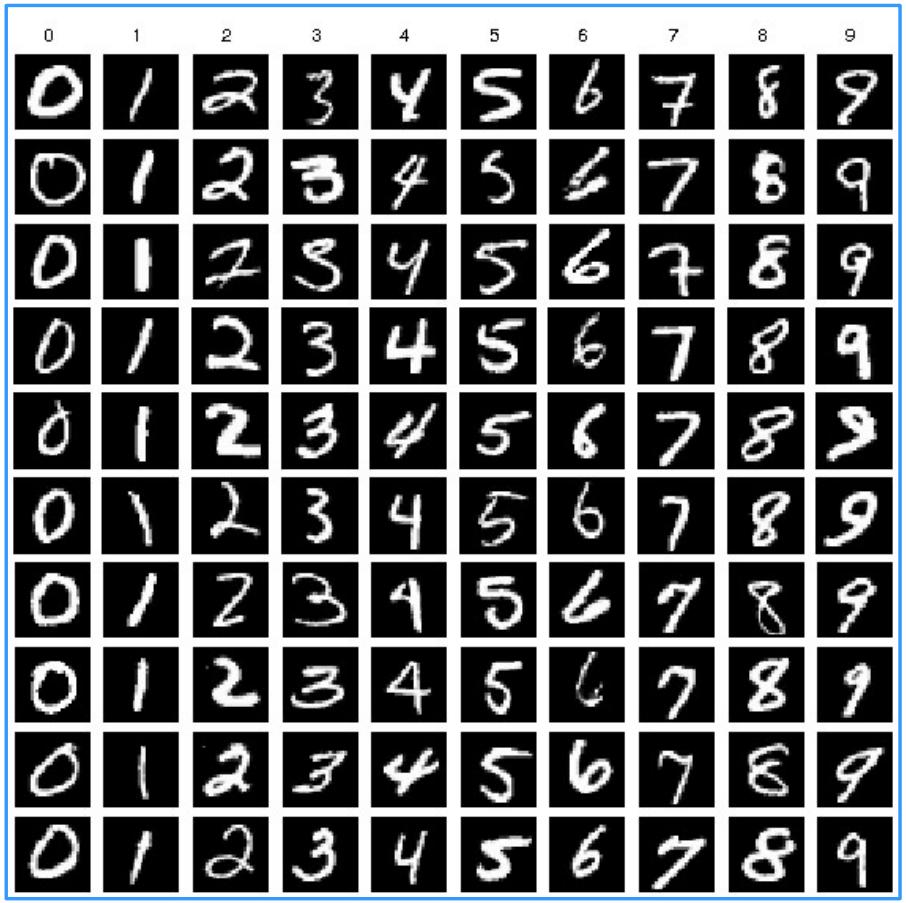
- Bayes Decision Theory
  - statistic method for pattern recognition
  - fundamentals of most pattern recognition algorithms
- Basic Assumptions
  - The decision problem is posed in probabilistic terms
  - all relevant probability values are known

Bayes Decision Theory (2)

- Random Variable ω
  - c available classes: $\omega = \{\omega_1, \omega_2, \dots, \omega_c\}$
  - must be described in probability
- Prior probability $P(\omega)$
  - Probability distribution of random variable ω
  - Prior knowledge on random variable ω
  - $P(\omega) \geq 0$ (Non-negative)
  - $\sum_{i=1}^{c} P(\omega_i) = 1$ (normalization)

Bayes Decision Theory (3)

- MNIST database (Modified National Institute of Standards and Technology database)
  - 10,000 handwritten digits
  - Each image size is 28x28
  - Each digit has an equal probability



Bayes Decision Theory (4)

- Statistical nature of feature vectors: $x = [x_1, x_2, \ldots, x_l]^T$

- Assign a pattern with feature vector x to the most probable of the available classes $\omega = \{\omega_1, \omega_2, \ldots, \omega_c\}$

$$x \longrightarrow \omega_i : \quad \max P(\omega_i | x)$$

Bayes Decision Theory (5)

- **Decision before Observation**
  - **Problem**: make a decision where
    - Prior probability $P(\omega)$ is known
    - No observation is allowed
- Naïve Decision Rule:

$$\text{Decide } \omega_1 \text{ if } P(\omega_1) > P(\omega_2); \text{ Otherwise decide } \omega_2$$

Bayes Decision Theory (6)

- Naïve Decision Rule (Cont.):
  - This is the best we can do with only prior information
  - Fixed prior probabilities $\Rightarrow$ Same decision all the time

- Good when $P(\omega_1)$ is much greater (less) than $P(\omega_2)$
- Poor when $P(\omega_1)$ is close to $P(\omega_2)$
- Only 50% chance of being right if $P(\omega_1) = P(\omega_2)$

Making decision with observations !!

Bayes Decision Theory (7)

- **Bayes Formula:**

$$P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{p(x)}$$

$$posterior = \frac{likelihood \times prior}{evidence}$$

  - Likelihood: class-conditional PDF, $p(x|\omega_j)$ where $1 \le j \le c$
  - Prior probability $P(\omega_j)$
  - Evidence: unconditional density of x, $p(x) = \sum_{j=1}^{c} p(x|\omega_j)P(\omega_j)$
  - Unknown posterior probability, $p(\omega_j|x)$

Bayes Decision Theory (8)

- Bayes Decision Rule:

$$\text{If} \quad P(\omega_j|x) > P(\omega_i|x), \ \forall i \neq j \quad \Rightarrow \quad \text{Decide} \quad \omega_j$$

  - $P(\omega_j)$ and $p(x|\omega_j)$ are assumed to be known
  - $p(x)$ is irrelevant to Bayesian decision
  - $p(x)$ is a normalization factor and unrelated to any class

$$p(x) = \sum_{j=1}^{c} p(\omega_j, x) = \sum_{j=1}^{c} p(x|\omega_j) P(\omega_j)$$

Bayes Decision Theory (9)

- Special Case #1:
  - Equal prior probability: $P(\omega_1) = P(\omega_2) = \cdots = P(\omega_c) = \frac{1}{c}$

  ⇛ Depends on the likelihood $p(x|\omega_j)$
- Special Case #2:
  - Equal likelihood: $p(x|\omega_1) = p(x|\omega_2) = \cdots = p(x|\omega_c)$

  ⇛ Reverse back to **Naïve Decision Rule**

Bayes Decision Theory (10)

- Bayes Decision Rule (for two classes $c = 2$):

$$\text{If } P(\omega_1|x) > P(\omega_2|x), \text{Decide } \omega_1; \text{Otherwise } \omega_2$$

- Equivalently,

$$\text{If } \ p(x|\omega_1)P(\omega_1) > p(x|\omega_2)P(\omega_2), \text{Decide } \omega_1; \text{Otherwise } \omega_2$$

- For equiprobable classes: $P(\omega_1) = P(\omega_2)$

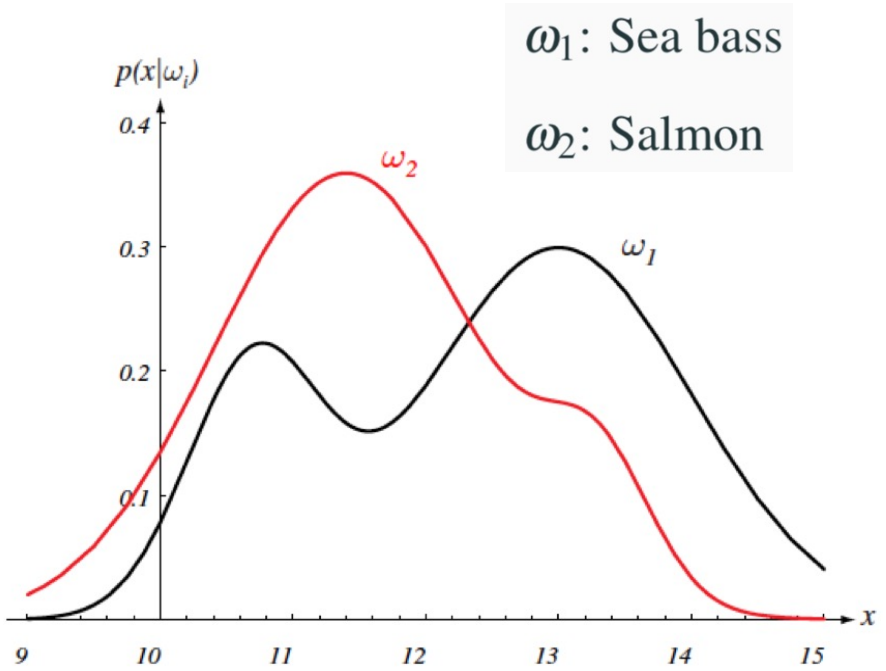
$$\text{If } \ p(x|\omega_1) > p(x|\omega_2), \text{Decide } \omega_1; \text{Otherwise } \omega_2$$

Outline

- Bayes Decision Theory
- Classification Examples
- Loss Functions
- Discriminant Functions
- Bayesian Classification for Normal Distributions
- Conclusions

Classification Examples (1)

- Example 1: Sea bass vs. Salmon



$$P(\omega_1) = \frac{2}{3}$$

$$P(\omega_2) = \frac{1}{3}$$

Likelihood of observing x (lightness) for given class label ω_j (fish type)

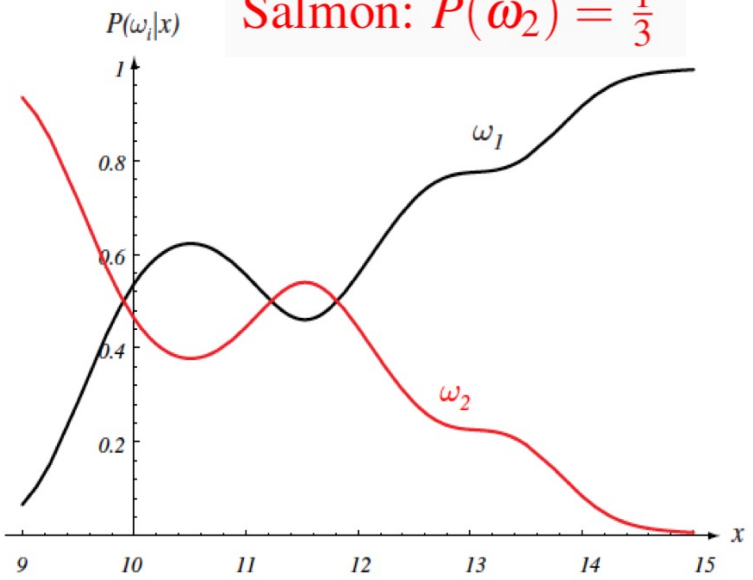
What will be the posterior probability for either fish type look like?

Classification Examples (2)

- Example 1: Sea bass vs. Salmon (Cont.)

Sea bass: $P(\omega_1) = \frac{2}{3}$

Salmon: $P(\omega_2) = \frac{1}{3}$



Posterior probability for either fish type

For example: $x = 14$

$P(\omega_1|x) = 0.92$
$P(\omega_2|x) = 0.08$

For each value of x

- the higher curve yields the output of Bayesian decision
- the posteriors of either curve sum to 1.0

Classification Examples (3)

- **Example 2: Cancer detection**

  Problem statement

  - A new medical test is used to detect whether a patient has a certain cancer or not. His test result is either + (positive) or - (negative)
  - For the patient with this cancer, the probability of returning positive test result is 0.98
  - For the patient without this cancer, the probability of returning negative test result is 0.97
  - The probability of any person having this cancer is 0.008

  Question

  If positive test result is returned for some person, does the patient have this cancer or not?

Classification Examples (4)

ω_1: cancer ω_2: no cancer $x \in \{+, -\}$: test result

$$P(\omega_1) = 0.008 \qquad P(\omega_2) = 1 - P(\omega_2) = 0.992$$

$$P(+|\omega_1) = 0.98 \qquad P(-|\omega_1) = 1 - P(+|\omega_1) = 1 - 0.98 = 0.02$$

$$P(-|\omega_2) = 0.97 \qquad P(+|\omega_2) = 1 - P(-|\omega_2) = 1 - 0.97 = 0.03$$

$$P(\omega_1|+) = \frac{P(\omega_1)P(+|\omega_1)}{P(+)} = \frac{P(\omega_1)P(+|\omega_1)}{P(\omega_1)P(+|\omega_1) + P(\omega_2)P(+|\omega_2)}$$

$$= \frac{0.008 \times 0.98}{0.008 \times 0.98 + 0.992 \times 0.03} = 0.2085$$

$$P(\omega_2|+) > P(\omega_1|+)$$

No cancer !!

$$P(\omega_2|+) = 1 - P(\omega_1|+) = 1 - 0.2085 = 0.7915$$

Classification Examples (5)

- **Example 3: Car price**

Problem statement

Based on the height of a car in our campus, decide its cost more than \$50,000 ?

ω_1: price $>$\$50,000

ω_2: price $\leqslant$\$50,000

x: car height

$P(\omega_1|x) > P(\omega_2|x)$?

or

$P(\omega_1|x) < P(\omega_2|x)$?

Classification Examples (6)

Collecting samples

suppose we randomly select 1500 cars in our campus, and have their prices and heights.

cars in ω_1: 300

$$P(\omega_1) = \frac{300}{1500} = 0.2$$

cars in ω_2: 1200

$$P(\omega_2) = \frac{1200}{1500} = 0.8$$

Compute $p(x|\omega_1)$ and $p(x|\omega_2)$

Collect # of cars whose height falling into range $R \in [1.0m, 1.1m]$

cars of ω_1: 30

$$p(x = 1.05|\omega_1) = \frac{30}{300} = 0.1$$

cars of ω_2: 60

$$p(x = 1.05|\omega_1) = \frac{60}{1200} = 0.05$$

Classification Examples (7)

Question

For a car with height 1.05m, is its price greater than \$50,000?

We have known:

$P(\omega_1) = 0.2$ $\quad p(x = 1.05|\omega_1) = 0.1$

$P(\omega_2) = 0.8$ $\quad p(x = 1.05|\omega_2) = 0.05$

Classification Examples (8)

Bayesian Decision

$$\frac{P(\omega_2|x=1.05)}{P(\omega_1|x=1.05)} = \frac{P(\omega_2)p(x=1.05|\omega_2)}{p(x=1.05)} \Big/ \frac{P(\omega_1)p(x=1.05|\omega_1)}{p(x=1.05)}$$

$$= \frac{P(\omega_2)p(x=1.05|\omega_2)}{P(\omega_1)p(x=1.05|\omega_1)} = \frac{0.8 \times 0.05}{0.2 \times 0.1} = 2$$

Thus, $P(\omega_2|x) > P(\omega_1|x)$

For care height $x = 1.05m$, its price $\leqslant \$50,000$

Outline

- Bayes Decision Theory
- Classification Examples
- Loss Functions
- Discriminant Functions
- Bayesian Classification for Normal Distributions
- Conclusions

Loss Functions (1)

- How to know the Bayes decision rule is optimal?
  - probability statement ⇛ Decision
  - some classification mistakes can be more costly than others
  - need a decision evaluation method
- Bayes Decision Rule (for two classes $c = 2$)

$$\text{If } P(\omega_1|x) > P(\omega_2|x), \quad x \rightarrow \omega_1$$

$$\text{If } P(\omega_2|x) > P(\omega_1|x), \quad x \rightarrow \omega_2$$

Loss Functions (2)

- Whenever we observe a particular x, the probability of error is:

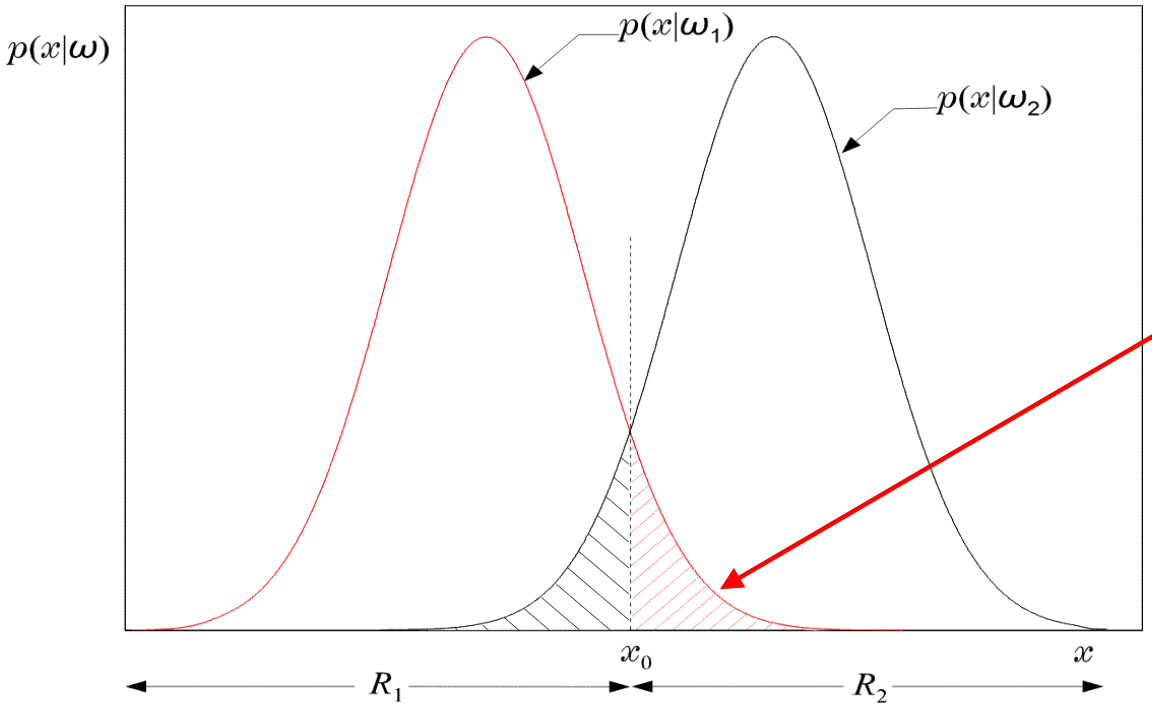
$$P(error|x) = \begin{cases} P(\omega_1|x) & if\ decide\ \omega_2 \\ P(\omega_2|x) & if\ decide\ \omega_1 \end{cases}$$

- Using Bayes decision rule,

$$P(error|x) = min[P(\omega_1|x), P(\omega_2|x)]$$

- For each x, $P(error|x)$ should be as small as possible
- average probability of error should be as small as possible for all possible x

Loss Functions (3)



probability of error:

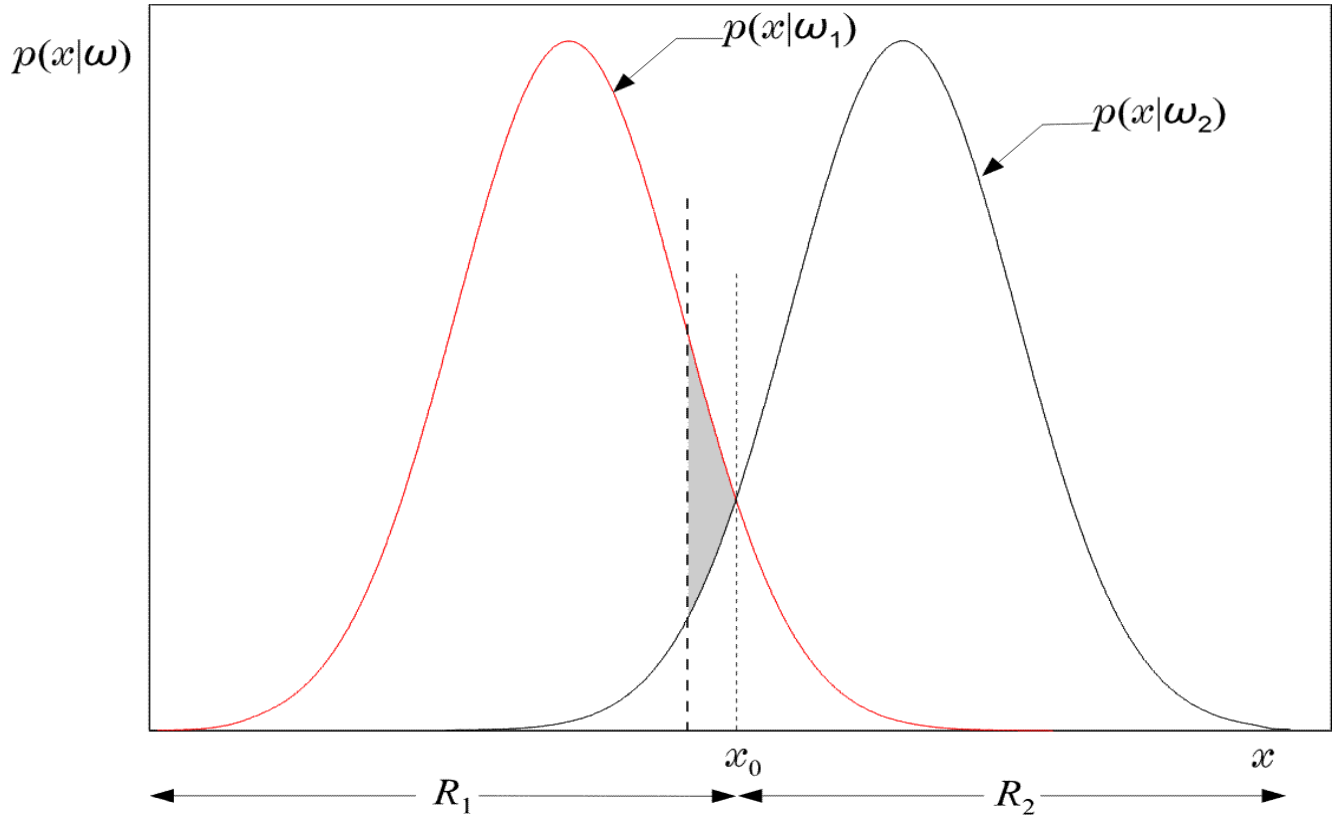
$$P_e = \frac{1}{2}\int_{-\infty}^{x_0} p(x|\omega_2)\,\mathrm{d}x + \frac{1}{2}\int_{x_0}^{+\infty} p(x|\omega_1)\,\mathrm{d}x$$

$R_1 \rightarrow \omega_1$

$R_2 \rightarrow \omega_2$

Bayesian classifier is optimal with respect to minimizing the classification error probability !!!

Loss Functions (4)



moving the threshold the total shaded area increases by the extra “gray” area

Loss Functions (5)

- A loss function is more general than probability of error
  - $\lambda(\alpha_i|\omega_j)$: $\Omega \times \mathcal{A} \rightarrow \boldsymbol{R}$ (loss function)
  - $\lambda(\alpha_i|\omega_j)$: the loss of taking action α_i when the class is ω_j

where

- more than one feature,

  $x \in \boldsymbol{R} \Longrightarrow x \in \boldsymbol{R}^d$ (d-dimensional Euclidean space)
- c classes (states of nature): $\Omega = \{\omega_1, \omega_2, \ldots, \omega_c\}$
- actions (instead of only deciding the class):

  $\mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_a\}$ (finite set of a possible actions, $a \neq c$)

Loss Functions (6)

- Problem:
  - decide taking action α_i for a given x
  - need to know the loss of taking each action α_i $(1 \leq i \leq a)$
  - incur the loss of $\lambda(\alpha_i|\omega_j)$ for the true class ω_j and action α_i
  - the true class ω_j is unknown ⇛ Expected (Average) loss
- Expected loss (Conditional risk)

$$R(\alpha_i|x) = \sum_{j=1}^{c} \lambda(\alpha_i|\omega_j) P(\omega_j|x)$$

$\lambda(\alpha_i|\omega_j)$: the incurred loss of taking action α_i in the true class ω_j

$P(\omega_j|x)$: the probability of ω_j being the true class of x

Loss Functions (7)

- Example:

Class \ Action	α_1 = "Recipe A"	α_2 = "Recipe B"	α_3 = "No Recipe"
ω_1 = "Cancer"	5	50	10,000
ω_2 = "No cancer"	60	3	0

For a specific x: $P(\omega_1|x) = 0.01 \qquad P(\omega_2|x) = 0.99$

$$\mathbf{R}(\alpha_1|\mathbf{x}) = \sum_{j=1}^{2} \lambda(\alpha_1|\omega_j)P(\omega_j|x)$$

$$= \lambda(\alpha_1|\omega_1)P(\omega_1|x) + \lambda(\alpha_1|\omega_2)P(\omega_2|x)$$

$$= 5 \times 0.01 + 60 \times 0.99 = 59.45$$

Similarly, we can obtain $\mathbf{R}(\alpha_2|\mathbf{x}) = 3.47$ and $\mathbf{R}(\alpha_3|\mathbf{x}) = 100$

Loss Functions (8)

- Task (General case):
  - to find a mapping from patterns to actions

$$\alpha: \boldsymbol{R}^d \rightarrow \mathcal{A} \quad \text{(decision function)}$$

  - for every x, the decision function $\alpha(x)$ determines one of a actions $(\alpha_1, \alpha_2, \ldots, \alpha_a)$

Loss Functions (9)

- Overall risk $\mathbf{R}$:

$$\boldsymbol{R} = \int \boldsymbol{R}(\alpha(x)|x)p(x)dx$$

where $p(x)$: PDF of patterns, and

$\boldsymbol{R}(\alpha(x)|x)$: Conditional risk of pattern x with action $\alpha(x)$

- it is the expected loss with decision function $\alpha(\cdot)$
- for every x, conditional risk $\boldsymbol{R}(\alpha(x)|x)$ must be as small as possible
- the overall risk for all possible x must be as small as possible

Loss Functions (10)

- Bayes risk (General case):

$$\alpha(x) = \underset{\alpha_i \in \boldsymbol{A}}{\operatorname{argmin}} \boldsymbol{R}(\alpha_i|x) = \underset{\alpha_i \in \boldsymbol{A}}{\operatorname{argmin}} \sum_{j=1}^{c} \lambda(\alpha_i|\omega_j) P(\omega_j|x)$$

  - $\lambda_{ij} = \lambda(\alpha_i|\omega_j)$: the loss incurred for taking α_i when the true state is ω_j
  - Minimizing the expected loss $\boldsymbol{R}(\alpha_i|x)$, a decision α_i is made for every x
  - Minimizing the overall risk for all possible x leads to Bayes risk
  - The best performance can be achieved for given $p(x)$ and loss function

Loss Functions (11)

- **Special case**: two-category classification

- Define the loss matrix: $L = \begin{pmatrix} \lambda(\alpha_1|\omega_1) & \lambda(\alpha_1|\omega_2) \\ \lambda(\alpha_2|\omega_1) & \lambda(\alpha_2|\omega_2) \end{pmatrix} = \begin{pmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{pmatrix}$

- Conditional risk:

$$R(\alpha_1|x) = \lambda_{11}P(\omega_1|x) + \lambda_{12}P(\omega_2|x)$$

$$R(\alpha_2|x) = \lambda_{21}P(\omega_1|x) + \lambda_{22}P(\omega_2|x)$$

Loss Functions (12)

- When decide ω_1, we have conditional risk $R(\alpha_1|x) < R(\alpha_2|x)$

$$\Rightarrow \quad \lambda_{11}P(\omega_1|x) + \lambda_{12}P(\omega_2|x) < \lambda_{21}P(\omega_1|x) + \lambda_{22}P(\omega_2|x)$$

$$\Rightarrow \quad (\lambda_{21} - \lambda_{11})P(\omega_1|x) > (\lambda_{12} - \lambda_{22})P(\omega_2|x)$$

- Because the loss of being error is greater than the loss of being correct, $\lambda_{21} - \lambda_{11} > 0$, re-arrange the above equation,

$$\frac{P(\omega_1|x)}{P(\omega_2|x)} > \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{11}}$$

Loss Functions (13)

- According to Bayes formula

$$\frac{P(\omega_1|x)}{P(\omega_2|x)} = \frac{\mathrm{p}(x|\omega_1)P(\omega_1)}{p(x|\omega_2)P(\omega_2)} > \frac{\lambda_{12}-\lambda_{22}}{\lambda_{21}-\lambda_{11}} \Rightarrow \frac{\mathrm{p}(x|\omega_1)}{p(x|\omega_2)} > \frac{\lambda_{12}-\lambda_{22}}{\lambda_{21}-\lambda_{11}} \cdot \frac{P(\omega_2)}{P(\omega_1)}$$

- Let constant $\theta_\lambda = \frac{\lambda_{12}-\lambda_{22}}{\lambda_{21}-\lambda_{11}} \cdot \frac{P(\omega_2)}{P(\omega_1)}$, it is independent of x
- We have

Decide ω_1, if likelihood ratio $\frac{p(x|\omega_1)}{p(x|\omega_2)} > \theta_\lambda$

Loss Functions (14)

- If $P(\omega_1) = P(\omega_2) = \frac{1}{2}$ and $\lambda_{11} = \lambda_{22} = 0$

$$x \to \omega_1 \text{ if } P(x|\omega_1) > P(x|\omega_2)\frac{\lambda_{12}}{\lambda_{21}}$$

$$x \to \omega_2 \text{ if } P(x|\omega_2) > P(x|\omega_1)\frac{\lambda_{21}}{\lambda_{12}}$$

- If $\lambda_{21} = \lambda_{12}$: Minimum classification error probability

Loss Functions (15)

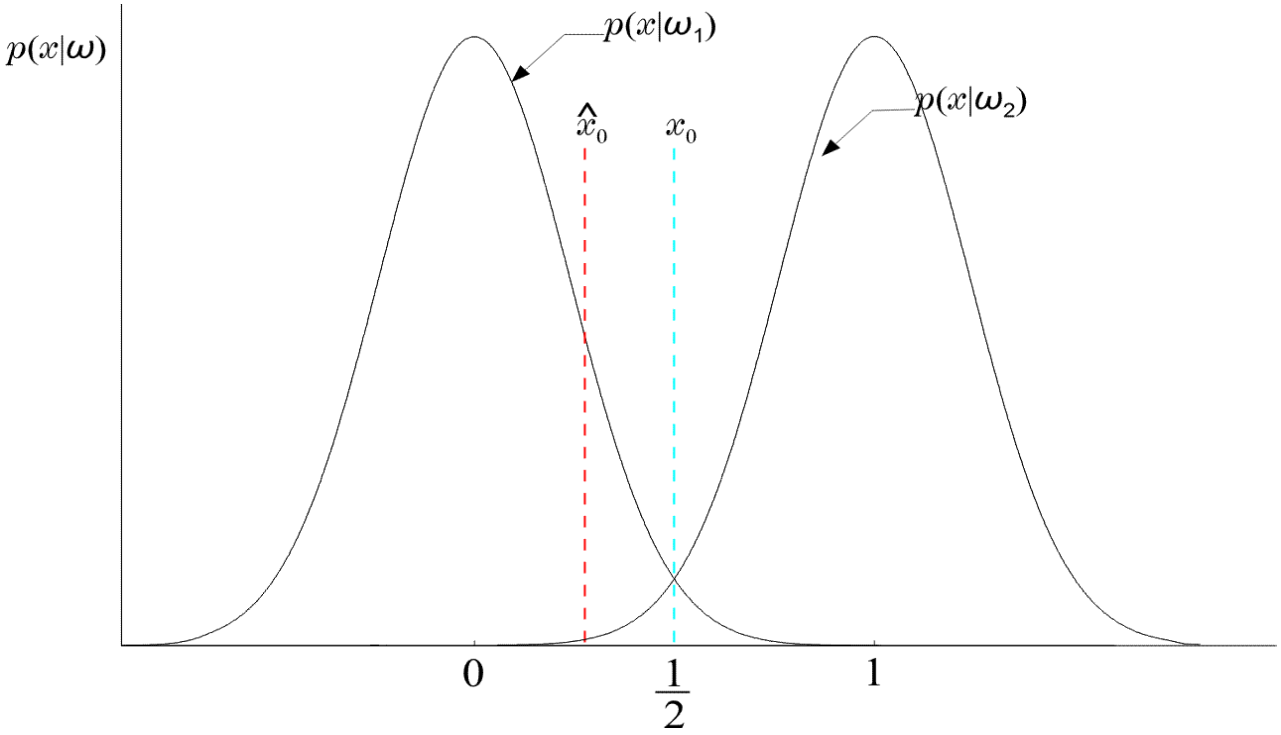
- Example:

$$P(x|\omega_1) = \frac{1}{\sqrt{\pi}} e^{-x^2}$$

$$P(x|\omega_2) = \frac{1}{\sqrt{\pi}} e^{-(x-1)^2}$$

$$P(\omega_1) = P(\omega_2) = \frac{1}{2}$$

$$\mathrm{L} = \begin{pmatrix} 0 & 1.0 \\ 0.5 & 0 \end{pmatrix}$$



Loss Functions (16)

- the threshold value is x_0 for minimum P_e

$$P(x_0|\omega_1) = P(x_0|\omega_2) \Rightarrow \frac{1}{\sqrt{\pi}} e^{-x_0^2} = \frac{1}{\sqrt{\pi}} e^{-(x_0-1)^2} \Rightarrow x_0 = \frac{1}{2}$$

- threshold $\hat{x}_0$ for minimum $R(\alpha_i|x)$

$$P(\hat{x}_0|\omega_1) = P(\hat{x}_0|\omega_2)\frac{\lambda_{12}}{\lambda_{21}} = 2P(\hat{x}_0|\omega_2)$$

$$\Rightarrow \frac{1}{\sqrt{\pi}} e^{-\hat{x}_0^2} = \frac{2}{\sqrt{\pi}} e^{-(\hat{x}_0-1)^2}$$

$$\Rightarrow \hat{x}_0 = \frac{1 - ln2}{2} < \frac{1}{2}$$

Loss Functions (17)

- Minimum-Error-Rate Classification
  - $\Omega = \{\omega_1, \omega_2, \ldots, \omega_c\}$: c possible classes
  - $\mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_c\}$: $\alpha_i = decide\ \ \omega_i, 1 \leq i \leq c$
  -
  - Zero-one loss function

$$\lambda(\alpha_i|\omega_j) = \begin{cases} 0 & i = j \\ 1 & i \neq j \end{cases} \quad 1 \leq i, j \leq c$$

  - assign no loss (i.e. 0) to a correct decision
  - assign a unit loss (i.e. 1) to any incorrect decision
  - all errors are equally costly

Loss Functions (18)

- Conditional risk

$$\boldsymbol{R}(\alpha_i|x) = \sum_{j=1}^{c} \lambda(\alpha_i|\omega_j) P(\omega_j|x)$$

$$= \sum_{j \neq i} \lambda(\alpha_i|\omega_j) P(\omega_j|x) + \lambda(\alpha_i|\omega_i) P(\omega_i|x)$$

$$= \sum_{j \neq i} P(\omega_j|x) = 1 - P(\omega_i|x)$$

$$\mathbf{R}(\alpha_i|x) = 1 - P(\omega_i|x)$$

  - the conditional risk is error rate/probability that action α_i is wrong

- Minimum error rate classification:

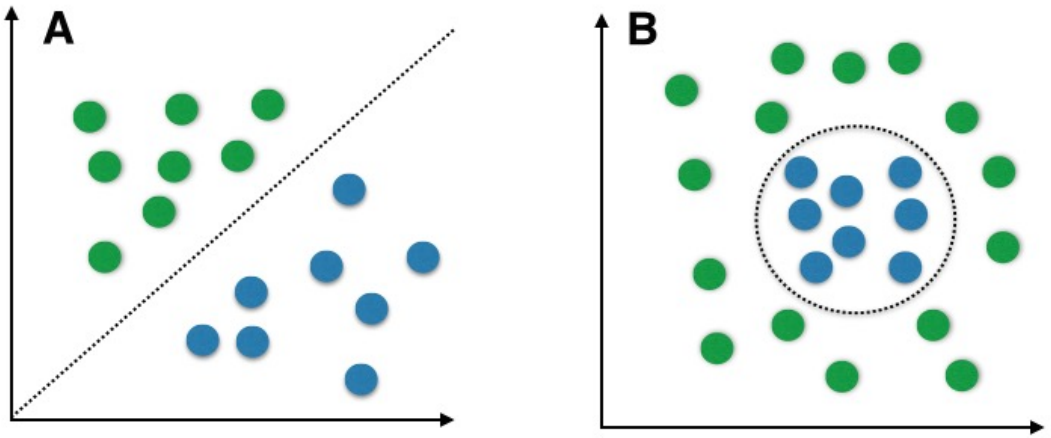
$$\text{Decide } \omega_i, \text{ if } P(\omega_i|x) > P(\omega_j|x) \text{ for all } j \neq i$$

Outline

- Bayes Decision Theory
- Classification Examples
- Loss Functions
- Discriminant Functions
- Bayesian Classification for Normal Distributions
- Conclusions

Discriminant Functions (1)

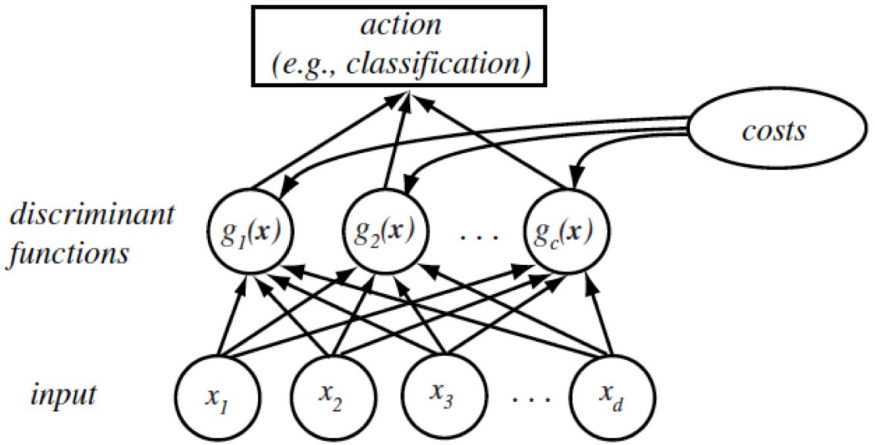
- If R_i and R_j are contiguous, define $g(x) \equiv P(\omega_i|x) - P(\omega_j|x) = 0$
  - $g(x) = 0$: the surface separating the regions is decision surface



  - $g(x) > 0$: region R_i where $P(\omega_i|x) > P(\omega_j|x)$
  - $g(x) < 0$: region R_j where $P(\omega_i|x) < P(\omega_j|x)$

Discriminant Functions (2)

- Classification
  - Pattern ⇛ Class
  - Actions ⇛ decide classes



- If $f(\cdot)$ monotonically increases, then

$$x \to \omega_i \text{ , if } f\left(P(\omega_i|x)\right) > f\left(P\left(\omega_j|x\right)\right) \quad \forall i \neq j$$

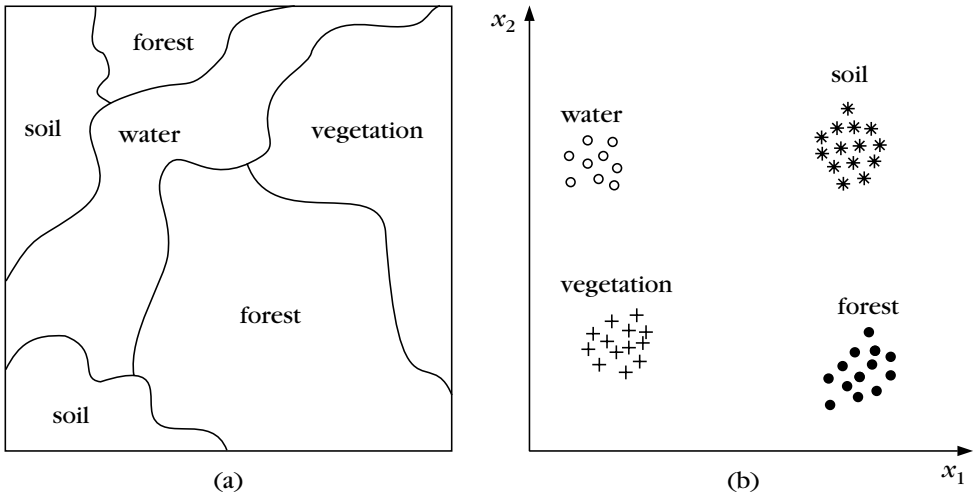
- $g_i(x) \equiv f\left(P(\omega_i|x)\right)$ is a discriminant function

Discriminant Functions (3)

- Discriminant functions

Decide ω_i, if $g_i(x) > g_j(x)$ for $\forall i \neq j$

- can be defined independent of the Bayes rule
- partition the feature space into c decision regions
- useful way to represent classifiers
- one function per class



Discriminant Functions (4)

- Minimum risk: $g_i(x) = -R(\alpha_i|x) \quad (1 \le i \le c)$
- Minimum-error-rate: $g_i(x) = P(\omega_i|x) \quad (1 \le i \le c)$
- various discriminant functions ⇛ identical classification results
- $f(\cdot)$ is a monotonically increasing functions ⇛

$$f\big(g_i(x)\big) \Leftrightarrow g_i(x)$$ (i.e. equivalent in decision)

- For example

$$f(x) = kx \ \ (k > 0) \quad \Leftrightarrow \quad f\big(g_i(x)\big) = kg_i(x) \ \ (1 \le i \le c)$$

$$f(x) = lnx \quad \Leftrightarrow \quad f\big(g_i(x)\big) = lng_i(x) \ \ (1 \le i \le c)$$

Discriminant Functions (5)

- Decision region
  - c discriminant functions $\quad\quad$ c decision regions

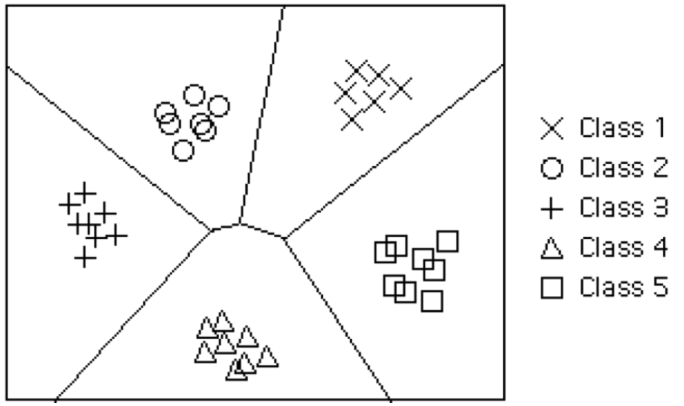
$$g_i(x) \quad (1 \le i \le c) \quad \Longrightarrow \quad R_i \subset \boldsymbol{R}^d \quad (1 \le i \le c)$$

$$R_i = \{x | x \in \boldsymbol{R}^d : g_i(x) > g_j(x) \ \forall j \neq i\}$$

where $R_i \cap R_j = \emptyset \ (j \neq i)$ and $\cup_{i=1}^{c} R_i = \boldsymbol{R}^d$

- Decision boundary

  surface in feature space where ties occur among several largest discriminant functions



Outline

- Bayes Decision Theory
- Classification Examples
- Loss Functions
- Discriminant Functions
- Bayesian Classification for Normal Distributions
- Conclusions

Bayesian Classification for Normal Distributions (1)

- Normal/Gaussian distribution: one-dimensional case

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad x \sim N(\mu, \sigma^2)$$

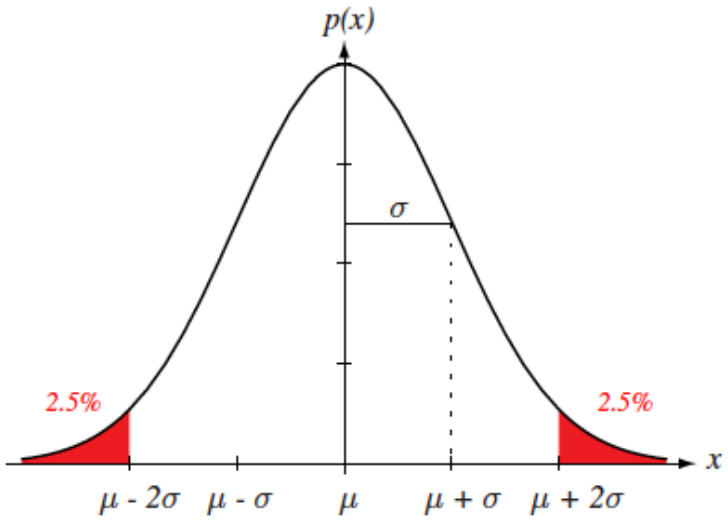
where

- μ is the mean value,

$$\mu = E[x] = \int_{-\infty}^{+\infty} x p(x) dx$$

- σ^2 is the variance,

$$\sigma^2 = E[(x - E[x])^2] = \int_{-\infty}^{+\infty} (x-\mu)^2 p(x) dx$$



Bayesian Classification for Normal Distributions (2)

- Normal distribution: Multivariate (multi-dimensional) case

$$X \sim N(\mu, \Sigma): \quad p(X) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma|^{\frac{1}{2}}} exp\left[-\frac{1}{2}(X-\mu)^T \Sigma^{-1}(X-\mu)\right]$$

  - d-dimensional random variables: $X = (x_1, x_2, \ldots, x_d)^T$
  - d-dimensional mean vector: $\mu = (\mu_1, \mu_2, \ldots, \mu_d)^T$ where $\mu_i = E[x_i]$
  - $d \times d$ covariance matrix:

$$\Sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_{dd} \end{pmatrix}$$

$\sigma_{ij} = \sigma_{ji} = E[(x_i - \mu_i)(x_j - \mu_j)]$

Σ : Symmetric, positive semidefinite

$|\Sigma|$: determinant

Σ^{-1}: inverse

Bayesian Classification for Normal Distributions (3)

- $(X-\mu)^T$: $1\times d$ matrix;
- Σ^{-1}: $\mathrm{d}\times d$ matrix;
- $(X-\mu)$: $\mathrm{d}\times 1$ matrix

$\Rrightarrow (X-\mu)^T\Sigma^{-1}(X-\mu)$: Scalar (1×1 matrix)

- Σ : positive definite; $\Rrightarrow$ Σ^{-1}: positive definite

$\Rrightarrow \quad (X-\mu)^T\Sigma^{-1}(X-\mu) \geq 0$

$\Rrightarrow \quad -\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu) \leq 0$

Bayesian Classification for Normal Distributions (4)

- Example: two-dimensional case

$X \sim N(\mu, \Sigma)$: $$p(X) = \frac{1}{2\pi|\Sigma|^{\frac{1}{2}}} exp\left[-\frac{1}{2}(x_1 - \mu_1, x_2 - \mu_2)\Sigma^{-1}\begin{pmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{pmatrix}\right]$$

where

$$\mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} = \begin{pmatrix} E[x_1] \\ E[x_2] \end{pmatrix}, \quad \Sigma = E\left[\begin{pmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{pmatrix}(x_1 - \mu_1, x_2 - \mu_2)\right] = \begin{pmatrix} \sigma_1^2 & \sigma \\ \sigma & \sigma_2^2 \end{pmatrix},$$

and $\sigma = E[(x_1 - \mu_1)(x_2 - \mu_2)]$

Bayesian Classification for Normal Distributions (5)

- Minimum-error-rate classification:

$$g_i(X) = P(\omega_i|X) \qquad 1 \le i \le c$$

$$g_i(X) = P(\omega_i|X) \quad \Rightarrow \quad g_i(X) = lnP(\omega_i|X) = ln\big(p(X|\omega_i)P(\omega_i)\big)$$

$$\Rightarrow \quad g_i(X) = lnp(X|\omega_i) + \ln P(\omega_i)$$

constant can be ignored

$$p(X|\omega_i) \sim N(\mu_i, \Sigma_i)$$

$$g_i(X) = -\frac{1}{2}(X-\mu_i)^T\Sigma_i^{-1}(X-\mu_i) - \frac{d}{2}ln2\pi - \frac{1}{2}ln|\Sigma_i| + \ln P(\omega_i)$$

Bayesian Classification for Normal Distributions (6)

- Case I: $\Sigma_i = \sigma^2 I$

Covariance matrix: $\Sigma_i = \sigma^2 I \quad \Rightarrow \quad |\Sigma_i| = \sigma^{2d}$ and $\quad \Sigma_i^{-1} = \frac{1}{\sigma^2} I$

$$g_i(X) = -\frac{\|X - \mu_i\|^2}{2\sigma^2} + \ln P(\omega_i)$$

where $\|\cdot\|$ is Euclidean distance; $\|X - \mu_i\|^2 = (X - \mu_i)^T (X - \mu_i)$

then $g_i(X) = -\frac{1}{2\sigma^2}\left(X^T X - 2\mu_i^T X + \mu_i^T \mu_i\right) + \ln P(\omega_i)$

where $X^T X$ is the same for any class (state of nature), can be ignored.

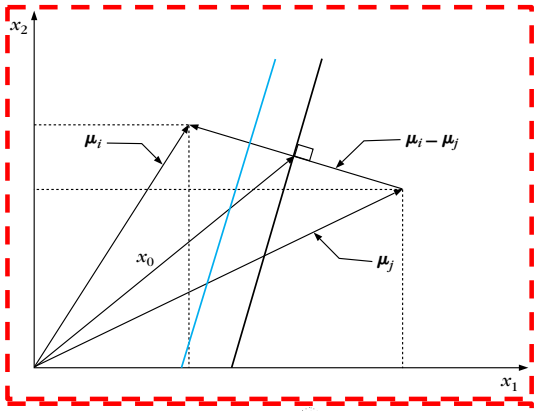
Bayesian Classification for Normal Distributions (7)

- **Linear Discriminant Functions**: If $P(\omega_1) = P(\omega_2)$, then $x_0 = \frac{1}{2}(\mu_i + \mu_j)$

$$g_i(X) = w_i^T X + w_{i0}$$

where $w_i = \frac{1}{\sigma^2}\mu_i$ Weight vector

$w_{i0} = -\frac{1}{2\sigma^2}\mu_i^T\mu_i + \ln P(\omega_i)$ Threshold or Bias

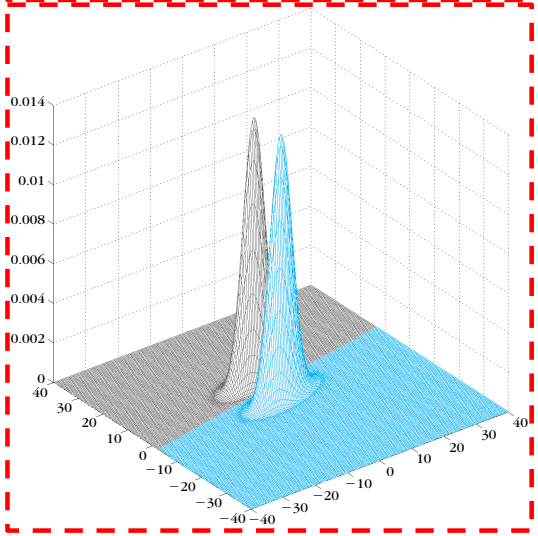


- **Decision surface**:

$$g_i(x) - g_j(x) = \frac{1}{\sigma^2}\mu_i x + w_{i0} - \left(\frac{1}{\sigma^2}\mu_j x + w_{j0}\right) = 0$$

then $w^T(x - x_0) = 0$

where $w = \mu_i - \mu_j$ and $x_0 = \frac{1}{2}(\mu_i + \mu_j) - \sigma^2 ln\frac{P(\omega_i)}{P(\omega_j)}\frac{\mu_i - \mu_j}{\|\mu_i - \mu_j\|^2}$



Bayesian Classification for Normal Distributions (8)

- Case II: $\Sigma_i = \Sigma$

$$p(X|\omega_i) \sim N(\mu_i, \Sigma)$$

$$g_i(X) = -\frac{1}{2}(X-\mu_i)^T\Sigma^{-1}(X-\mu_i) - \frac{1}{2}ln|\Sigma| + \ln P(\omega_i)$$

Covariance matrix Σ is identical for all classes, term $\frac{1}{2}ln|\Sigma|$ can be ignored.

then $g_i(X) = -\frac{1}{2}(X-\mu_i)^T\Sigma^{-1}(X-\mu_i) + \ln P(\omega_i)$

where $(X-\mu_i)^T\Sigma^{-1}(X-\mu_i)$ is squared Mahalanobis distance

when $\Sigma = I$, it reduces to Euclidean distance

Bayesian Classification for Normal Distributions (9)

$$g_i(X) = -\frac{1}{2}(X-\mu_i)^T\Sigma^{-1}(X-\mu_i) + \ln P(\omega_i) \quad \Rrightarrow$$

$$g_i(X) = -\frac{1}{2}\left(X^T\Sigma^{-1}X - 2\mu_i^T\Sigma^{-1}X + \mu_i^T\Sigma^{-1}\mu_i\right) + \ln P(\omega_i)$$

where $X^T\Sigma^{-1}X$ is the same for all classes, it then can be ignored.

- Linear Discriminant Functions:

$$g_i(X) = w_i^T X + w_{i0}$$

where

$w_i = \mu_i$ Weight vector

$w_{i0} = -\frac{1}{2}\mu_i^T\mu_i + \ln P(\omega_i)$ Threshold or Bias

Bayesian Classification for Normal Distributions (10)

- Case III: $\Sigma_i = \text{arbitrary}$

$$p(X|\omega_i) \sim N(\mu_i, \Sigma_i)$$

$$g_i(X) = -\frac{1}{2}(X-\mu_i)^T\Sigma_i^{-1}(X-\mu_i) - \frac{1}{2}ln|\Sigma_i| + \ln P(\omega_i)$$

- Quadratic Discriminant Functions:

$$g_i(X) = X^T W_i X + w_i^T X + w_{i0}$$

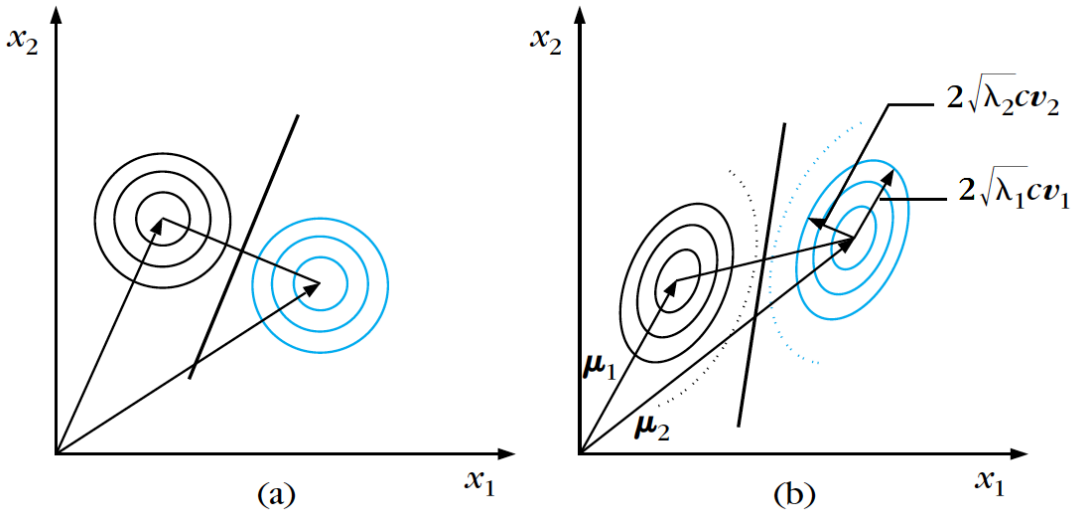
where $W_i = -\frac{1}{2}\Sigma_i^{-1}$ Quadratic matrix

$w_i = \Sigma_i^{-1}\mu_i$ Weight vector

$w_{i0} = -\frac{1}{2}\mu_i^T\Sigma_i^{-1}\mu_i - \frac{1}{2}ln|\Sigma_i| + \ln P(\omega_i)$ Threshold or Bias

Bayesian Classification for Normal Distributions (11)

- Minimum Distance Classifiers
  - $P(\omega_i) = \frac{1}{c}$, equiprobable
  - $g_i(x) = -\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)$



- $\Sigma_i = \sigma^2 I$:

  Assign $x \to \omega_i$, small Euclidean Distance: $d_E \equiv \|x-\mu_i\|$

- $\Sigma_i = \Sigma$:

  Assign $x \to \omega_i$, small Mahalanobis Distance: $d_m \equiv \sqrt{(x-\mu_i)^T\Sigma^{-1}(x-\mu_i)}$

Bayesian Classification for Normal Distributions (12)

- Example:

  Given ω_1, ω_2: $P(\omega_1) = P(\omega_2)$

  $p(x|\omega_1) \sim N(\mu_1, \Sigma)$, $p(x|\omega_2) \sim N(\mu_2, \Sigma)$,

  and $\mu_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\mu_2 = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$, $\Sigma = \begin{pmatrix} 1.1 & 0.3 \\ 0.3 & 1.9 \end{pmatrix}$.

  Classify the vector $x = \begin{pmatrix} 1.0 \\ 2.2 \end{pmatrix}$ using Bayesian classification

- $\Sigma^{-1} = \begin{pmatrix} 0.95 & -0.15 \\ -0.15 & 0.55 \end{pmatrix}$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

determinant

Bayesian Classification for Normal Distributions (13)

- Compute Mahalanobis distance d_m using μ_1, μ_2,i.e.

$$d_{m,i}^2 = (x - \mu_i)^T \Sigma^{-1} (x - \mu_i)$$

$$d_{m,1}^2 = (1.0, 2.2) \begin{pmatrix} 0.95 & -0.15 \\ -0.15 & 0.55 \end{pmatrix} \begin{pmatrix} 1.0 \\ 2.2 \end{pmatrix} = 2.952$$

$$d_{m,2}^2 = (-2.0, -0.8) \begin{pmatrix} 0.95 & -0.15 \\ -0.15 & 0.55 \end{pmatrix} \begin{pmatrix} -2.0 \\ -0.8 \end{pmatrix} = 3.672$$

- Classify $x \rightarrow \omega_1$, observing $d_{m,1}^2 < d_{m,2}^2$

Conclusions

- Bayes decision theory
- several classification examples
- loss functions to evaluation the classification performance
- discriminant functions for several cases
- Bayesian classification for several normal distributions

Estimation Methods

CISC3024 – Pattern Recognition

Prof. Yicong Zhou

yicongzhou@um.edu.mo

@Fall 2024

Outline

- l-norms and Distance Metrics
- Parameter Estimation
- Maximum Likelihood Estimation
- Bayesian Estimation
- Conclusion

l-norms and Distance Metrics (1)

X is a column vector in $\mathbb{R}^N$ space, $X = [x_1, x_2, \dots, x_N]^T$

- l_0-norm

$$\|X\|_0 \equiv \sum_{i=1}^{N} |x_i|^0 = |x_1|^0 + |x_2|^0 + \cdots + |x_N|^0$$

  - where $0^0 = 0$
  - l_0-norm is the number of non-zero elements in vector X

l-norms and Distance Metrics (2)

- l_1-norm

$$\|X\|_1 \equiv \sum_{i=1}^{N} |x_i| = |x_1| + |x_2| + \cdots + |x_N|$$

  - l_1-norm is the sum of the absolute values of vector X
  - l_1-norm is also called Taxicab norm or Manhattan norm

l-norms and Distance Metrics (3)

- l_2-norm

$$\|X\|_2 \equiv \left(\sum_{i=1}^{N} x_i^2\right)^{\frac{1}{2}} = \sqrt{x_1^2 + x_2^2 + \cdots + x_N^2}$$

  - l_2-norm can be expressed as matrix format $\|X\|_2 \equiv \sqrt{X^T X}$
  - l_2-norm is also called Euclidean norm

l-norms and Distance Metrics (4)

- l_∞-norm

$$\|X\|_\infty \equiv \max(|x_1|, |x_2|, \ldots, |x_N|)$$

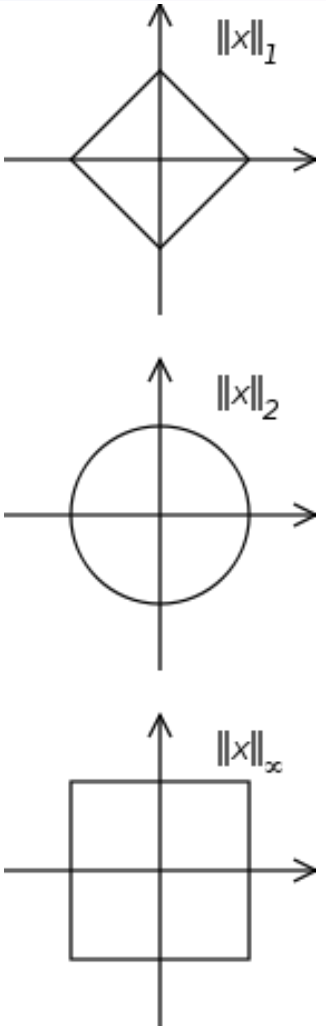
  - l_∞-norm takes the maximum of the absolute values of elements in vector X
  - l_∞-norm is also called maximum norm

l-norms and Distance Metrics (5)

- l_p-norm

$$\|X\|_p \equiv \left(\sum_{i=1}^{N} |x_i|^p\right)^{\frac{1}{p}} = (|x_1|^p + |x_2|^p + \cdots + |x_N|^p)^{\frac{1}{p}}$$

  - l_p−norm is a general form of l−norm
  - $p \geq 0$
  - $p = 0$, l_0-norm
  - $p = 1$, l_1-norm
  - $p = 2$, l_2-norm
  - $p \to \infty$, l_∞-norm



l-norms and Distance Metrics (6)

- **Euclidean Distance**

For two data sets $X = [x_1, x_2, \ldots, x_N]^T$ and $Y = [y_1, y_2, \ldots, y_N]^T \in \mathbb{R}^N$

$$d(X,Y) = \sqrt{\sum_{i=1}^{N}(x_i - y_i)^2} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_N - y_N)^2}$$

- Euclidean distance is the straight-line distance between X and Y.
- Euclidean distance is also called L_2 distance. $d(X,Y) = \sqrt{(X-Y)^T(X-Y)}$

l-norms and Distance Metrics (7)

- **Mahalanobis Distance**

For an observation $X = [x_1, x_2, \ldots, x_N]^T$ and a set of observations with mean $\mu = [\mu, \mu_2, \ldots, \mu_N]^T$ and covariance matrix Σ

$$D_M(X) = \sqrt{(X - \mu)^T \Sigma^{-1} (X - \mu)}$$

- It is a measure of the distance between a point and a distribution.
- It reverts to Euclidean distance, when $\Sigma = I$

Outline

- l-norms and Distance Metrics
- Parameter Estimation
- Maximum Likelihood Estimation
- Bayesian Estimation
- Conclusion

Parameter Estimation (1)

$$P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{p(x)} \quad 1 \le j \le c \quad \text{(Bayes Formula)}$$

- Likelihood $p(x|\omega_j)$ and Prior probability $P(\omega_j)$
  - are unknown
  - are estimated from training samples (supervised learning)
- Collect training samples $\mathrm{X} = \{x_1, x_2, \ldots, x_N\}$ distributed according to $p(x|\omega_j)$
- $x_1, x_2, \ldots, x_N$ are $i.i.d.$ (independent and identically distributed)

Parameter Estimation (2)

- For prior probability $P(\omega_j)$

$$P(\omega_j) = \frac{|x_j|}{\sum_{i=1}^{c} |x_i|}$$

where $|x|$ is the number of elements in x

- For class-conditional PDF $p(x|\omega_j)$
  - **Case I**: $p(x|\omega_j)$ has certain parametric form

    e.g. we assume $p(x|\omega_j)$ is a normal density with unknown parameters θ_j, i.e.

    $p(x|\omega_j) \sim N(\mu_j, \Sigma_j)$, parameters: $\theta_j = \{\mu_j, \Sigma_j\}$

    $p(x|\omega_j) \Rightarrow p(x|\omega_j, \theta_j)$
  - **Case II**: $p(x|\omega_j)$ has NO parametric form

Parameter Estimation (3)

- Estimation under parametric form: $p(x|\omega_j, \theta_j) \quad (1 \leq j \leq c)$
  - Maximum-likelihood (ML) estimation
    - consider parameters as fixed but unknown quantities
    - estimate parameters by maximizing likelihood of observing actual training examples
  - Bayesian estimation
    - consider parameters as random variables with some known prior distribution
    - Observing the actual training examples transforms the parameters' prior distribution into posterior distribution using Bayes formula

Outline

- l-norms and Distance Metrics
- Parameter Estimation
- Maximum Likelihood Estimation
- Bayesian Estimation
- Conclusion

Maximum Likelihood Estimation (1)

- Let $x_k \sim p(x|\theta),\ k = 1, 2, \ldots, N$
  - θ: parameters to be estimated
  - $\mathrm{X} = \{x_1, x_2, \ldots, x_N\}$: a set of $i.i.d.$ examples
- Objective function

$$p(X|\theta) \equiv p(x_1, x_2, \ldots, x_N|\theta) = \prod_{k=1}^{N} p(x_k|\theta)$$

  - $p(X|\theta)$ is the Likelihood of θ $w.r.t.$ X

Maximum Likelihood Estimation (2)

- **Maximum Likelihood Estimation:**

$$\hat{\theta}_{ML} = \underset{\theta}{\operatorname{argmax}}\, p(X|\theta) = \underset{\theta}{\operatorname{argmax}} \prod_{k=1}^{N} p(x_k|\theta)$$

  - $\hat{\theta}$ best agrees with the observed examples

- **log-likelihood function:**

$$L(\theta) = \ln p(X|\theta) = \sum_{k=1}^{N} \ln p(x_k|\theta)$$

$$\hat{\theta}_{ML} = \underset{\theta}{\operatorname{argmax}}\, p(X|\theta) \quad \Leftrightarrow \hat{\theta}_{ML} = \underset{\theta}{\operatorname{argmax}}\, L(\theta) = \underset{\theta}{\operatorname{argmax}} \sum_{k=1}^{N} \ln p(x_k|\theta)$$

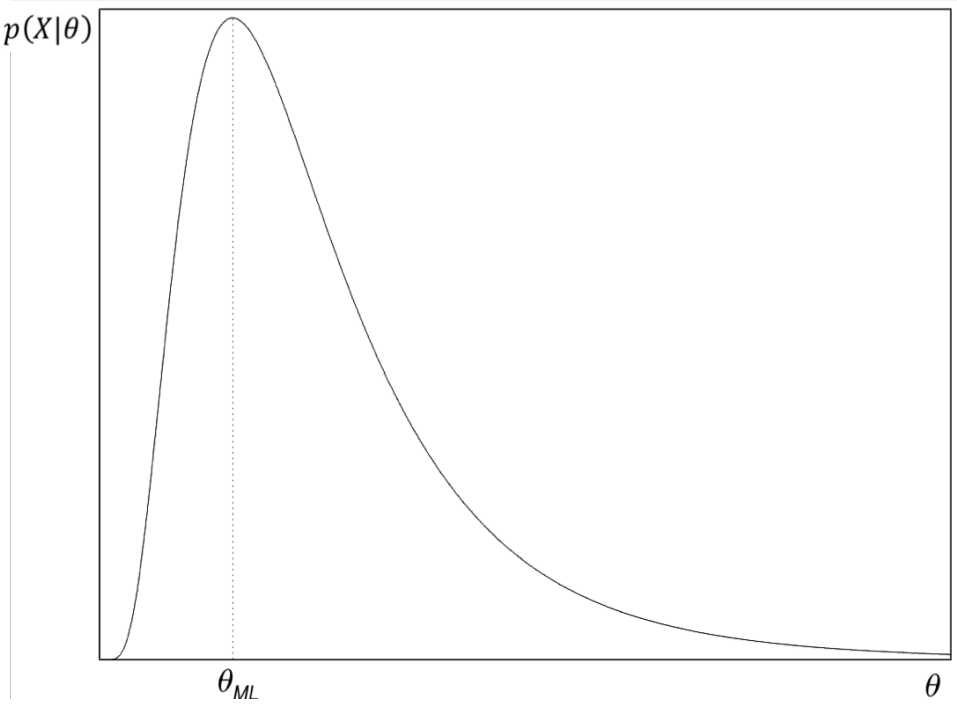
Maximum Likelihood Estimation (3)

- ML estimation of $\hat{\theta}$

$\hat{\theta}_{ML}$: $\dfrac{\partial L(\theta)}{\partial \theta} = 0$

Namely,

$\hat{\theta}_{ML}$: $\displaystyle\sum_{k=1}^{N} \frac{\partial \ln p(x_k|\theta)}{\partial \theta} = 0$



Maximum Likelihood Estimation (5)

- **Case I**: $x_k \sim N(\mu, \Sigma)$ where μ is unknown $\Rightarrow \theta = \{\mu\}$

$$p(x_k|\mu) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} exp\left[-\frac{1}{2}(x_k - \mu)^T\Sigma^{-1}(x_k - \mu)\right]$$

$$\Rightarrow \ln p(x_k|\mu) = -\frac{1}{2}\ln[(2\pi)^d|\Sigma|] - \frac{1}{2}(x_k - \mu)^T\Sigma^{-1}(x_k - \mu)$$

$$= -\frac{1}{2}\ln[(2\pi)^d|\Sigma|] - \frac{1}{2}x_k^T\Sigma^{-1}x_k + \mu^T\Sigma^{-1}x_k - \frac{1}{2}\mu^T\Sigma^{-1}\mu$$

$$\Rightarrow \frac{\partial \ln p(x_k|\mu)}{\partial \mu} = \Sigma^{-1}(x_k - \mu)$$

Maximum Likelihood Estimation (6)

- ML estimation of $\hat{\mu}$:

$$\hat{\mu}_{ML}: \frac{\partial L(\mu)}{\partial \mu} = \sum_{k=1}^{N} \frac{\partial \ln p(x_k|\mu)}{\partial \mu} = 0$$

$$\Rightarrow \quad \sum_{k=1}^{N} \Sigma^{-1}(x_k - \hat{\mu}) = 0 \quad \Rightarrow \quad \sum_{k=1}^{N} (x_k - \hat{\mu}) = 0$$

$$\Rightarrow \quad \hat{\mu} = \frac{1}{N} \sum_{k=1}^{N} x_k$$

Maximum Likelihood Estimation (7)

- **Case II**: $x_k \sim N(\mu, \Sigma)$ where μ and Σ are unknown $\Rightarrow \theta = \{\mu, \Sigma\}$
- For univariate case:

$$p(x_k|\theta) = \frac{1}{\sqrt{2\pi}\sigma} exp\left[-\frac{(x_k-\mu)^2}{2\sigma^2}\right] = \frac{1}{\sqrt{2\pi\theta_2}} exp\left[-\frac{(x_k-\theta_1)^2}{2\theta_2}\right] \text{ where } \theta = \begin{bmatrix}\theta_1 \\ \theta_2\end{bmatrix} = \begin{bmatrix}\mu \\ \sigma^2\end{bmatrix}$$

$$\Rightarrow \ln p(x_k|\theta) = -\frac{1}{2}\ln(2\pi\theta_2) - \frac{1}{2\theta_2}(x_k - \theta_1)^2$$

$$\Rightarrow \frac{\partial \ln p(x_k|\theta)}{\partial \theta} = \begin{bmatrix} \frac{(x_k - \theta_1)}{\theta_2} \\ -\frac{1}{2\theta_2} + \frac{(x_k - \theta_1)^2}{2\theta_2^2} \end{bmatrix}$$

Maximum Likelihood Estimation (8)

- **ML estimation** of $\hat{\mu}$ and $\hat{\sigma}^2$:

$$\frac{\partial L(\theta)}{\partial \theta} = \sum_{k=1}^{N} \frac{\partial \ln p(x_k|\theta)}{\partial \theta} = \begin{bmatrix} \sum_{k=1}^{N} \frac{(x_k - \theta_1)}{\theta_2} \\ \sum_{k=1}^{N} \left(-\frac{1}{2\theta_2} + \frac{(x_k - \theta_1)^2}{2\theta_2^2} \right) \end{bmatrix} = 0$$

Namely,

$$\sum_{k=1}^{N} \frac{(x_k - \hat{\theta}_1)}{\hat{\theta}_2} = 0 \quad \Rightarrow \quad \sum_{k=1}^{N} (x_k - \hat{\theta}_1) = 0 \quad \Rightarrow \quad \hat{\theta}_1 = \frac{1}{N} \sum_{k=1}^{N} x_k$$

$$\sum_{k=1}^{N} \left(-\frac{1}{2\hat{\theta}_2} + \frac{(x_k - \hat{\theta}_1)^2}{2\hat{\theta}_2^2} \right) = 0 \quad \Rightarrow \quad \hat{\theta}_2 = \frac{1}{N} \sum_{k=1}^{N} (x_k - \hat{\theta}_1)^2$$

$$\Rightarrow \quad \hat{\mu} = \frac{1}{N} \sum_{k=1}^{N} x_k \qquad \hat{\sigma}^2 = \frac{1}{N} \sum_{k=1}^{N} (x_k - \hat{\mu})^2$$

Maximum Likelihood Estimation (9)

- For multivariate case:

$$p(x_k|\theta) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} exp\left[-\frac{1}{2}(x_k-\mu)^T\Sigma^{-1}(x_k-\mu)\right] \text{ where } \theta = \begin{bmatrix}\mu \\ \Sigma\end{bmatrix}$$

$$\hat{\mu} = \frac{1}{N}\sum_{k=1}^{N} x_k \quad \Rrightarrow$$

Arithmetic average of n vectors x_k

$$\hat{\Sigma} = \frac{1}{N}\sum_{k=1}^{N}(x_k-\hat{\mu})(x_k-\hat{\mu})^T \quad \Rrightarrow$$

arithmetic average of n matrics $(x_k-\hat{\mu})(x_k-\hat{\mu})^T$

Outline

- l-norms and Distance Metrics
- Parameter Estimation
- Maximum Likelihood Estimation
- Bayesian Estimation
- Conclusion

Bayesian Estimation (1)

- In ML estimation, θ was considered as a parameter with a fixed value
- Here, we look θ as an unknown random vector, described by a PDF $p(\theta)$

- Given $\mathrm{X} = \{x_1, x_2, \ldots, x_N\}$, we compute the maximum of $p(\theta|X)$
- From Bayes formula,

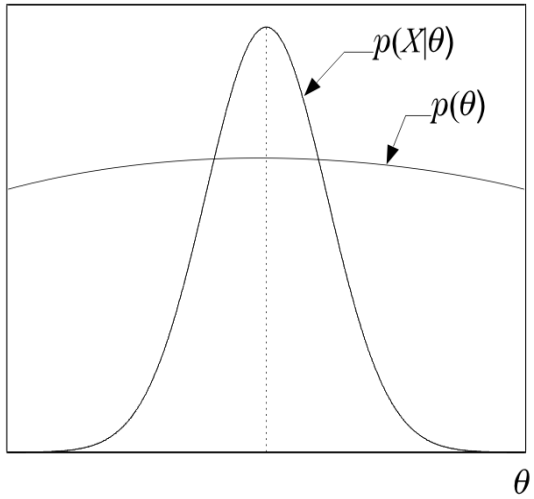
$$p(\theta|X) = \frac{p(X|\theta)p(\theta)}{p(X)}$$

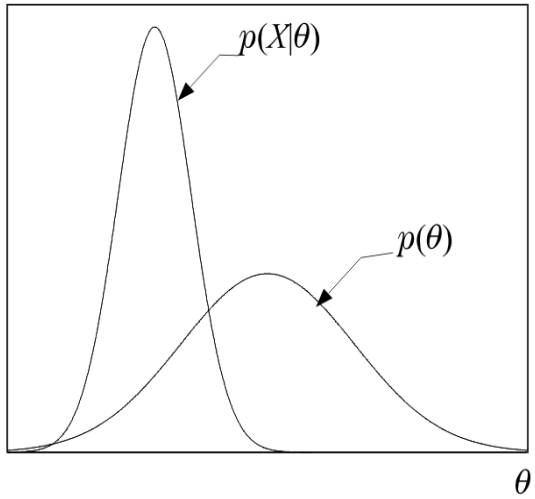
Bayesian Estimation (2)

- **Bayesian Estimation**:
  - also Maximum Aposterior Probability (MAP) Estimation

$$\hat{\theta}_{MAP} = \underset{\theta}{\operatorname{argmax}}\, p(\theta|X) = \underset{\theta}{\operatorname{argmax}}\, p(X|\theta)p(\theta)$$

or $\hat{\theta}_{MAP}$: $\frac{\partial}{\partial\theta}\left(p(X|\theta)p(\theta)\right) = 0 \quad \Rightarrow\!\!\!\Rightarrow \quad \frac{\partial}{\partial\theta}\ln\left(p(X|\theta)p(\theta)\right) = 0$





If $p(\theta)$ is uniform or broad enough,

$$\hat{\theta}_{MAP} \cong \hat{\theta}_{ML}$$

Bayesian Estimation (3)

- **Example**: $p(x_k|\mu) \sim N(\mu, \sigma^2)$, and μ is unknown with $p(\mu) \sim N(\mu_0, \sigma_0^2)$

$$p(x_k|\mu) = \frac{1}{\sqrt{2\pi}\sigma} exp\left[-\frac{(x_k-\mu)^2}{2\sigma^2}\right] \qquad p(\mu) = \frac{1}{\sqrt{2\pi}\sigma_0} exp\left[-\frac{(\mu-\mu_0)^2}{2\sigma_0^2}\right]$$

$\hat{\mu}_{MAP}$: $$\frac{\partial}{\partial\mu} \ln\big(p(X|\mu)p(\mu)\big) = \frac{\partial}{\partial\mu} \ln\left(\prod_{k=1}^{N} p(x_k|\mu)\, p(\mu)\right) = 0$$

$$\Rightarrow \quad \sum_{k=1}^{N} \frac{x_k - \hat{\mu}}{\sigma^2} - \frac{\hat{\mu} - \mu_0}{\sigma_0^2} = 0 \qquad \Rightarrow \qquad \hat{\mu}_{MAP} = \frac{\mu_0 + \frac{\sigma_0^2}{\sigma^2}\sum_{k=1}^{N} x_k}{1 + \frac{\sigma_0^2}{\sigma^2} N}$$

For $\frac{\sigma_0^2}{\sigma^2} \gg 1$ or $N \to \infty$, $\hat{\mu}_{MAP} \cong \hat{\mu}_{ML} = \frac{1}{N}\sum_{k=1}^{N} x_k$

Conclusion

- l-norms and Distance Metrics
- Maximum likelihood estimation
  - parameters are fixed but unknown values
  - objective function: Log-likelihood function
  - Necessary conditions: gradient of the objective function is zero
- Bayesian Estimation
  - parameters are random variables
  - form the objective function using Bayes formula

Dimension Reduction

CISC3024 – Pattern Recognition

Prof. Yicong Zhou

yicongzhou@um.edu.mo

@Fall 2024

Outline

- Dimension Reduction
- Singular Value Decomposition (SVD)
- Principle Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Conclusion

Dimension Reduction (1)

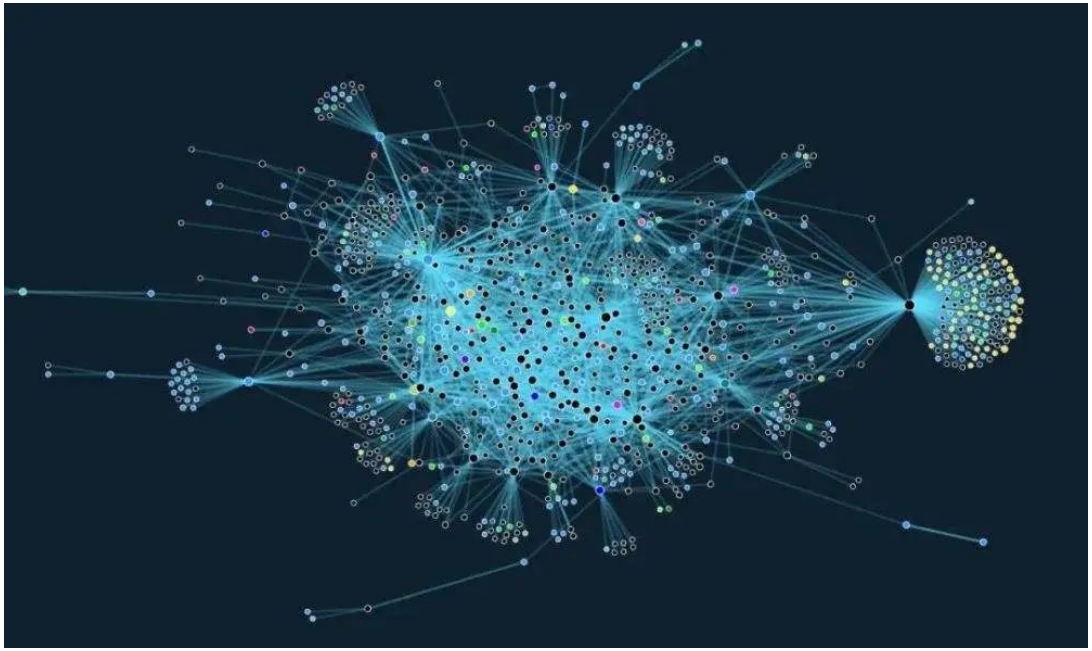
- Dimension Reduction
  - changes the data representation into a low-dimensional one
  - preserves the data structure
  - is usually unsupervised

Dimension Reduction (2)

- Why Dimension Reduction
  - Computation complexity
    - The computation complexity grows exponentially with the data dimension
  - Pre-processing stage before further learning
  - Data visualization:
    - Projection of high-dimensional data to 2D or 3D
    - 2D/3D plots of data make nice pictures
  - Data interpretation:
    - recovering the intrinsic dimensionality of the data
    - Some data features may be irrelevant

Dimension Reduction (3)

- Applications
  - Pattern recognition and analysis
  - Digital image and speech processing
  - Gene expression microarray data
  - Visualization of large networks



Dimension Reduction (4)

- Methods
  - Principal component analysis (PCA)
  - Linear discriminant analysis (LDA)
  - Locally linear embedding (LLE)
  - Independent component analysis (ICA)
  - Canonical correlation analysis (CCA)
  - …

Outline

- Dimension Reduction
- Singular Value Decomposition (SVD)
- Principle Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Conclusion

SVD (1)

- **SVD** (Singular Value Decomposition) definition

$$A_{m\times n} = U_{m\times m} S_{m\times n} V^T_{n\times n}$$

A is any $m\times n$ matrix

U is any $m\times m$ orthogonal matrix $UU^T = U^T U = \mathrm{I}$, $U^T = U^{-1}$

S is any $m\times n$ diagonal matrix

V is any $n\times n$ orthogonal matrix

SVD (2)

- Calculate AA^T and A^TA

$$AA^T = USV^T(USV^T)^T = USV^TVS^TU^T = U(SS^T)U^T$$

$$A^TA = (USV^T)^TUSV^T = VS^TU^TUSV^T = V(SS^T)V^T$$

$$AV = USV^TV = US$$

- Singular values $\sigma_1 > \sigma_2 > \cdots > \sigma_n > 0$ appear in descending order along the main diagonal of S
- $\sigma_1^2 > \sigma_2^2 > \cdots > \sigma_n^2$ are the eigenvalues of AA^T and A^TA

SVD (3)

- Calculation procedures of SVD
  - Step 1: Calculate AA^T and A^TA
  - Step 2: Eigenvalues and S
  - Step 3: Finding U
  - Step 4: Finding V
  - Step 5: Complete SVD

SVD (4)

- Step 1: Calculate AA^T and A^TA

Let $A = \begin{pmatrix} 4 & 0 \\ 3 & -5 \end{pmatrix}$, then $A^T = \begin{pmatrix} 4 & 3 \\ 0 & -5 \end{pmatrix}$

$$AA^T = \begin{pmatrix} 4 & 0 \\ 3 & -5 \end{pmatrix}\begin{pmatrix} 4 & 3 \\ 0 & -5 \end{pmatrix} = \begin{pmatrix} 16 & 12 \\ 12 & 34 \end{pmatrix}$$

$$A^TA = \begin{pmatrix} 4 & 3 \\ 0 & -5 \end{pmatrix}\begin{pmatrix} 4 & 0 \\ 3 & -5 \end{pmatrix} = \begin{pmatrix} 25 & -15 \\ -15 & 25 \end{pmatrix}$$

SVD (5)

- Step 2: Eigenvalues and S

$$|AA^T - \lambda I| = 0 \quad \Rightarrow \quad \left|\begin{pmatrix} 16 & 12 \\ 12 & 34 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right| = 0 \quad \Rightarrow$$

$$\begin{vmatrix} 16-\lambda & 12 \\ 12 & 34-\lambda \end{vmatrix} = 0 \quad \Rightarrow \quad (16-\lambda)(34-\lambda) - 12\times 12 = 0$$

Characteristic equation:

$$\lambda^2 - 50\lambda + 400 = 0 \quad \Rightarrow (\lambda - 40)(\lambda - 10) = 0$$

Eigenvalues: $\lambda_1 = 40, \quad \lambda_2 = 10$

SVD (6)

- Singular values:

$\sigma_1 = \sqrt{\lambda_1} = \sqrt{40} = 2\sqrt{10}$, $\sigma_2 = \sqrt{\lambda_2} = \sqrt{10}$

- Diagonal matrix S :

$$S = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} = \begin{pmatrix} 2\sqrt{10} & 0 \\ 0 & \sqrt{10} \end{pmatrix}$$

SVD (7)

- Step 3: Finding U

$(AA^T - \lambda I)x = 0$

For $\lambda_1 = 40$,

$$(AA^T - \lambda_1 I)x_1 = 0 \quad \Rrightarrow \quad \left(\begin{pmatrix} 16 & 12 \\ 12 & 34 \end{pmatrix} - 40\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right)x_1 = 0 \quad \Rrightarrow$$

$$\begin{pmatrix} -24 & 12 \\ 12 & -6 \end{pmatrix}x_1 = 0 \quad \Rrightarrow \quad x_1 = \begin{pmatrix} a \\ 2a \end{pmatrix} \quad \Rrightarrow \quad u_1 = \frac{x_1}{\|x_1\|} = \begin{pmatrix} \frac{1}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} \end{pmatrix}$$

SVD (8)

For $\lambda_2 = 10$,

$$(AA^T - \lambda_2 I)x_2 = 0 \quad \Rrightarrow \quad \left(\begin{pmatrix} 16 & 12 \\ 12 & 34 \end{pmatrix} - 10\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right) x_2 = 0$$

$$\Rrightarrow \begin{pmatrix} 6 & 12 \\ 12 & 24 \end{pmatrix} x_2 = 0 \quad \Rrightarrow \quad x_2 = \begin{pmatrix} 2\mathrm{b} \\ -\mathrm{b} \end{pmatrix} \quad \Rrightarrow$$

$$u_2 = \frac{x_2}{\|x_2\|} = \begin{pmatrix} \frac{2}{\sqrt{5}} \\ -\frac{1}{\sqrt{5}} \end{pmatrix} \quad \Rrightarrow \quad U = (u_1, u_2) = \begin{pmatrix} \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} \end{pmatrix}$$

SVD (9)

- Step 4: Finding V

Similarly, $(A^T\mathrm{A} - \lambda I)x = 0$

For $\lambda_1 = 40$,

$$(A^TA - \lambda_1 I)x_3 = 0 \quad \Rrightarrow \quad \left(\begin{pmatrix} 25 & -15 \\ -15 & 25 \end{pmatrix} - 40\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right)x_3 = 0 \quad \Rrightarrow$$

$$\begin{pmatrix} -15 & -15 \\ -15 & -15 \end{pmatrix}x_3 = 0 \quad \Rrightarrow \quad x_3 = \begin{pmatrix} c \\ -c \end{pmatrix} \quad \Rrightarrow \quad v_1 = \frac{x_3}{\|x_3\|} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix}$$

SVD (10)

For $\lambda_2 = 10$,

$$(A^T A - \lambda_2 I)x_4 = 0 \quad \Rightarrow \left(\begin{pmatrix} 25 & -15 \\ -15 & 25 \end{pmatrix} - 10 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right) x_4 = 0$$

$$\Rightarrow \begin{pmatrix} 15 & -15 \\ -15 & 15 \end{pmatrix} x_4 = 0 \quad \Rightarrow \quad x_4 = \begin{pmatrix} d \\ d \end{pmatrix}$$

$$\Rightarrow \quad v_2 = \frac{x_4}{\|x_4\|} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \quad \Rightarrow \quad V = (v_1, v_2) = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$$

SVD (11)

- Step 5: Complete SVD

$$A_{m\times n} = U_{m\times m} S_{m\times n} V^T_{n\times n}$$

$$\begin{pmatrix} 4 & 0 \\ 3 & -5 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} \end{pmatrix} \begin{pmatrix} 2\sqrt{10} & 0 \\ 0 & \sqrt{10} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$$

SVD (12)

- Exercise

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

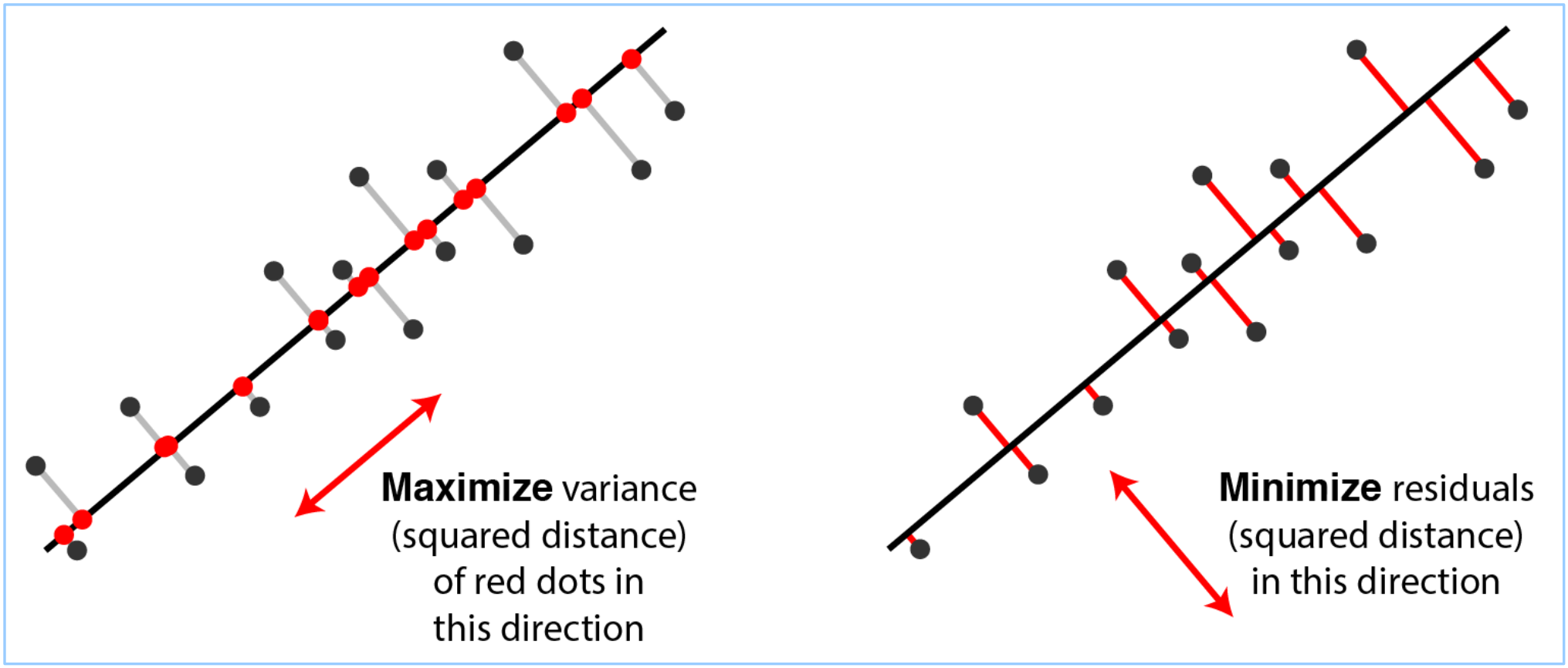
$$= \begin{pmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \sqrt{7} & 0 & 0 \\ 0 & \sqrt{3} & 0 \end{pmatrix} \begin{pmatrix} -\frac{3}{\sqrt{14}} & \frac{2}{\sqrt{14}} & -\frac{1}{\sqrt{14}} \\ \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ -\frac{2}{\sqrt{21}} & -\frac{1}{\sqrt{21}} & \frac{4}{\sqrt{21}} \end{pmatrix}$$

Outline

- Dimension Reduction
- Singular Value Decomposition (SVD)
- Principle Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Conclusion

PCA (1)

- PCA example



PCA (2)

- PCA
  - Performs a linear projection of the data to a lower-dimensional space
  - Maximizes the data variance in the low-dimensional representation
  - A simple and non-parametric method of extracting relevant information from confusing data
  - Straightforward way on how to reduce a complicate data set to a lower dimension

PCA (3)

- Problem formulation
  - Reduce the data set from n-dimensions to k-dimensions
  - Find k vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$
  - Project the data to minimize the projection error

PCA (4)

- Data preprocessing
  - An $\mathrm{m}\times n$ training data set $X = \left(x^{(1)}, x^{(2)}, \ldots, x^{(m)}\right)$, where $x^{(i)} \in \mathbb{R}^n$ and $1 \leq i \leq m$
  - Feature mean: $\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$
  - Feature scaling (mean normalization): replace $x_j^{(i)}$ with $x_j^{(i)} - \mu_j$, i.e. $x_j^{(i)} = x_j^{(i)} - \mu_j$
  - If different features on different scales, scale features to have comparable range of values

PCA (5)

- Reduce data from n-dimensions to k-dimensions
  - Compute "covariance matrix":

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}\left(x^{(i)}\right)^T = \frac{1}{m} XX^T$$

  - Compute "eigenvectors" of matrix Σ using Matlab function
    - "svd" (Singular Value Decomposition), or
    - "eig" (Eigenvalues and Eigenvectors) function

$$[U, S, V] = svd(\Sigma)$$

PCA (6)

$$U = \begin{pmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(n)} \\ | & | & & | \end{pmatrix} \in \mathbb{R}^{n \times n}$$, take the first k columns from U,

$$\Rrightarrow \quad U_{reduce} = \begin{pmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(k)} \\ | & | & & | \end{pmatrix} \in \mathbb{R}^{n \times k}$$

PCA (7)

We want to reduce $x^{(i)} \in \mathbb{R}^n \longrightarrow z^{(i)} \in \mathbb{R}^k$, let $z^{(i)} = U_{reduce}^T x^{(i)}$, then

$$z^{(i)} = \begin{pmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(k)} \\ | & | & & | \end{pmatrix}^T x^{(i)} = \begin{pmatrix} - & \left(u^{(1)}\right)^T & - \\ - & \left(u^{(2)}\right)^T & - \\ & \vdots & \\ - & \left(u^{(k)}\right)^T & - \end{pmatrix} x^{(i)}$$

PCA (8)

Algorithm 1 PCA algorithm

Input: an $m \times n$ data set $X = (x^{(1)}, x^{(2)}, \cdots, x^{(m)})$ and parameter k

1: mean normalization (feature scaling): $\Sigma = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)})(x^{(i)})^T$

2: $[U, S, V] = svd(\Sigma)$

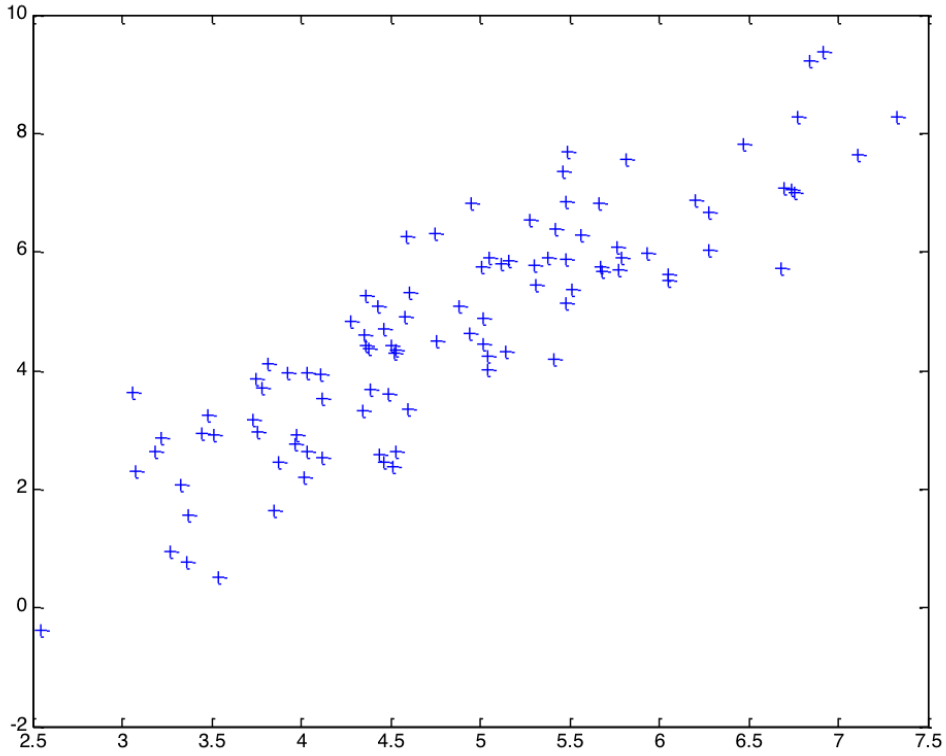
3: $U_{reduce} = U(:, 1:k)$

4: $z^{(i)} = U_{reduce}^T \, x^{(i)}$

Output: $z^{(i)} \in \mathbb{R}^k$

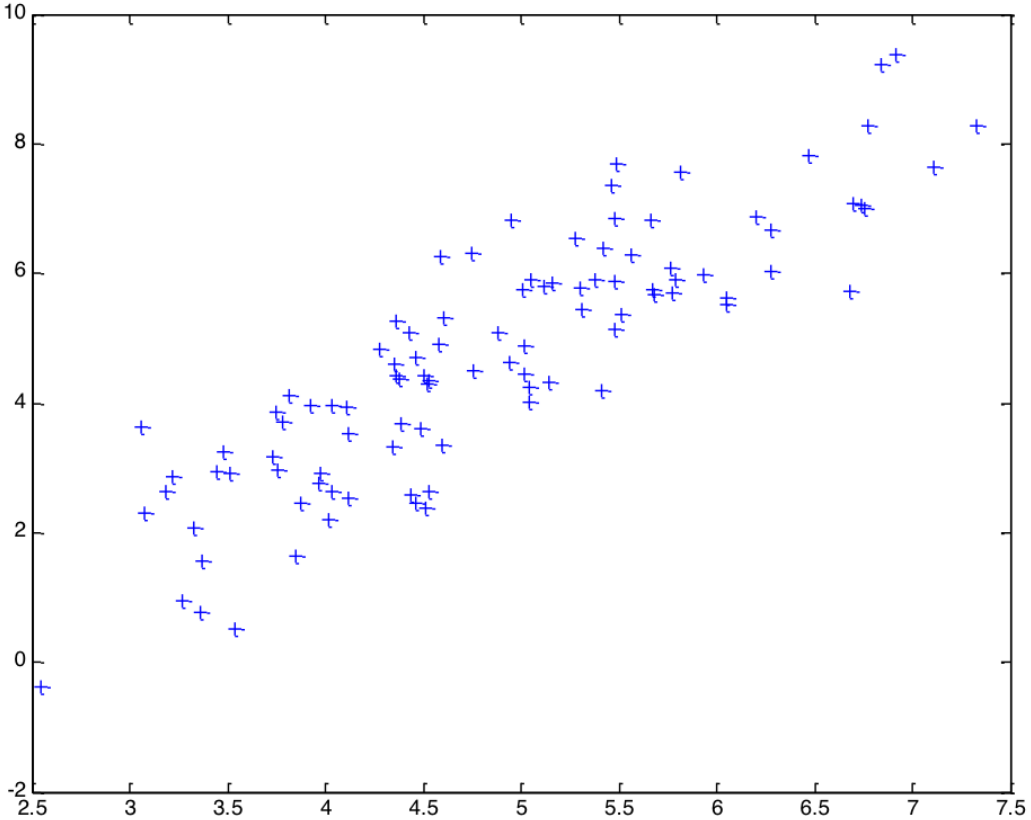
PCA (9)

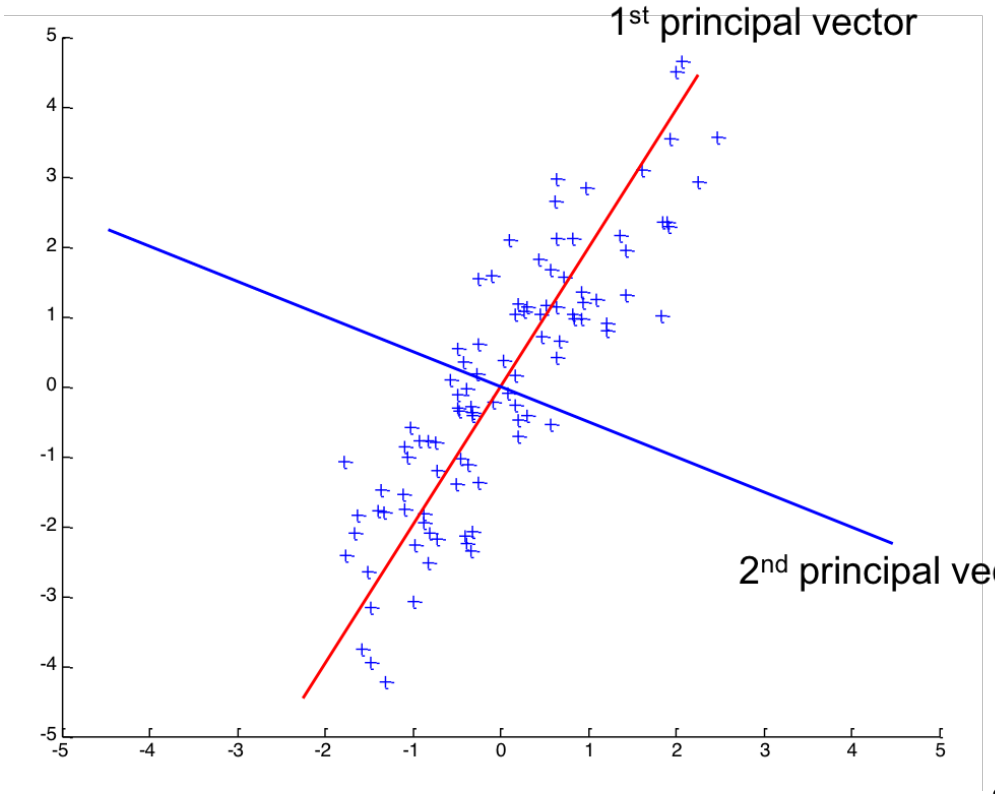
- 2D to 1D
  - Minimum root-mean-square error
  - Principle vectors are orthogonal



PCA (10)

- 2D to 1D



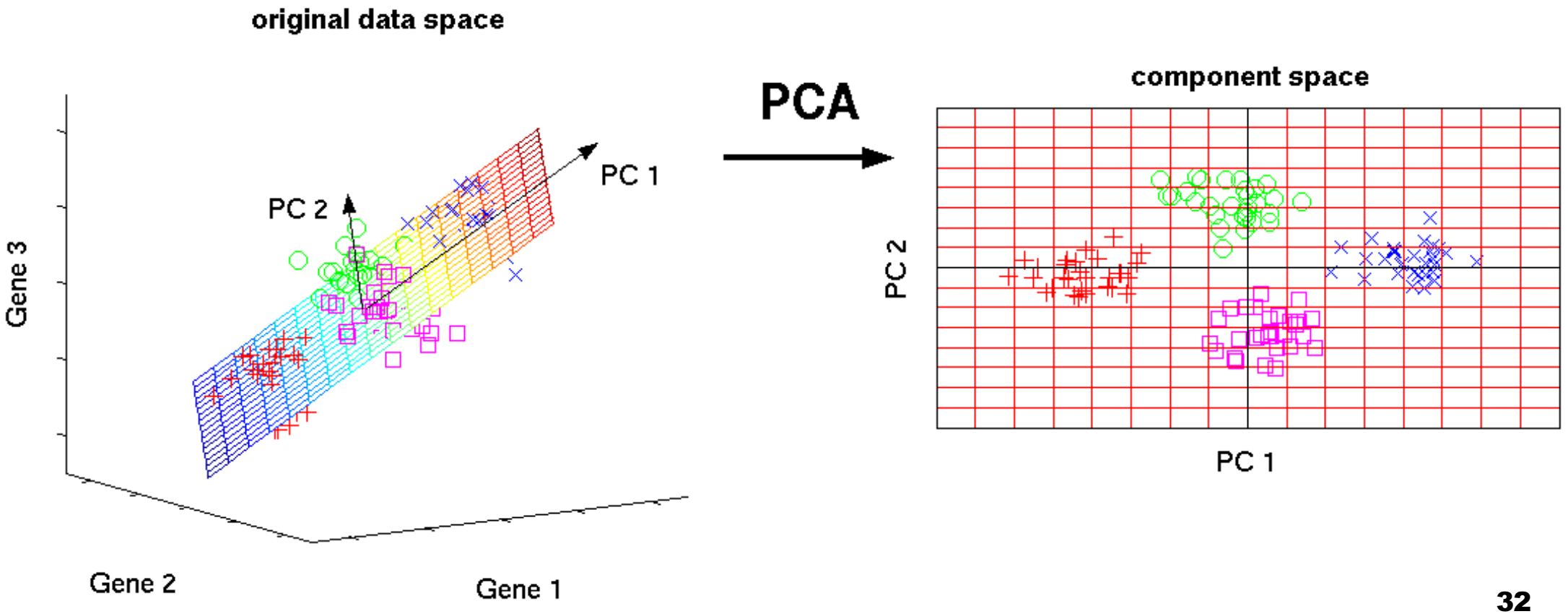


PCA (11)

- 3D to 2D
  - Seek the most accurate data representation in a lower dimensional space
  - Good direction/subspace for projection is usually in the direction of the large variance

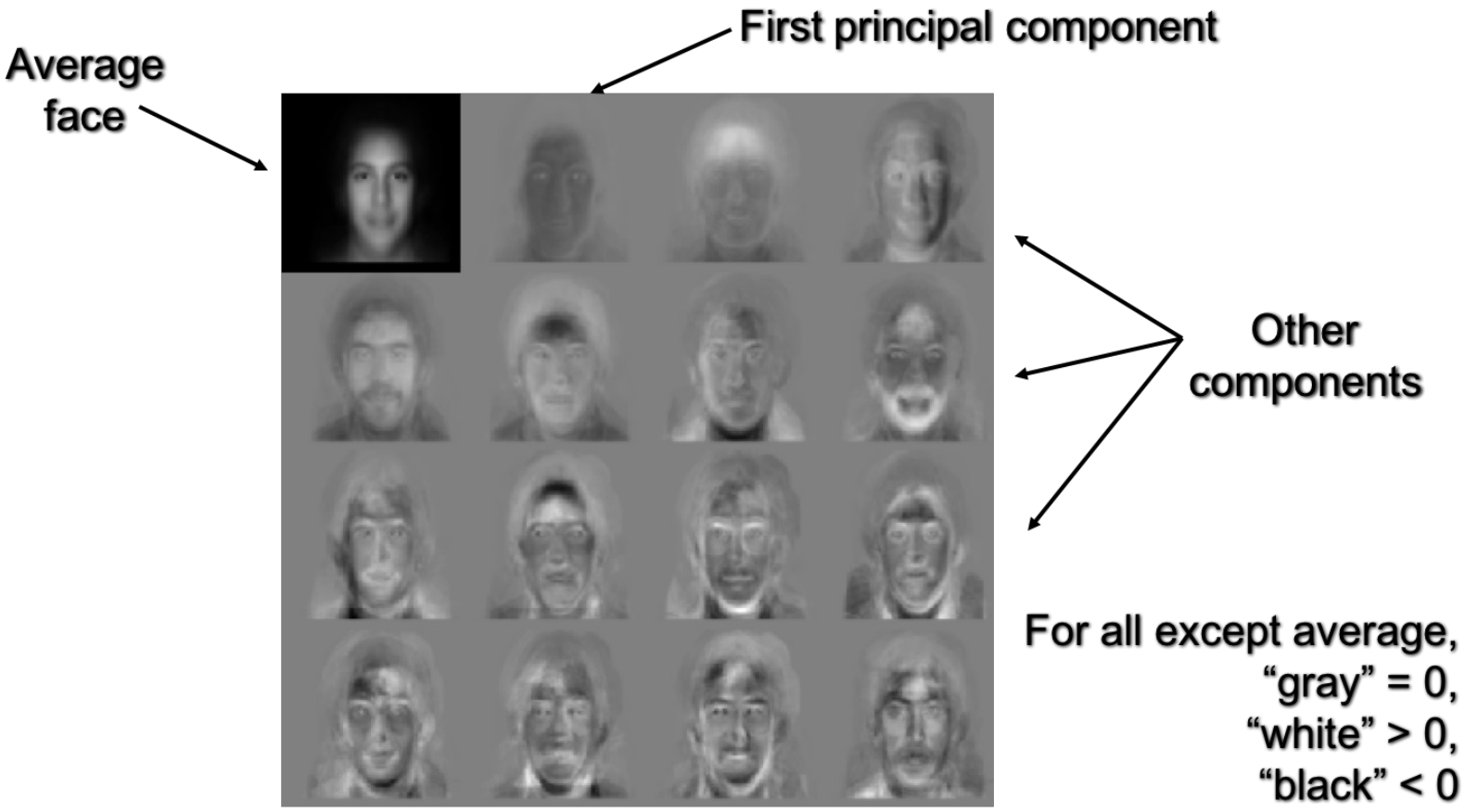
PCA (12)

- 3D to 2D



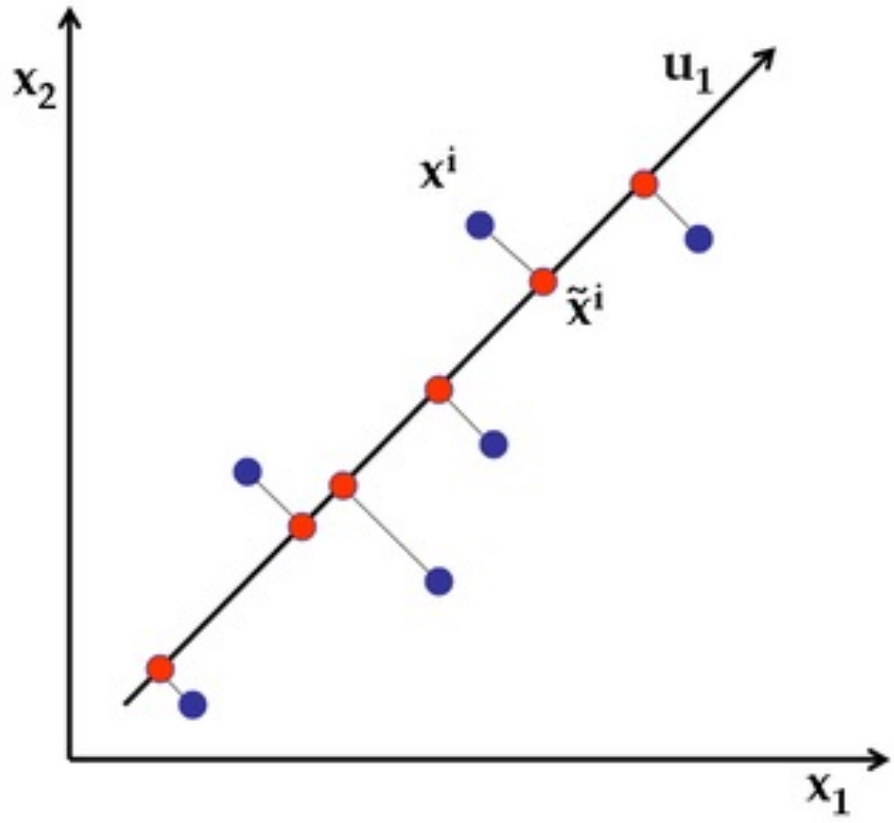
PCA (13)

- Eigenfaces



PCA (14)

- Reconstruct original data from compressed representation $z^{(i)}$



- After PCA, we obtain

$$z^{(i)} = U_{reduce}^{T} x^{(i)}$$

- Reconstruct original data from $z^{(i)}$

$$\tilde{x}^{(i)} = U_{reduce} z^{(i)}$$

PCA (15)

- Reconstruction example:

  eigenfaces

PCA (16)

- Choosing k (number of principle components)
  - Average squared projection error: $\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}-\tilde{x}^{(i)}\right\|^2$
  - Total variation in the data: $\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}\right\|^2$
  - Choose k to be the smallest value so that

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}-\tilde{x}^{(i)}\right\|^2}{\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}\right\|^2}\leq 0.01$$

  - 99% of variance is retained.

PCA (17)

- Algorithm of choosing k

Algorithm 2 Slow selection of k

Input: Data set $x^{(1)}, x^{(2)}, \cdots, x^{(m)}$ and parameter $k = 1$

while $\frac{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - \widetilde{x}^{(i)}\|^2}{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2} > 0.01$ **do**

2: Perform PCA, then compute U_{reduce}, $z^{(1)}$, $z^{(2)}, \cdots, z^{(m)}$, and $\widetilde{x}^{(1)}, \widetilde{x}^{(2)}, \cdots, \widetilde{x}^{(m)}$

$k = k + 1$

4: **end while**

Output: k

Algorithm 2: Slow selection of k

Input: Dataset $X = (x^{(1)}, x^{(2)}, ..., x^{(m)})$ and parameter $k = 1$

1. Mean normalization: $\Sigma = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)})(x^{(i)})^T$;
2. $[U, S, V] = svd(\Sigma)$;

repeat

$U_{reduce} = U(:, 1:k)$;

compute $z^{(1)}, z^{(2)}, ..., z^{(m)}$ where $z^{(i)} = U_{reduce}^T x^{(i)}$;

compute $\widetilde{x}^{(1)}, \widetilde{x}^{(2)}, ..., \widetilde{x}^{(m)}$ where $\widetilde{x}^{(i)} = U_{reduce} z^{(i)}$;

$k = k + 1$;

until $\frac{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - \widetilde{x}^{(i)}\|^2}{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2} \leq 0.01$;

Output: k

PCA (18)

- Algorithm of choosing k (Cont.)
  - After $[U, S, V] = svd(\Sigma)$
  - Pick the smallest k for

$$\frac{\sum_{i=1}^{k} s_{ii}}{\sum_{i=1}^{n} s_{ii}} \geq 0.99$$

  - Namely 99% of variance retained

Algorithm 3 Fast selection of k

Input: Data set $x^{(1)}, x^{(2)}, \cdots, x^{(m)}$ and parameter $k = 1$

$[U, S, V] = svd(\Sigma)$, then

$S = diag(s_{11}, s_{22}, \cdots, s_{nn})$

while $\frac{\sum_{i=1}^{k} s_{ii}}{\sum_{i=1}^{n} s_{ii}} < 0.99$ **do**

3: $k = k + 1$

end while

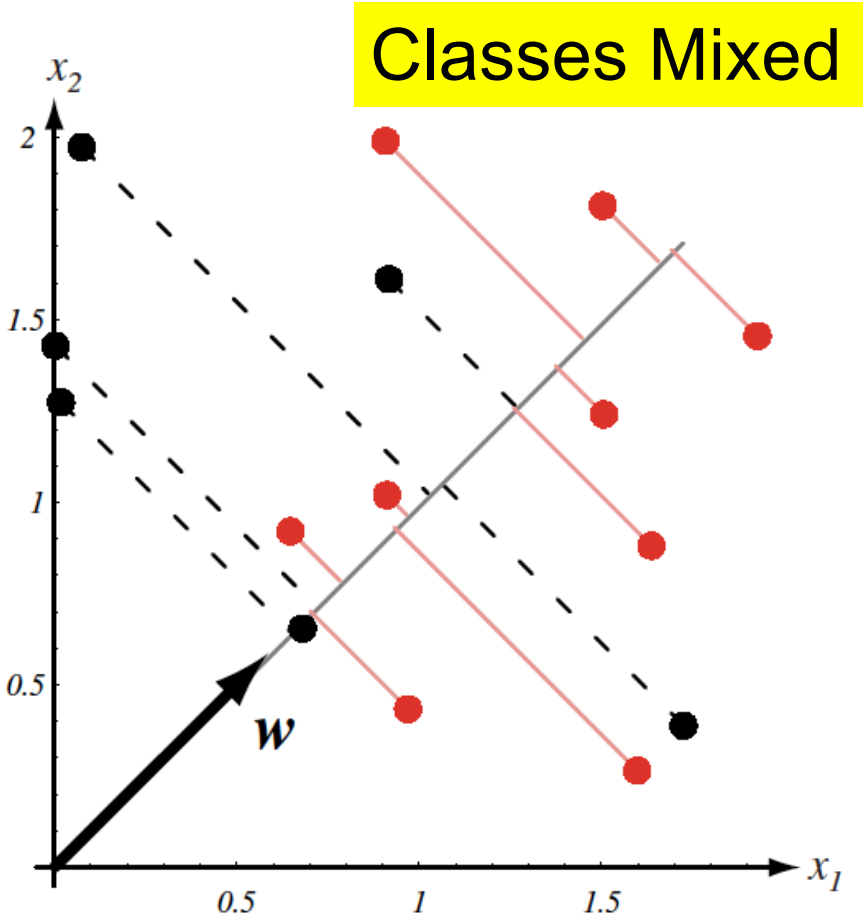
Output: k

Outline

- Dimension Reduction
- Singular Value Decomposition (SVD)
- Principle Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Conclusion

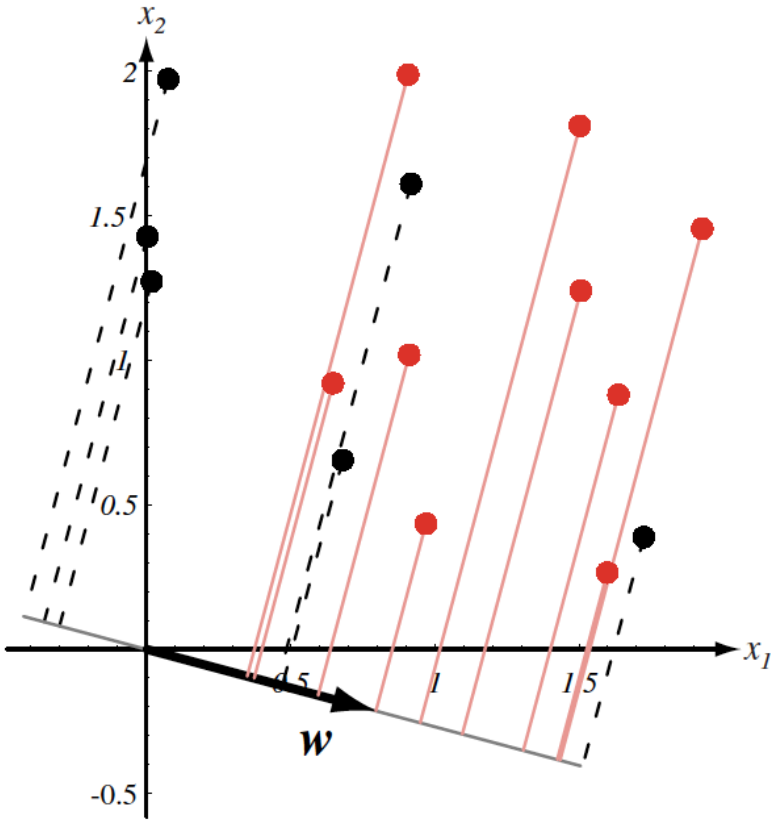
LDA (1)

- PCA
  - finds the most accurate data representation in a lower dimensional space
  - Projects data in the directions of maximum variance
- Problem:
  - The directions of maximum variance may be useless for classification



LDA (2)

- LDA
  - is also called Fisher Linear discriminant analysis
  - projects data to the direction useful for data classification
- Objective
  - LDA reduces data dimensionality while preserving the class discriminative information



better separation

LDA (3)

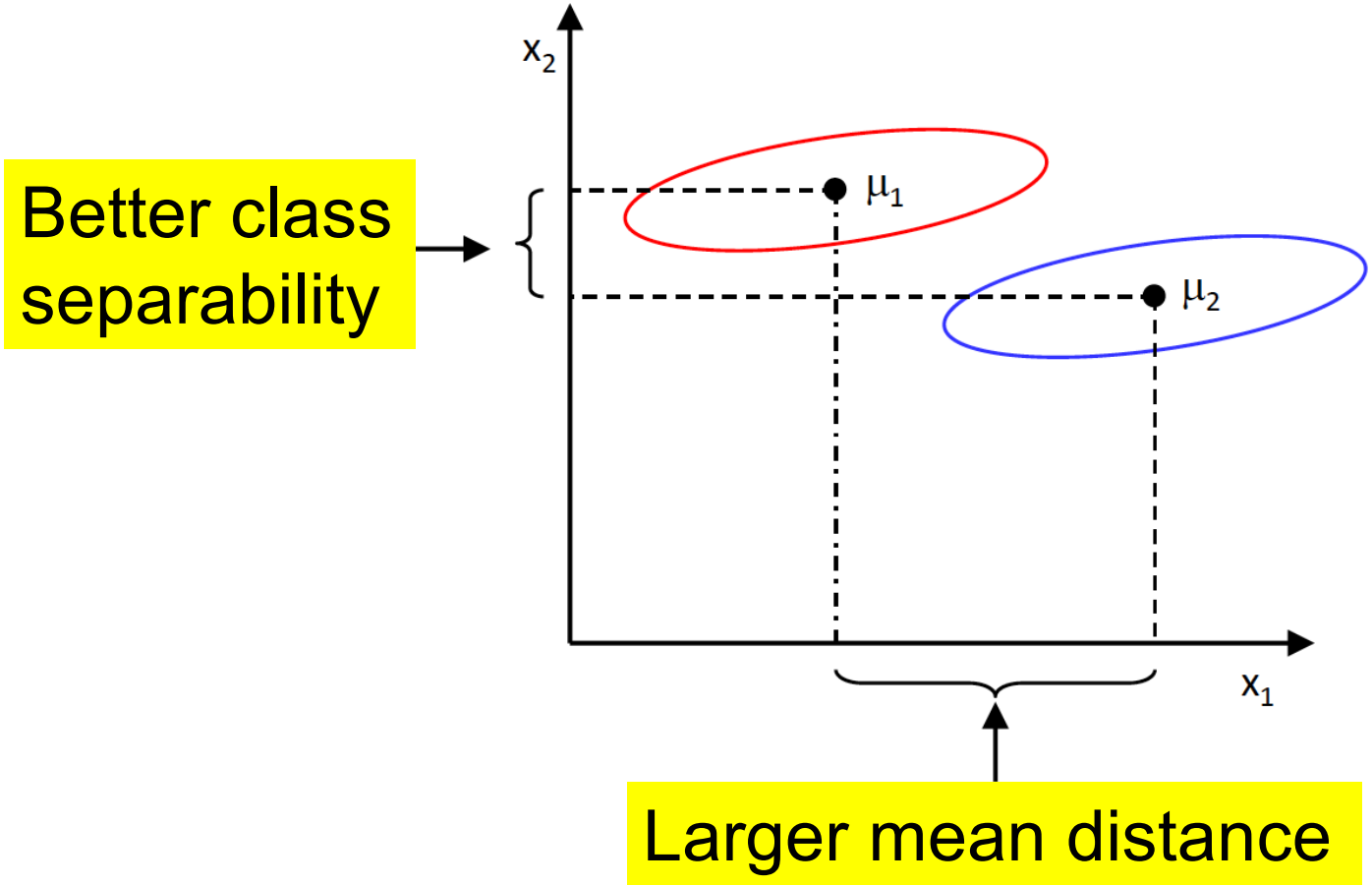
- A set of d-dimensional samples $x = \{x_1, x_2, \ldots, x_N\}$
  - N_1 of which belong to class ω_1
  - N_2 of which belong to class ω_2
- We seek to a scalar $y = \{y_1, y_2, \ldots, y_N\}$ by projecting samples onto a line

$$y = w^T x$$

- LDA selects the line that maximizes the separability of the scalars

LDA (4)

- define a measure of separation for two-class classification



LDA (5)

- Sample means of each class in x-space and y-space

$$\mu_i = \frac{1}{N_i} \sum_{x \in \omega_i} x$$

$$\tilde{\mu}_i = \frac{1}{N_i} \sum_{y \in \omega_i} y = \frac{1}{N_i} \sum_{x \in \omega_i} w^T x = w^T \mu_i$$

- Distance between projected means is

$$|\tilde{\mu}_1 - \tilde{\mu}_2| = |w^T(\mu_1 - \mu_2)|$$

  - It ignores the standard deviation within classes

LDA (6)

- Fisher's solution is to maximize the difference between the means, normalized by a measure of the within-class scatter
- Scatter is equivalent of the variance of each class

$$\tilde{s}_i^2 = \frac{1}{N_i} \sum_{y \in \omega_i} (y - \tilde{\mu}_i)^2$$

- The within-class scatter of the projected samples: $(\tilde{s}_1^2 + \tilde{s}_2^2)$
- the criterion function

$$J(w) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

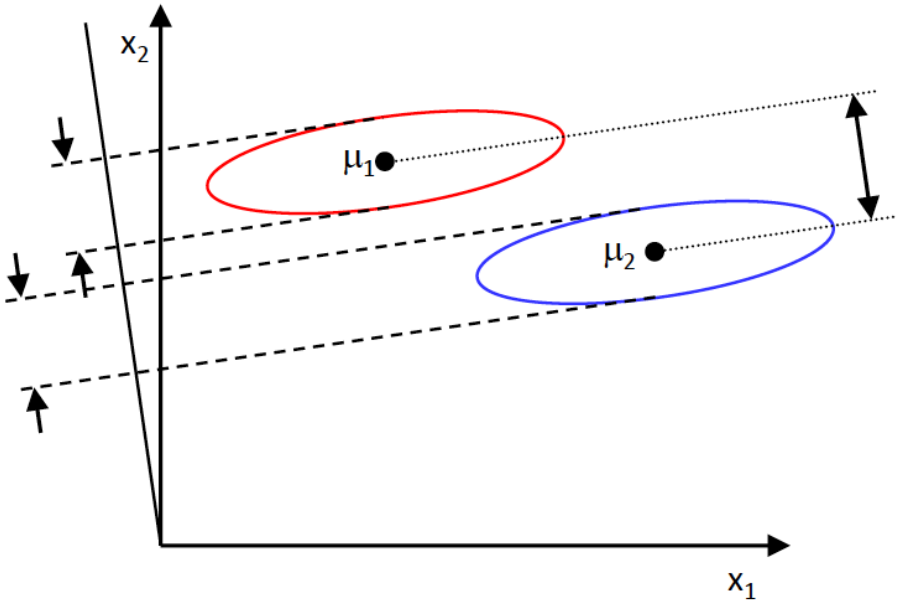
LDA (7)

- Fisher linear discriminant is defined as a linear function $w^T x$ that maximizes the criterion function $J(w)$

- LDA seeks for a projection

  allowing

  - samples from the same class are very close to each other
  - projected means are as farther apart as possible



LDA (8)

- To find the optimal w^*, we must express $J(w)$ as a function of w
- The scatter in x-space

$$S_i = \frac{1}{N_i}\sum_{x\in\omega_i}(x-\mu_i)^2 = \frac{1}{N_i}\sum_{x\in\omega_i}(x-\mu_i)(x-\mu_i)^T$$

- The within-class scatter matrix: $S_W = S_1 + S_2$
- The scatter in y-space

$$\tilde{s}_i^2 = \frac{1}{N_i}\sum_{y\in\omega_i}(y-\tilde{\mu}_i)^2 = \frac{1}{N_i}\sum_{x\in\omega_i}(w^Tx - w^T\mu_i)^2 = \frac{1}{N_i}\sum_{x\in\omega_i} w^T(x-\mu_i)(x-\mu_i)^Tw = w^TS_iw$$

$$\tilde{s}_1^2 + \tilde{s}_2^2 = w^TS_Ww$$

LDA (9)

- The between-class scatter: $S_B = |\mu_1 - \mu_2|^2 = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$
- The difference between the projected means:

$$(\tilde{\mu}_1 - \tilde{\mu}_2)^2 = (w^T\mu_1 - w^T\mu_2)^2 = w^T(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T w = w^T S_B w$$

- Fisher criterion becomes

$$J(w) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2} = \frac{w^T S_B w}{w^T S_W w}$$

- the optimal w^*

$$w^* = \underset{w}{\operatorname{argmax}} J(w) = \underset{w}{\operatorname{argmax}} \frac{w^T S_B w}{w^T S_W w}$$

LDA (10)

- Solve the optimization problem

$$\frac{d}{dw}J(w) = \frac{d}{dw}\left(\frac{w^T S_B w}{w^T S_W w}\right) = \frac{w^T S_W w\left(\frac{d}{dw}(w^T S_B w)\right) - w^T S_B w\left(\frac{d}{dw}(w^T S_W w)\right)}{(w^T S_W w)^2}$$

$$= \frac{w^T S_W w(2S_B w) - w^T S_B w(2S_W w)}{(w^T S_W w)^2} = 0$$

$$\Rightarrow\ w^T S_W w(2S_B w) - w^T S_B w(2S_W w) = 0 \quad \Rightarrow \quad \frac{w^T S_W w}{w^T S_W w} S_B w - \frac{w^T S_B w}{w^T S_W w} S_W w = 0$$

- Set constant $\lambda = \frac{w^T S_B w}{w^T S_W w} = J_{max}$, the problem is transferred into a generalized eigenvalue problem

$$S_B w = \lambda S_W w$$

LDA (11)

- If the inverse of S_W exists, it is a standard eigenvalue problem

$$S_W^{-1} S_B w = \lambda w$$

- For any vector x, $S_B x$ points in the same direction as $\mu_1 - \mu_2$

$$S_B x = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T x = \alpha(\mu_1 - \mu_2) \text{ where } \alpha = (\mu_1 - \mu_2)^T x$$

- Solve the eigenvalue problem as

$$w = S_W^{-1}(\mu_1 - \mu_2)$$

Because

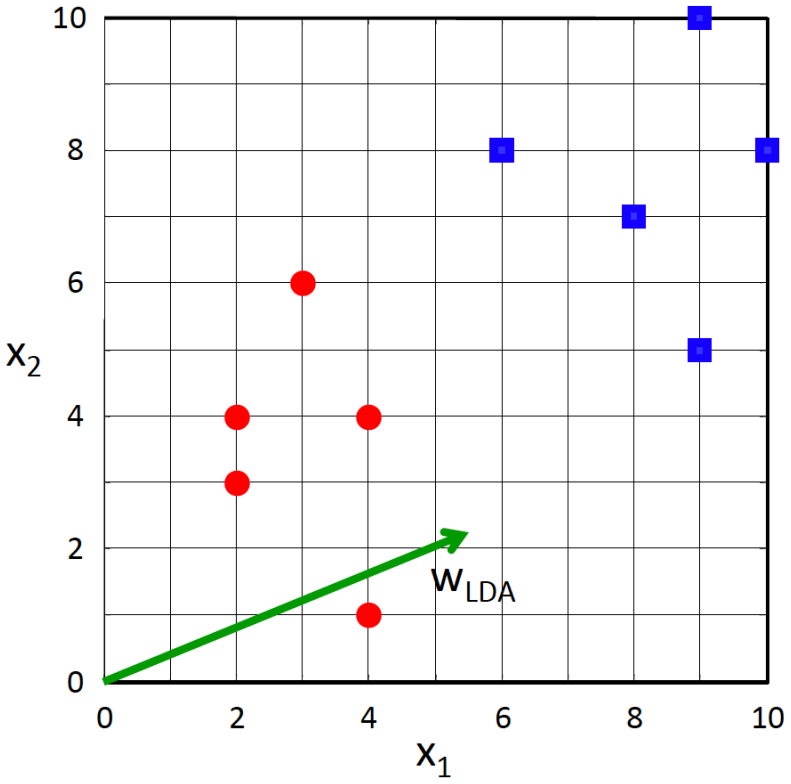
$$S_W^{-1} S_B w = S_W^{-1} S_B \left(S_W^{-1}(\mu_1 - \mu_2) \right) = S_W^{-1} \left(\alpha(\mu_1 - \mu_2) \right) = \alpha \left(S_W^{-1}(\mu_1 - \mu_2) \right) = \lambda w$$

LDA (12)

- Example: computer LDA projection of the following 2D Dataset
  - $X_1 = \{(4,1),(2,4),(2,3),(3,6),(4,4)\}$
  - $X_2 = \{(9,10),(6,8),(9,5),(8,7),(10,8)\}$
- LDA solution:
  - Arrange data into 2 separate matrices

$$X_1 = \begin{pmatrix} 4 & 2 & 2 & 3 & 4 \\ 1 & 4 & 3 & 6 & 4 \end{pmatrix}$$

$$X_2 = \begin{pmatrix} 9 & 6 & 9 & 8 & 10 \\ 10 & 8 & 5 & 7 & 8 \end{pmatrix}$$



LDA (13)

- Class statistics are

$$\mu_1 = \begin{pmatrix} \frac{4+2+2+3+4}{5} \\ \frac{1+4+3+6+4}{5} \end{pmatrix} = \begin{pmatrix} 3.0 \\ 3.6 \end{pmatrix} \quad \mu_2 = \begin{pmatrix} \frac{9+6+9+8+10}{5} \\ \frac{10+8+5+7+8}{5} \end{pmatrix} = \begin{pmatrix} 8.4 \\ 7.6 \end{pmatrix}$$

$$S_1 = \frac{1}{N_1} \sum_{x \in \omega_1} (x - \mu_1)(x - \mu_1)^T = E\left[\begin{pmatrix} 1 & -1 & -1 & 0 & 1 \\ -2.6 & 0.4 & -0.6 & 2.4 & 0.4 \end{pmatrix} \begin{pmatrix} 1 & -1 & -1 & 0 & 1 \\ -2.6 & 0.4 & -0.6 & 2.4 & 0.4 \end{pmatrix}^T \right] = \begin{pmatrix} 0.8 & -0.4 \\ -0.4 & 2.64 \end{pmatrix}$$

$$S_2 = \frac{1}{N_2} \sum_{x \in \omega_2} (x - \mu_2)(x - \mu_2)^T = E\left[\begin{pmatrix} 0.6 & -2.4 & 0.6 & -0.4 & 1.6 \\ 2.4 & 0.4 & -2.6 & -0.6 & 0.4 \end{pmatrix} \begin{pmatrix} 0.6 & -2.4 & 0.6 & -0.4 & 1.6 \\ 2.4 & 0.4 & -2.6 & -0.6 & 0.4 \end{pmatrix}^T \right] = \begin{pmatrix} 1.84 & -0.04 \\ -0.04 & 2.64 \end{pmatrix}$$

LDA (14)

- The within- and between-class scatters are

$$S_W = S_1 + S_2 = \begin{pmatrix} 0.8 & -0.4 \\ -0.4 & 2.64 \end{pmatrix} + \begin{pmatrix} 1.84 & -0.04 \\ -0.04 & 2.64 \end{pmatrix} = \begin{pmatrix} 2.64 & -0.44 \\ -0.44 & 5.28 \end{pmatrix}$$

$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T = \begin{pmatrix} -5.4 \\ -4 \end{pmatrix} (-5.4 \quad -4) = \begin{pmatrix} 29.16 & 21.6 \\ 21.6 & 16.0 \end{pmatrix}$$

- The LDA projection is

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

determinant

$$w = S_W^{-1}(\mu_1 - \mu_2) = \begin{pmatrix} 2.64 & -0.44 \\ -0.44 & 5.28 \end{pmatrix}^{-1} \begin{pmatrix} -5.4 \\ -4 \end{pmatrix} = \begin{pmatrix} -2.20 \\ -0.94 \end{pmatrix}$$

Conclusion

- Dimension reduction changes data representation to low-dimension while preserving data structure
- SVD is a factorization of a real or complex matrix
- PCA performs a linear projection of the data to a lower-dimensional space while maximizing the data variance
- LDA projects data to the direction useful for data classification

Support Vector Machines

CISC3024 – Pattern Recognition

Prof. Yicong Zhou

yicongzhou@um.edu.mo

@Fall 2024

Outline

- Motivation
- SVM
- Soft Margin Classification
- Nonlinear SVM
- Conclusion

Motivation (1)

- Support Vector Machines (SVMs)
  - Supervised learning algorithm for classification and regression
  - Derived from statistical learning theory
  - Exceptional performance in handwritten digital recognition

Motivation (2)

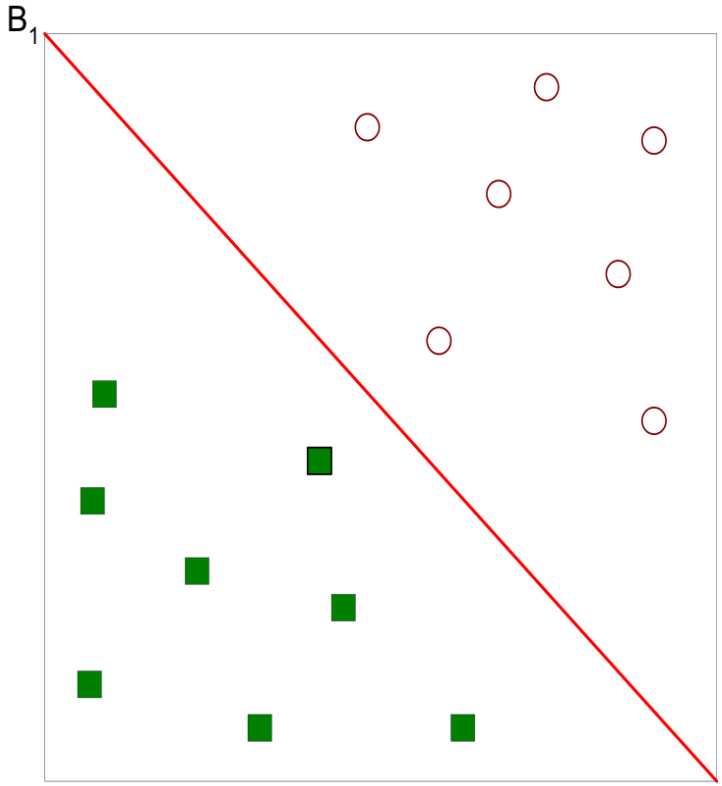
- Philosophy
  - Finding an optimized separating hyperplane to maximize margins
  - Converting problem to the "dual problem"
  - Allowing for errors in classification using "slack variables"
  - Using kernel mapping for better linear separation of nonlinearly separable data
  - A kernel is like using an infinite number of features

SVM (1)

- Given two linearly separable classes, Each hyperplane is characterized by:

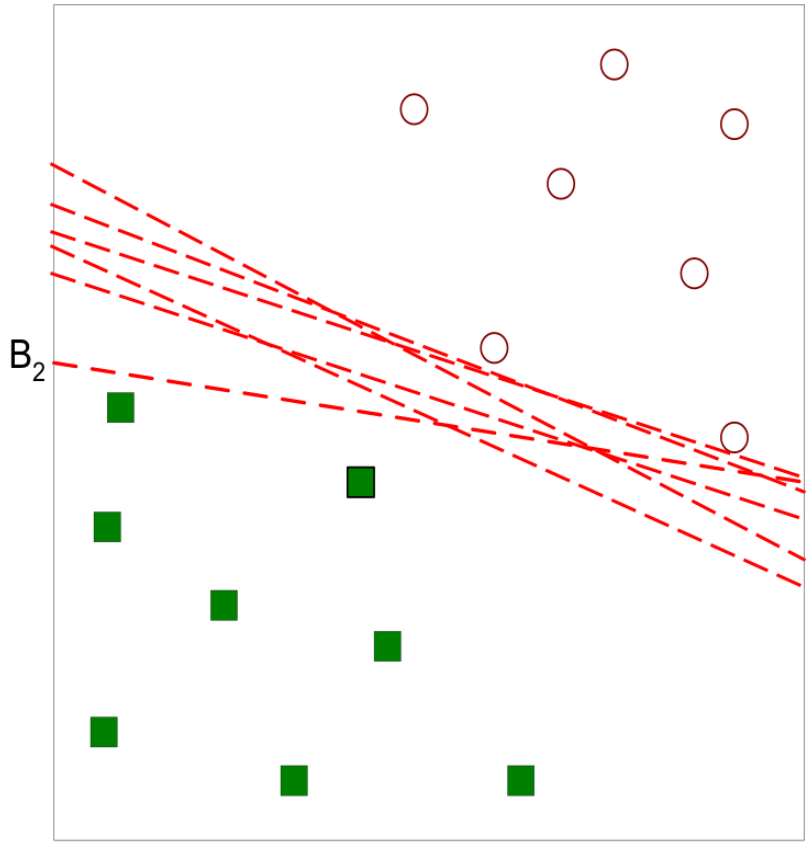
$$f(x) = w^T x + b = 0$$

  - Its direction in space, w
  - Its position in space, b



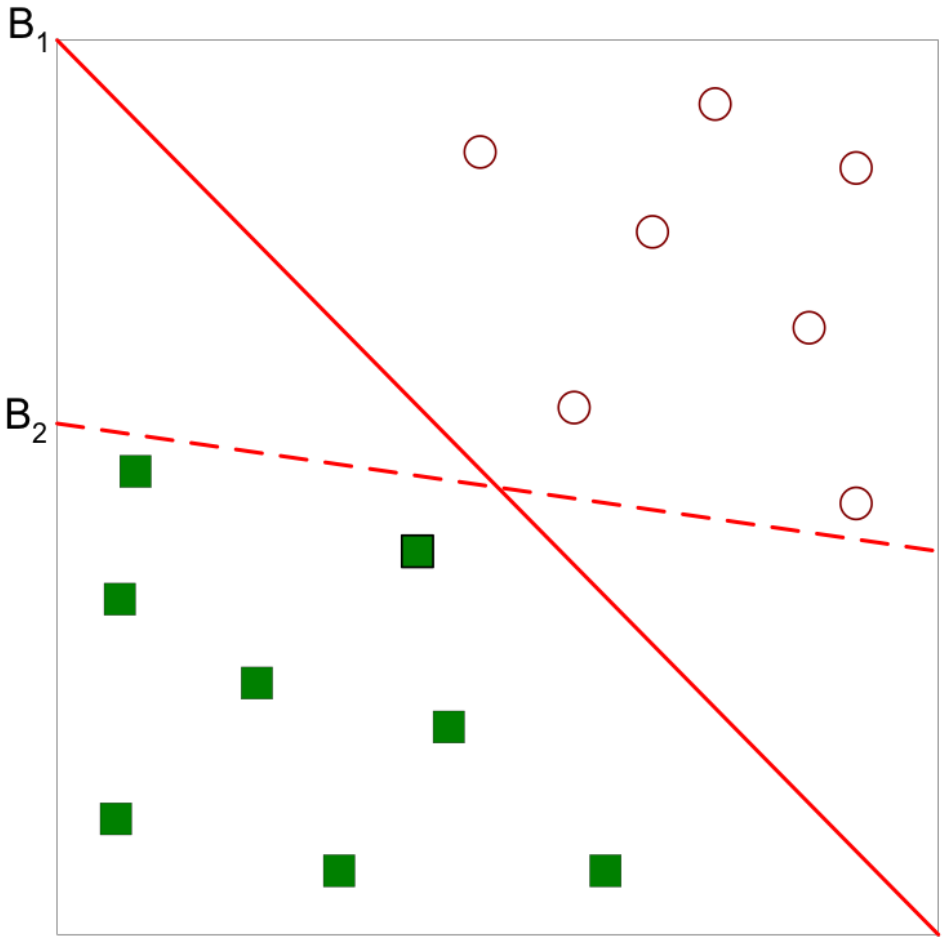
SVM (2)

- SVMs use a single hyperplane to distinguish two classes



Many possible solutions !!

SVM (3)

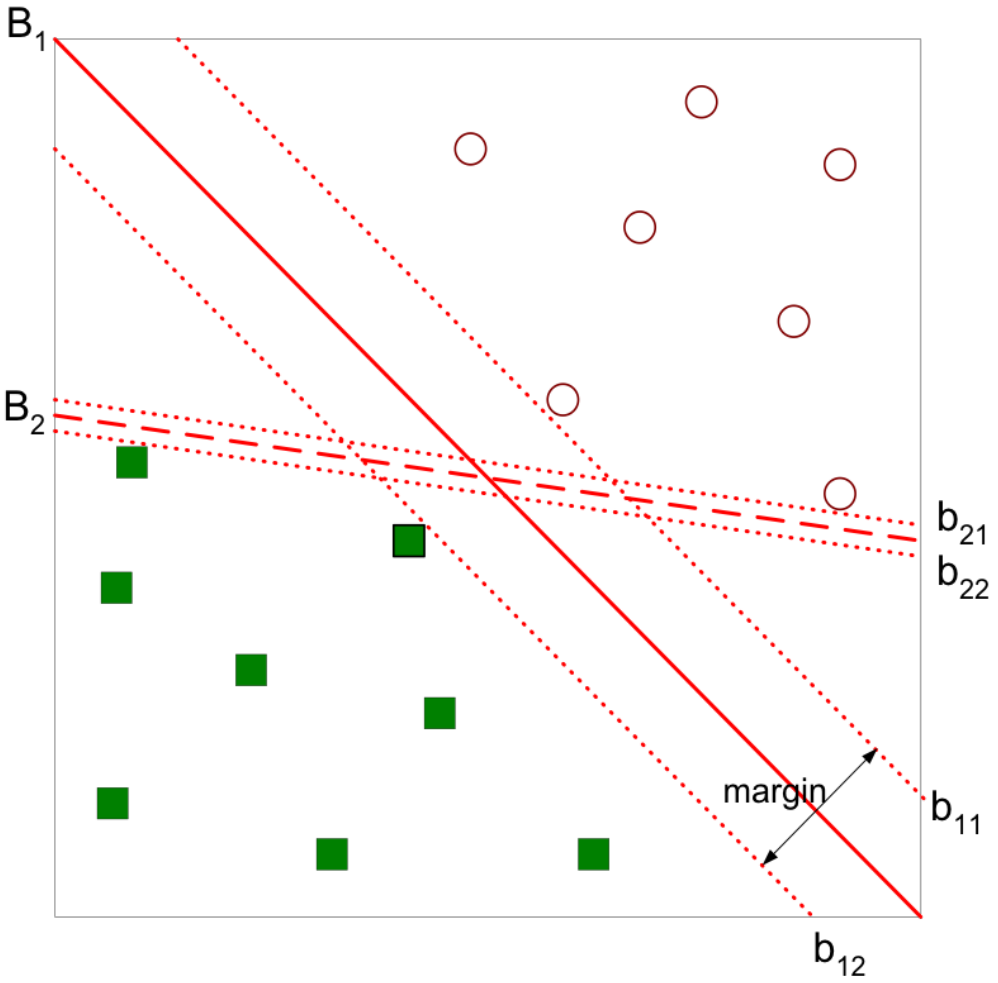


- Which one is better?

  B_1 or B_2 ?

- How to define better?

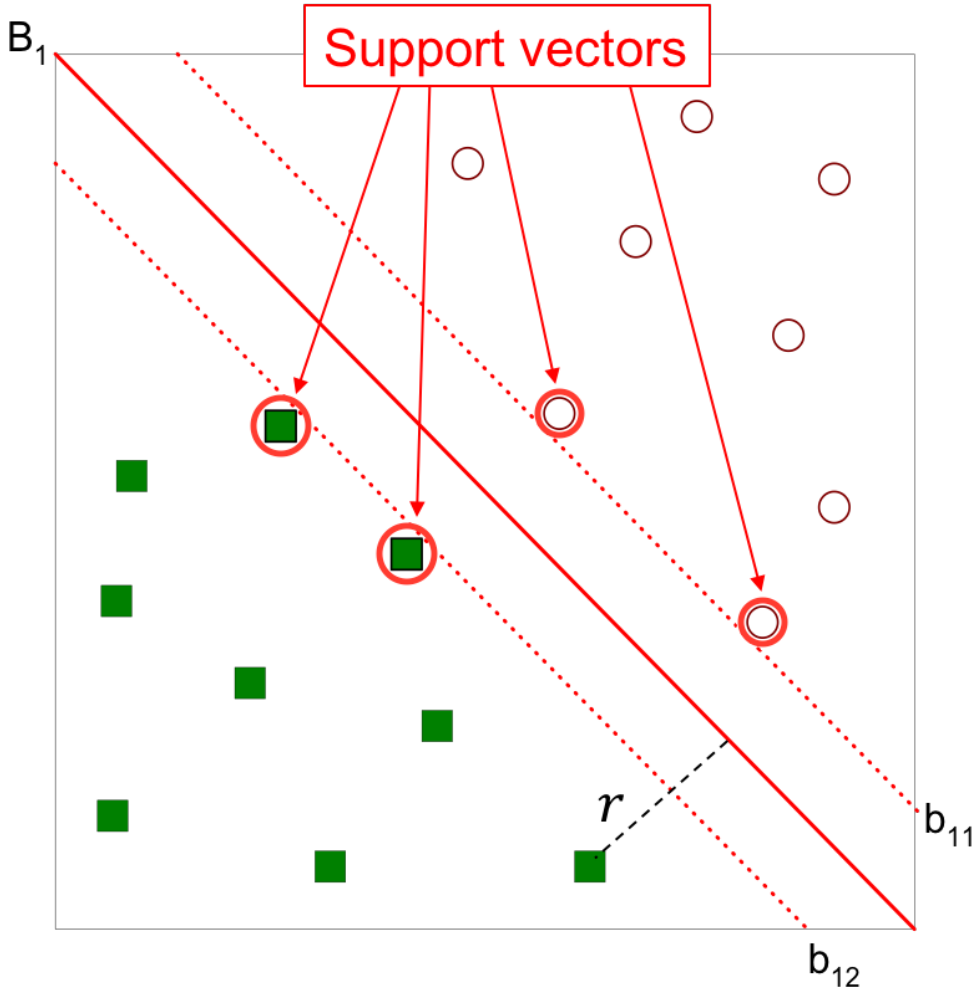
SVM (4)



- Hyperplane maximizes the margin

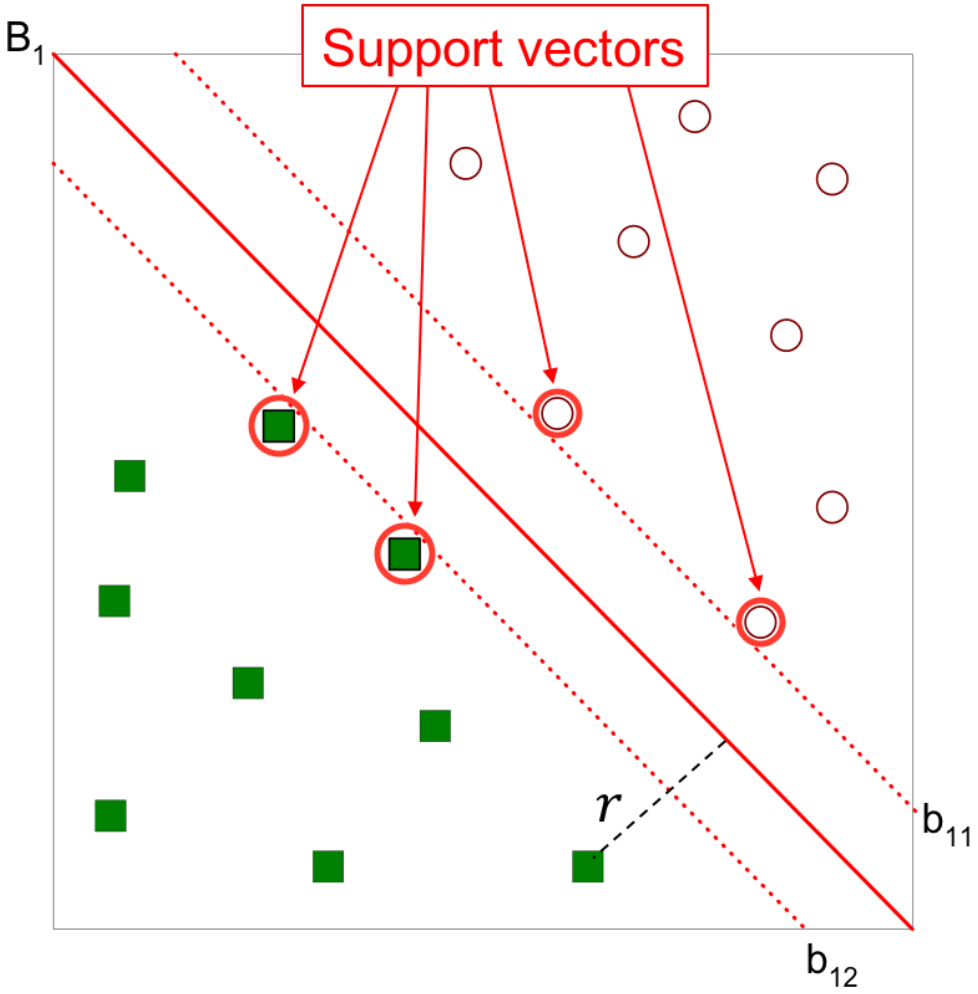
B_1 is better than B_2

SVM (5)



- Support vectors
  - a subset of training samples
  - samples closest to hyperplane
  - the most difficulty to classify

SVM (6)



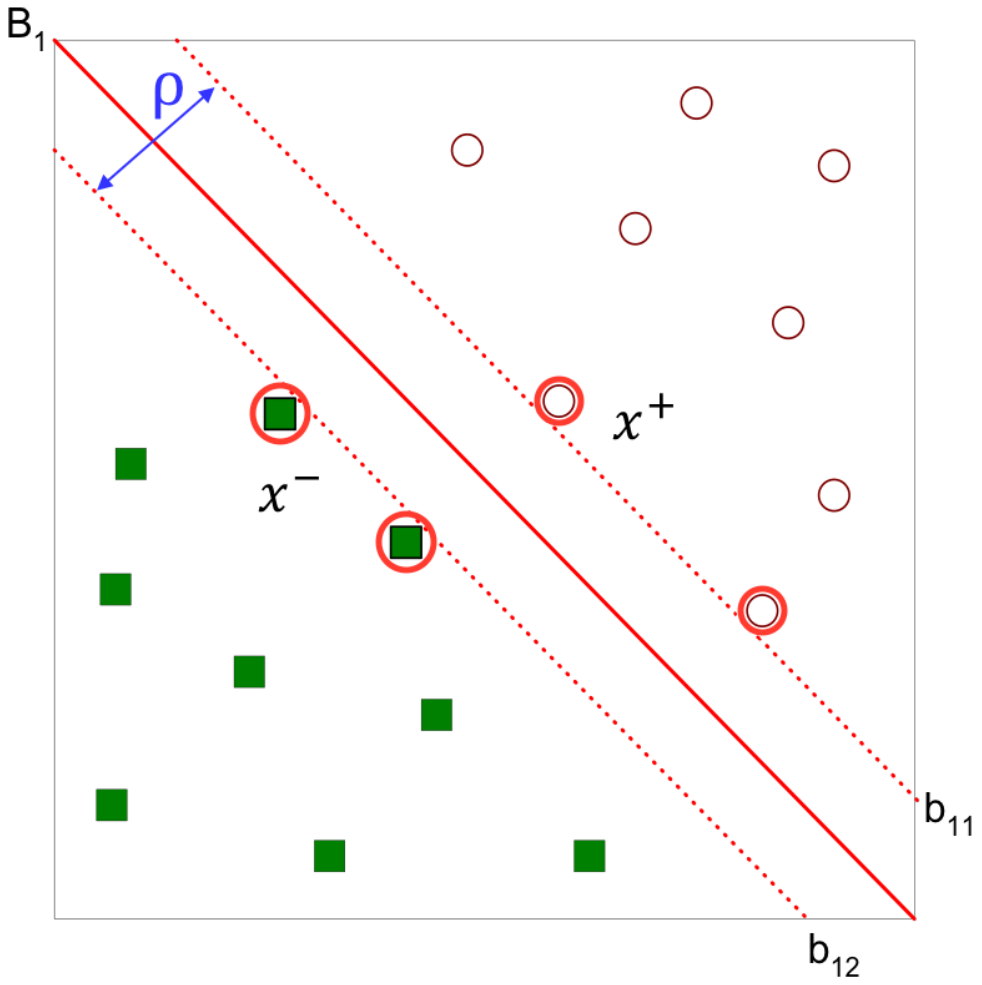
- Distance from each sample x_i to the hyperplane is

$$r = \frac{w^T x_i + b}{\|w\|}$$

w is the weight vector

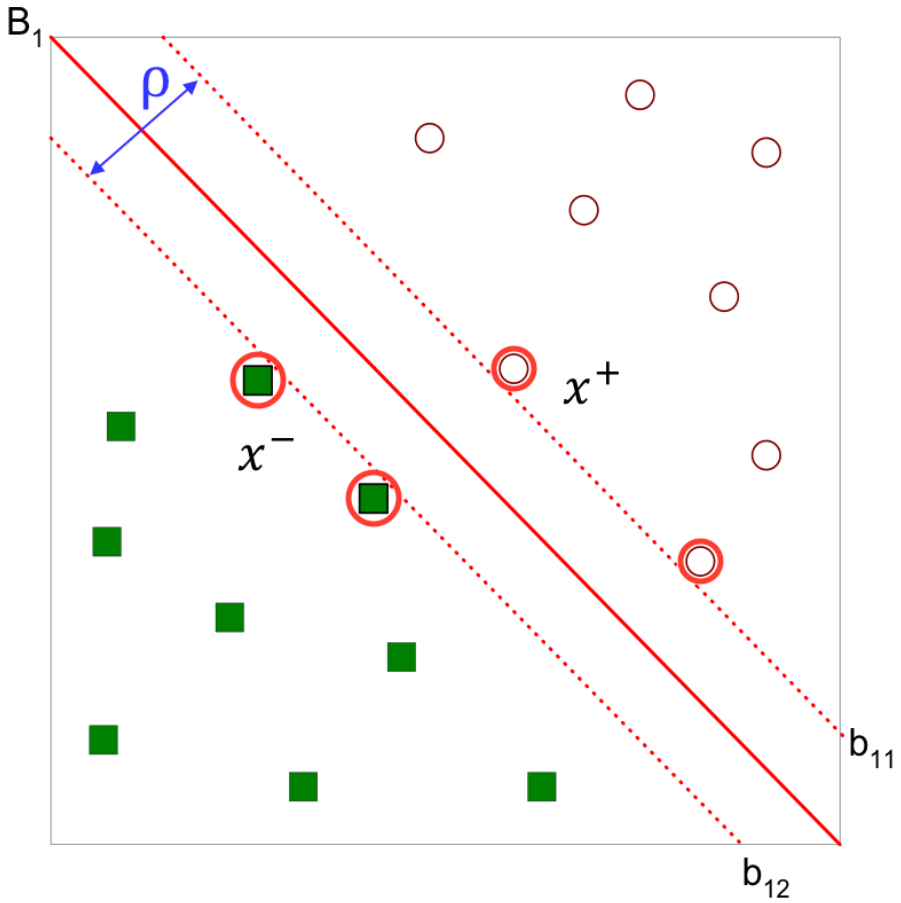
b is the bias

SVM (7)



- Margin ρ is the distance between support vectors
- Hyperplane is to maximize the margin ρ

SVM (8)



$$B_1: \; w^T x + b = 0$$

$$b_{11}: \; w^T x^+ + b = 1$$

$$b_{12}: \; w^T x^- + b = -1$$

Then, $w^T(x^+ - x^-) = 2$

Margin: $\rho = \frac{w}{\|w\|}(x^+ - x^-) = \frac{w(x^+ - x^-)}{\|w\|}$

Then, $\rho = \frac{2}{\|w\|}$

SVM (9)

- Let $X = (x_1, x_2, \ldots, x_N)$ to be the data set, and $y_i \in \{1, -1\}$ be the class label of x_i.
- We can formulate the quadratic optimization problem of SVM

$$\max_{w,b} \frac{2}{\|w\|} \quad s.t. \begin{cases} w^T x_i + b \geq 1 & \text{if } y_i = +1 \\ w^T x_i + b \leq -1 & \text{if } y_i = -1 \end{cases} \quad \text{for } i = 1, 2, \ldots, N$$

- Maximize the margin $\rho = \frac{2}{\|w\|}$ is equivalent to minimize

$$\frac{1}{2}\|w\|^2 = \frac{1}{2} w^T w$$

SVM (10)

- We can formulate the quadratic optimization problem of SVM

$$\min_{w,b} \frac{1}{2}\|w\|^2 \quad s.t. \;\; y_i(w^T x_i + b) \geq 1 \;\; \forall i = 1, 2, \ldots, N$$

- This quadratic optimization problem subject to linear constraints
- There is a unique minimum.
- SVM learns a linear classifier:

$$f(x) = w^T x + b$$

SVM (11)

$$\min_{w,b} \frac{1}{2}\|w\|^2 \quad s.t. \;\; 1 - y_i(w^T x_i + b) \leq 0 \;\; \forall i = 1, 2, \ldots, N$$

- The Lagrangian of the quadratic optimization problem is

$$L(w, b) = \frac{1}{2} w^T w + \sum_{i=1}^{N} \lambda_i \left(1 - y_i(w^T x_i + b)\right)$$

where $\|w\|^2 = w^T w$, and $\lambda_i \geq 0$ is a Lagrangian multiplier

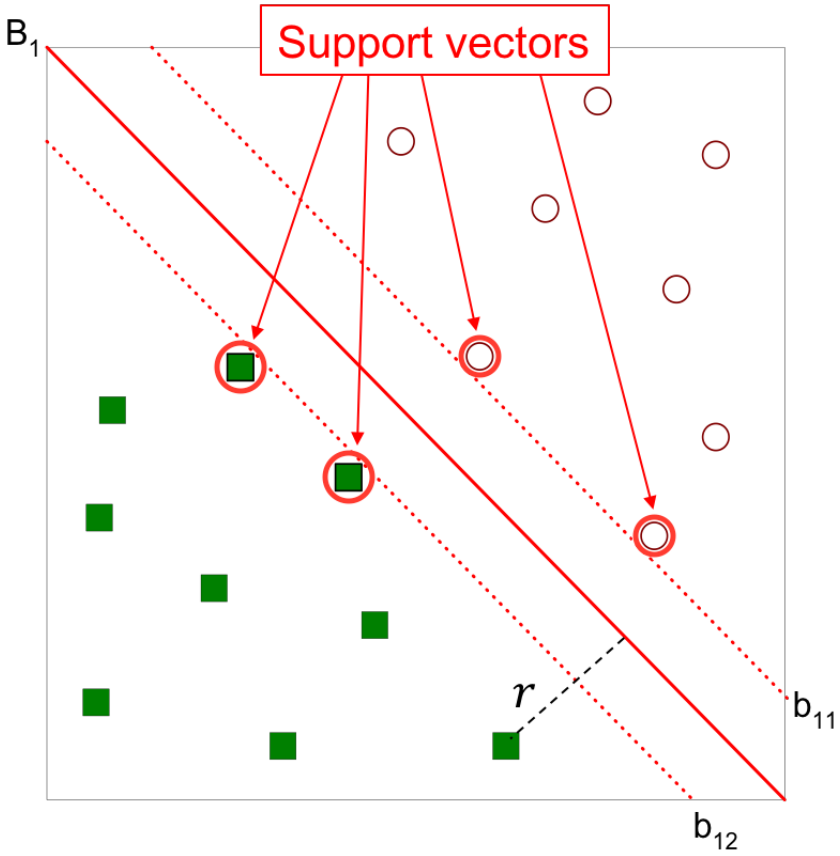
SVM (12)

- Setting the gradient of $L \;\; w.r.t \;\; w$ and b to zero, we have

$$\frac{dL}{dw} = w + \sum_{i=1}^{n} \lambda_i(-y_i)x_i = 0 \quad \Rrightarrow \quad w = \sum_{i=1}^{N} \lambda_i y_i x_i$$

$$\frac{dL}{db} = \sum_{i=1}^{n} \lambda_i(-y_i) = 0 \quad \Rrightarrow \quad \sum_{i=1}^{N} \lambda_i y_i = 0$$

SVM (13)



- Support vectors are the closest vectors for each class to the classifier

$$w^T x_i + b = \pm 1$$

- The optimal hyperplane classifier of a support vector machine is unique.

SVM (14)

- The quadratic optimization problem

$$L(w,b) = \frac{1}{2} w^T w + \sum_{i=1}^{N} \lambda_i \left(1 - y_i (w^T x_i + b)\right) = \frac{1}{2} \left(\sum_{i=1}^{N} \lambda_i y_i x_i\right)^T \left(\sum_{j=1}^{N} \lambda_j y_j x_j\right) + \sum_{i=1}^{N} \lambda_i \left(1 - y_i \left(\left(\sum_{j=1}^{N} \lambda_j y_j x_j\right)^T x_i + b\right)\right)$$

$$= \frac{1}{2} \sum_{i=1,j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j + \sum_{i=1}^{N} \lambda_i - \sum_{i=1}^{N} \lambda_i y_i \left(\sum_{j=1}^{N} \lambda_j y_j x_j{}^T x_i + b\right) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1,j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j - b \sum_{i=1,j=1}^{N} \lambda_i y_i$$

SVM (15)

- We can transform the problem to its dual problem

$$\max_{\lambda_i} W(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1,j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j$$

$$s.t. \quad \lambda_i \geq 0, \qquad \sum_{i=1}^{N} \lambda_i y_i = 0 \quad \forall \lambda_i$$

SVM (16)

- This dual problem is a convex quadratic programming (QP) problem
  - Global maximum of λ_i can always be found
  - Well established tools for solving this optimization problem
  - The solution involves constructing a *dual problem*
  - *Lagrange multiplier* λ_i is associated with every inequality constraint in the original problem

SVM (17)

- Given a solution $\lambda_1, \lambda_2, \ldots, \lambda_N$ to the dual problem, solution to the original problem is

$$w = \sum_{i=1}^{N} \lambda_i y_i x_i \qquad b = y_k - \sum_{i=1}^{N} \lambda_i w^T x_k \quad \forall \lambda_k > 0$$

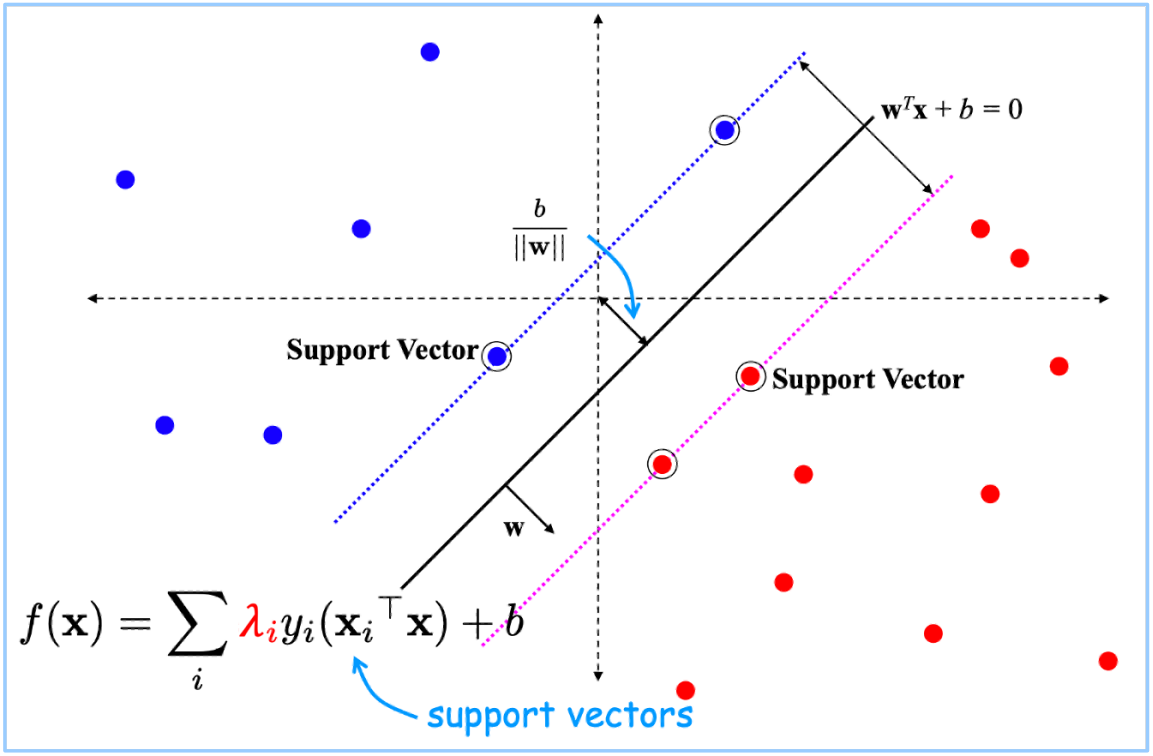
SVM (18)

- Primal version of classifier:

$$f(x) = w^T x + b$$

- Dual version of classifier:

$$f(x) = \sum_{i=1}^{N} \lambda_i y_i (x_i{}^T x) + b$$



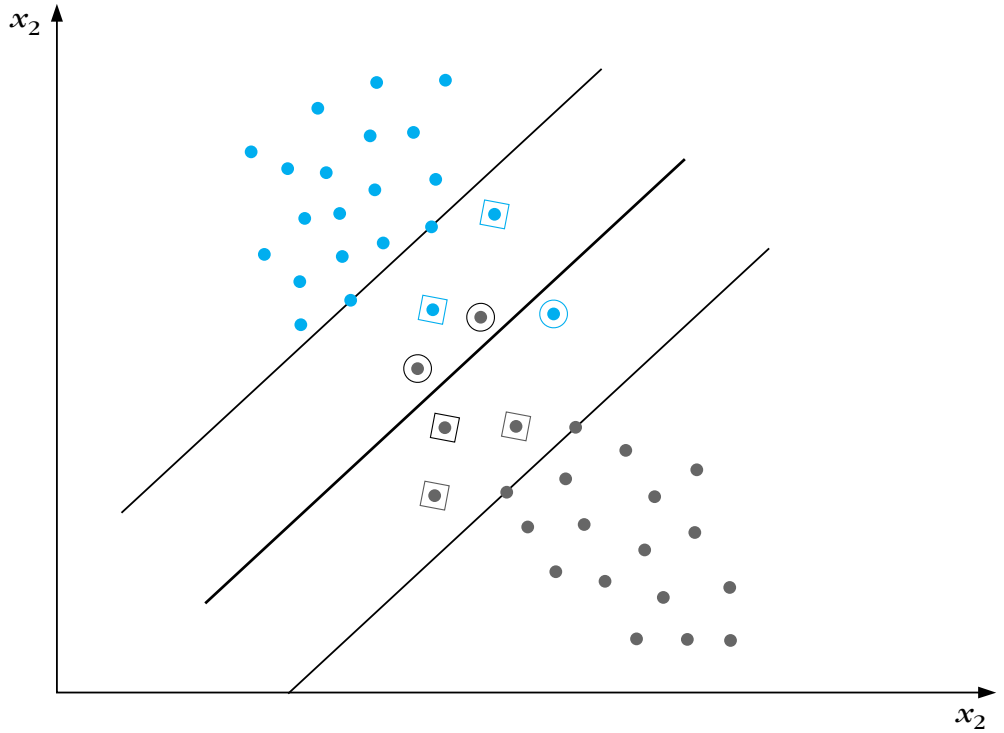
Outline

- Motivation
- SVM
- Soft Margin Classification
- Nonlinear SVM
- Conclusion

Soft Margin Classification (1)

- Non-separable Classes
  - There is no hyperplane satisfying

$$y_i(w^T x + b) > 1 \quad \forall x$$



Soft Margin Classification (2)

- Training samples belong to one of three possible categories
  - Samples outside the margin (correctly classified)

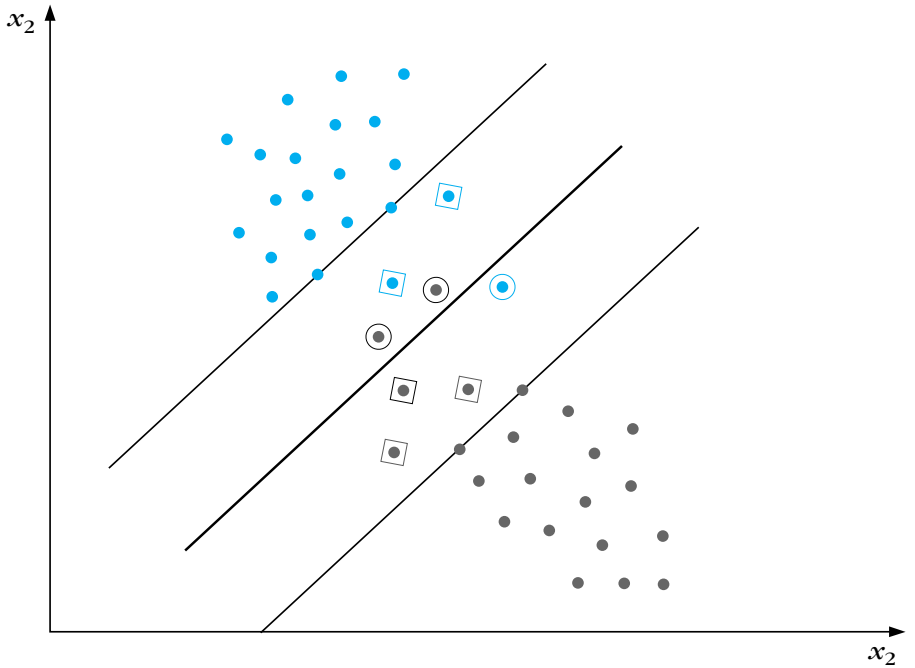
$$y_i(w^T x + b) > 1$$

  - Samples inside the margin (correctly classified)-- **Margin Violation**

$$0 \leq y_i(w^T x + b) < 1$$

  - Misclassified samples

$$y_i(w^T x + b) < 0$$

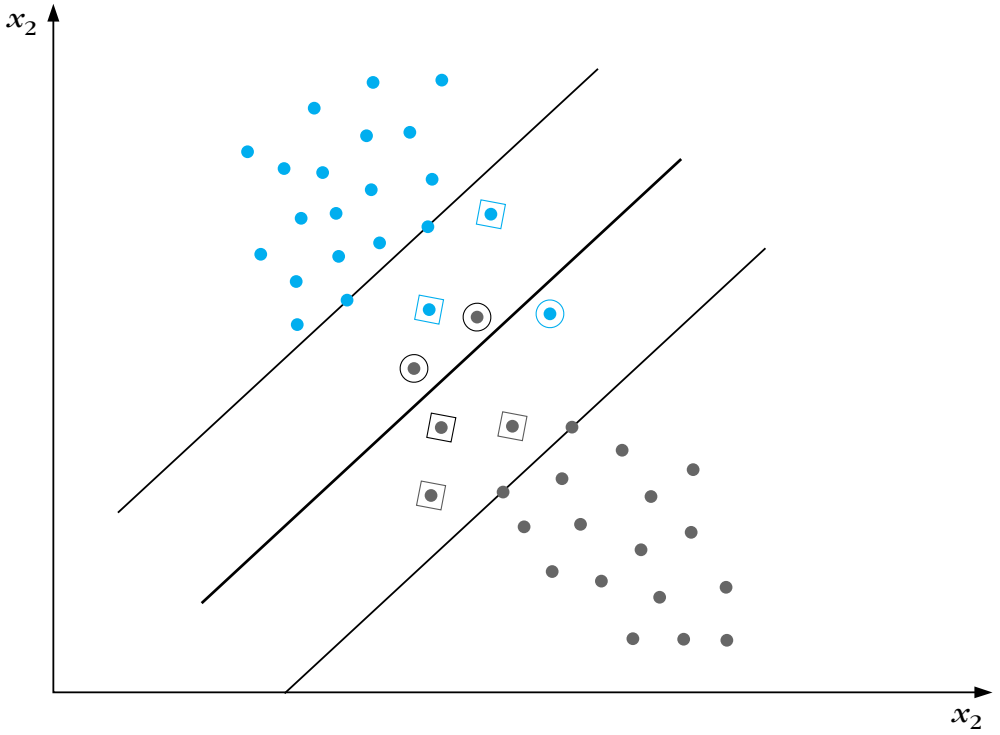


Soft Margin Classification (3)

- **Slack variables** $\xi_i \geq 0$

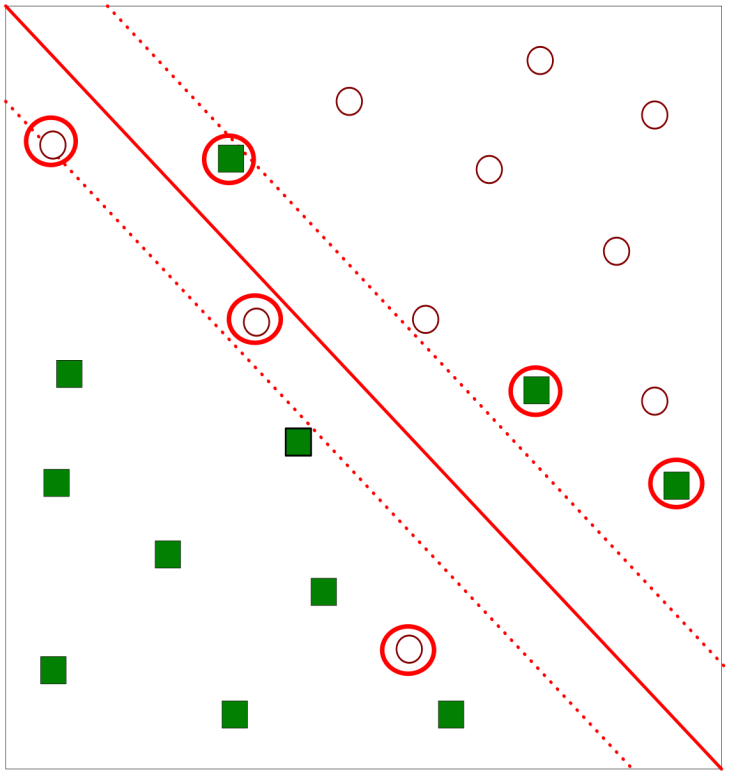
$$y_i(w^T x + b) \geq 1 - \xi_i$$

  - Samples outside the margin (correctly classified): $\boldsymbol{\xi_i = 0}$
  - Margin Violation: $\boldsymbol{0 < \xi_i \leq 1}$
  - Misclassified samples: $\boldsymbol{\xi_i > 1}$



Soft Margin Classification (4)

- **Slack variables** $\xi_i \geq 0$
  - ξ_i allows misclassification of difficult or noisy samples
  - Resulting margin is called **soft margin**
  - ξ_i is based on the output of the discriminant function $w^T x + b$
  - ξ_i approximates the number of misclassified samples



Soft Margin Classification (5)

- The optimization problem becomes

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \quad s.t. \;\; y_i(w^T x_i + b) \geq 1 - \xi_i \quad \forall i = 1, 2, \ldots, N$$

  - This is still a quadratic optimization problem
  - There is a unique minimum
  - Every constraint can be satisfied if ξ_i is sufficiently large

Soft Margin Classification (7)

- Transfer the optimization problem into its dual problem as:

$$\max_{\lambda_i} W(\lambda) = \max_{\lambda_i} \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1,j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j$$

$$s.t. \quad 0 \leq \lambda_i \leq C, \qquad \sum_{i=1}^{N} \lambda_i y_i = 0 \quad \forall \lambda_i$$

Soft Margin Classification (8)

- The dual problem is identical to separable case except for an upper bound C on λ_i
- x_i with non-zero λ_i will be support vectors
- Solution to the dual problem is:

$$w = \sum_{i=1}^{N} \lambda_i y_i x_i \qquad b = y_k(1 - \xi_k) - \sum_{i=1}^{N} \lambda_i y_i x_i^T x_k \quad \forall \lambda_k > 0$$

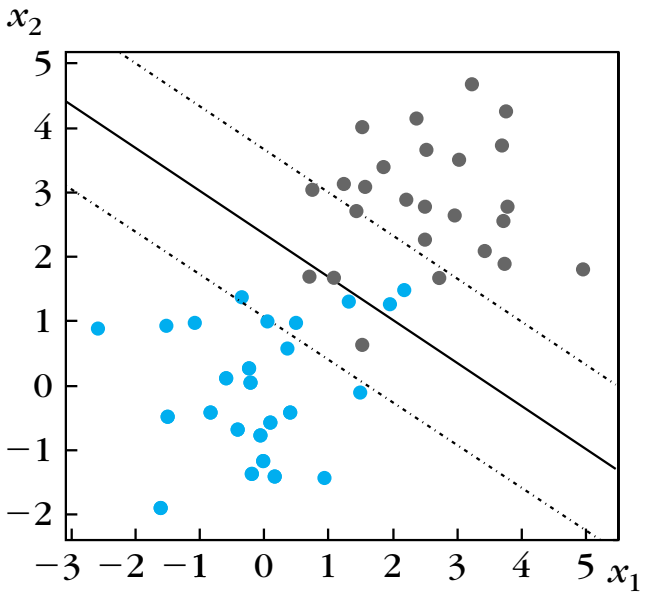
Soft Margin Classification (6)

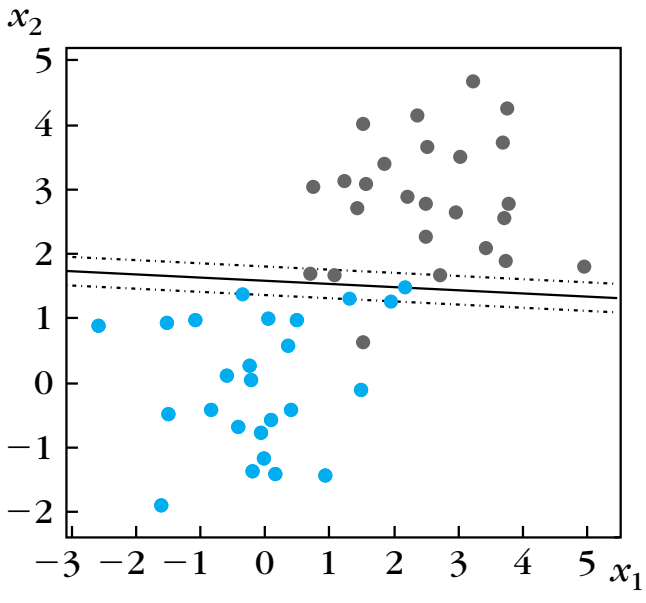
- Parameter C is a user-selected regularization parameter:
  - C is a tradeoff parameter between the error and margin
  - C can be viewed as a way to control overfitting
  - Small C allows constraints to be easily ignored ⇛ **large margin**
  - Large C means a higher penalty to errors, making constraints hard to ignore ⇛ **narrow margin**
  - $C = \infty$ enforces all constraints: **hard margin**

Soft Margin Classification (9)

- Multi-class generalization
  - Transfer multi-class into multiple two-class problems (one against all others)
- Observe the effect of different C values in non-separable classes

Small C





Large C

Outline

- Motivation
- SVM
- Soft Margin Classification
- Nonlinear SVM
- Conclusion

Nonlinear SVM (1)

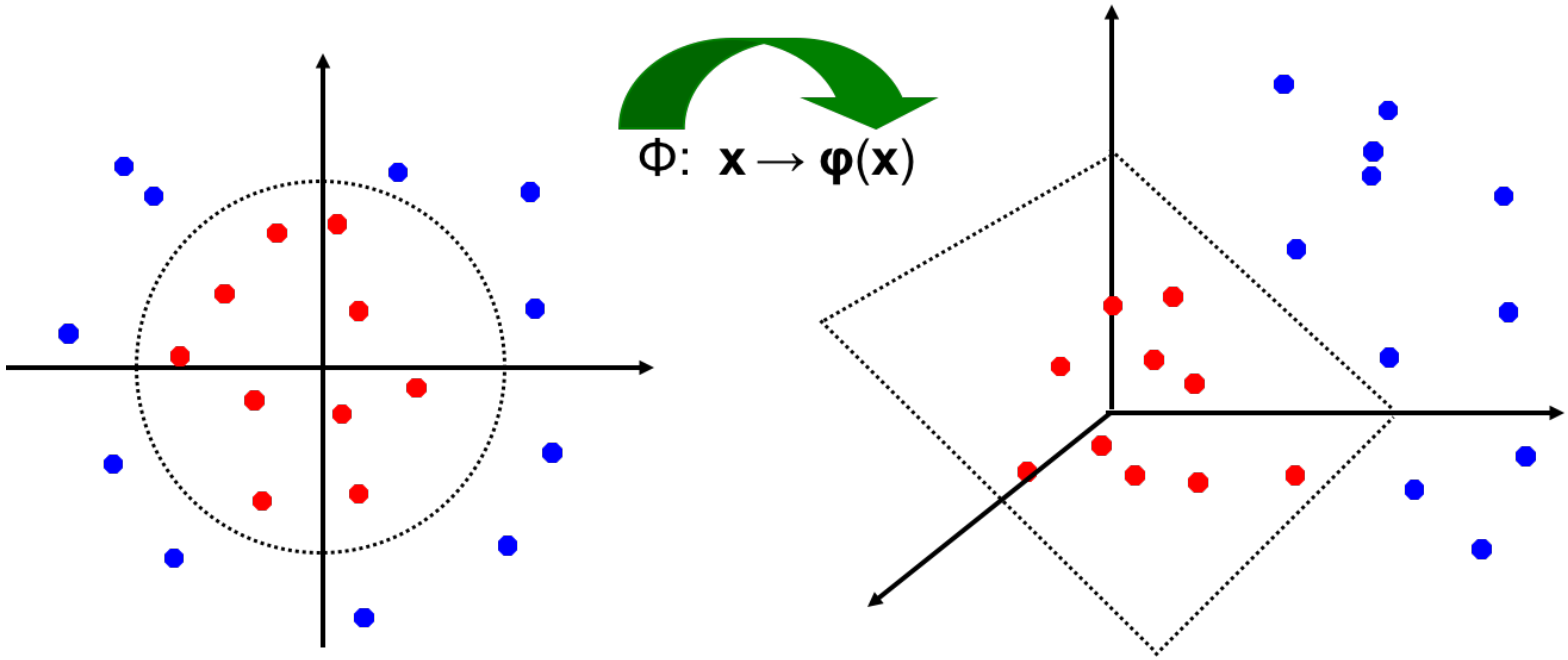
- Linear SVM
  - The classifier is a separating hyperplane
  - Most important training points are support vectors
  - Support vectors define the hyperplane
  - Quadratic optimization algorithms can identify which training points x_i are support vectors with non-zero Lagrangian multipliers λ_i

Nonlinear SVM (2)

- Non-linear SVM
  - Datasets may be too hard for linear separation
  - Transform data into a higher dimensional space **H** via a mapping function **Φ** such that the data appears of the form $\Phi(x_i)\Phi(x_j)$
  - Linear separation in **H** is equivalent to nonlinear separation in the original input space

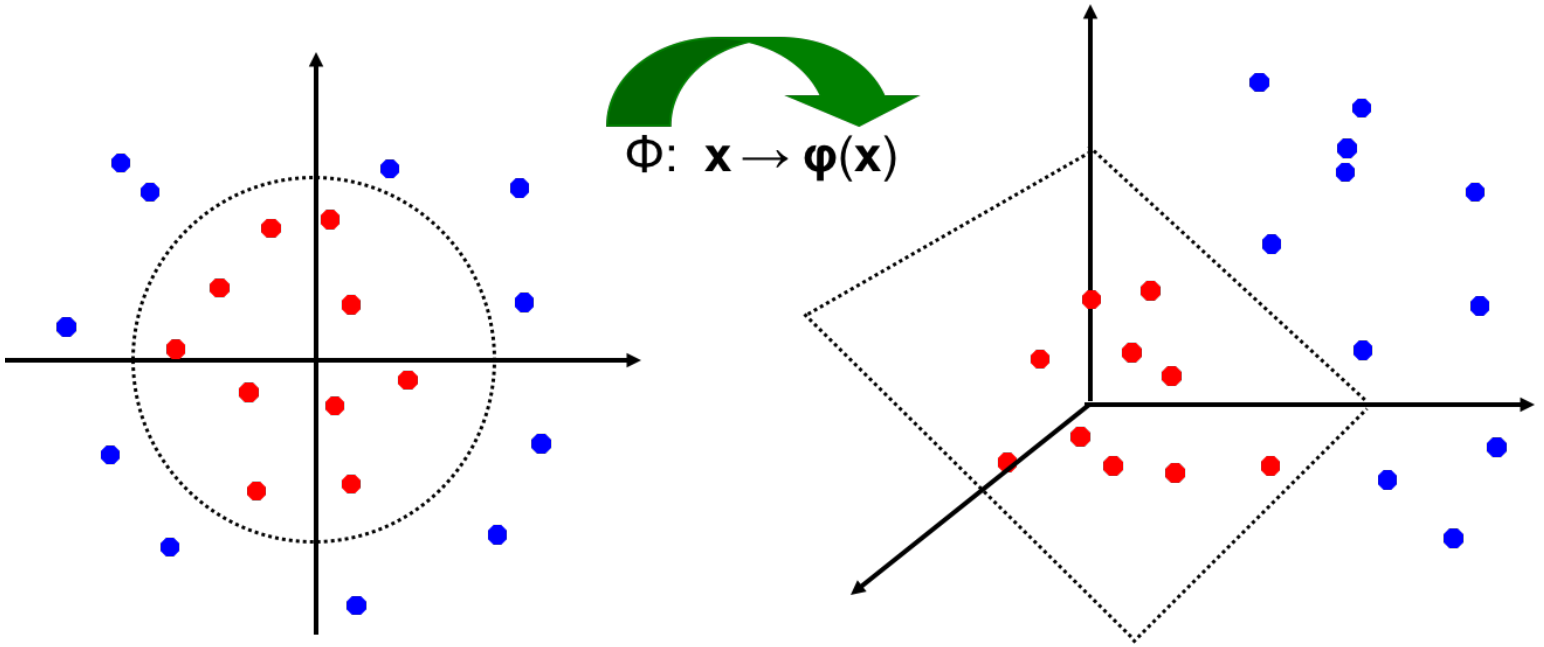
Nonlinear SVM (3)

- General idea:
  - The original feature space can always be mapped to some higher-dimensional feature space $\Phi(x)$ where the training set is separable.



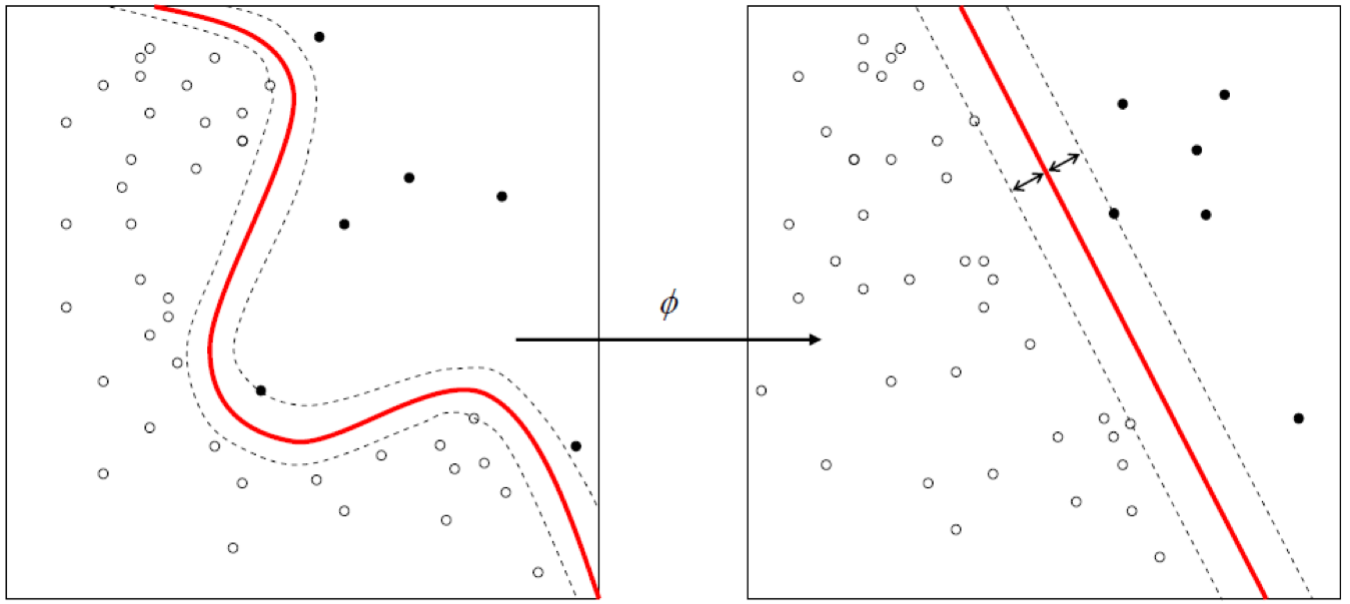
Nonlinear SVM (4)

- Possible transformation problems:
  - High computation burden due to high dimensionality
  - Hard to obtain a good estimation



Nonlinear SVM (5)

- SVM solves these two issues simultaneously
  - "Kernel tricks" for efficient computation
  - Minimize $\|w\|^2$ can lead to a "good" classifier



Nonlinear SVM (6)

- Kernel function
  - Inner product between vectors: $K(x_i, x_j) = x_i^T x_j$
  - If each data point is mapped into high-dimensional space via transformation Φ: $x \longrightarrow \varphi(x)$, inner product becomes: $K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$
  - Kernel function is equivalent to an inner product in feature space
  - Kernel function implicitly maps data to a high-dimensional space without computing each $\varphi(x)$ explicitly.

Nonlinear SVM (7)

- **Kernel example**

2D vectors $x = (x_1; x_2)$, let $K(x_i, x_j) = (1 + x_i^T x_j)^2$, need to show that $K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$

$$K(x_i, x_j) = (1 + x_i^T x_j)^2 = \left(1 + (x_{i1}, x_{i2})^T (x_{j1}, x_{j2})\right)^2$$

$$= 1 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} + x_{i1}^2 x_{j1}^2 + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1}x_{i2}x_{j2}$$

$$= \left(1 \ \sqrt{2}x_{j1} \ \sqrt{2}x_{j2} \ x_{j1}^2 \ x_{j2}^2 \ \sqrt{2}x_{j1}x_{j2}\right)\left(1 \ \sqrt{2}x_{i1} \ \sqrt{2}x_{i2} \ x_{i1}^2 \ x_{i2}^2 \ \sqrt{2}x_{i1}x_{i2}\right)^T$$

$$= \varphi(x_i)^T \varphi(x_j) \quad \text{where} \quad \varphi(x) = \left(1 \ \sqrt{2}x_1 \ \sqrt{2}x_2 \ x_1^2 \ x_2^2 \ \sqrt{2}x_1 x_2\right)^T$$

Nonlinear SVM (8)

- Examples of kernel functions
  - Linear: $K(x_i, x_j) = x_i^T x_j$
    - Mapping $\Phi: x \longrightarrow \varphi(x)$, where $\varphi(x)$ is x itself
  - Polynomial of power p: $K(x_i, x_j) = (1 + x_i^T x_j)^p$
    - Mapping $\Phi: x \longrightarrow \varphi(x)$, where $\varphi(x)$ has $\binom{d+p}{P}$ dimensions
  - Sigmoid: $K(x_i, x_j) = \tanh(\beta_0 x_i^T x_j + \beta_1)$

Nonlinear SVM (9)

- Examples of kernel functions (Cont.)
  - Gaussian (Radial-basis function network): $K(x_i, x_j) = exp\left(\frac{\|x_i - x_j\|^2}{2\sigma}\right)$
    - Mapping $\Phi: x \longrightarrow \varphi(x)$, where $\varphi(x)$ is *infinite-dimensional*
    - Every point is mapped to a function (a Gaussian)
    - Closely related to radial basis function neural networks
    - Combination of functions for support vectors is the separating hyperplane

Nonlinear SVM (10)

- The dual problem formation:

$$\max_{\lambda_i} W(\lambda) = \max_{\lambda_i} \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1,j=1}^{N} \lambda_i \lambda_j y_i y_j K(x_i, x_j)$$

$$s.t. \sum_{i=1}^{N} \lambda_i y_i = 0 \quad \lambda_i \geq 0 \quad \forall \lambda_i$$

- The solution is:

$$f(y) = \sum_{i=1}^{N} \lambda_i y_i K(x_i, y) + b$$

- Optimization techniques for finding λ_i remain the same!

Nonlinear SVM (11)

- Numerical example

Suppose we have five 1D data points: $x_1 = 1,\ x_2 = 2,\ x_3 = 3,$

$x_4 = 5, x_5 = 6$; x_1, x_2 and x_5 are Class 1, x_3 and x_4 are Class 2;

$\Rrightarrow\ \ y_1 = 1,\ y_2 = 1,\ y_3 = -1,\ y_4 = -1,$ and $y_5 = 1$

- We use the polynomial kernel of degree 2: $K(x, y) = (xy + 1)^2$, and

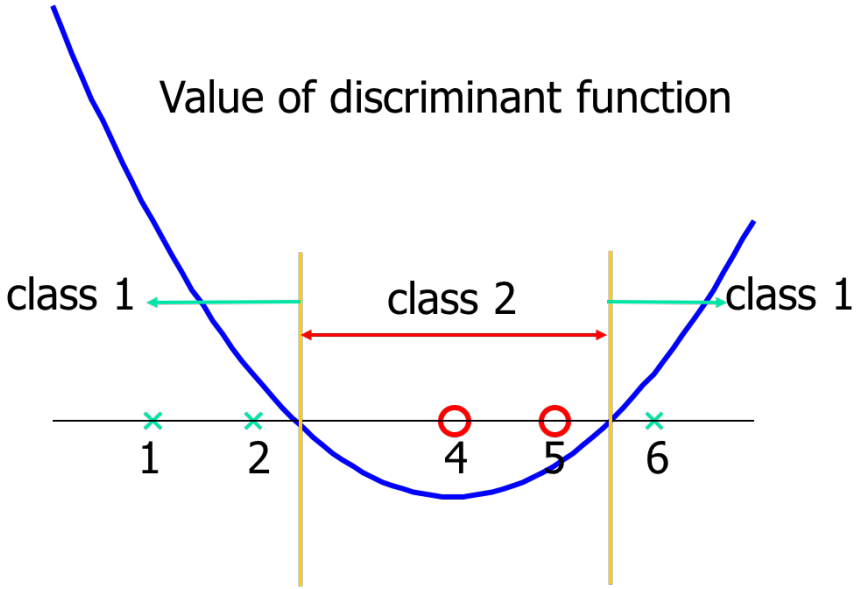
  C is set to 100

Nonlinear SVM (12)

- We first find λ_i $(i = 1, \ldots, 5)$ by:

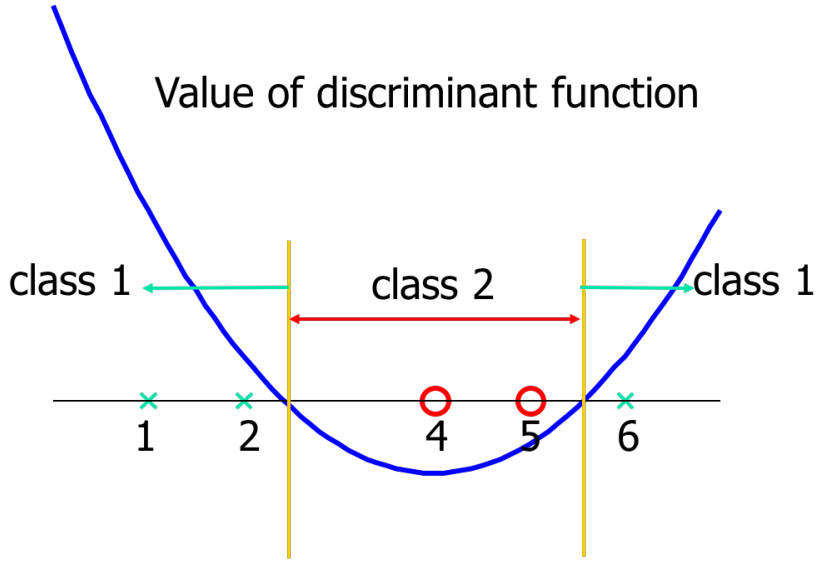
$$\max_{\lambda_i} \sum_{i=1}^{5} \lambda_i - \frac{1}{2} \sum_{i=1,j=1}^{5} \lambda_i \lambda_j y_i y_j \left(x_i x_j + 1\right)^2$$

$$s.t. \sum_{i=1}^{5} \lambda_i y_i = 0 \quad 0 \le \lambda_i \le 100$$



Nonlinear SVM (13)

- Using a quadratic problem solver, we obtain
  - $\lambda_1 = 0,\ \lambda_2 = 2.5,\ \lambda_3 = 0,\ \lambda_4 = 7.333,\ \lambda_5 = 4.833$
  - The support vectors are $\{x_2 = 2,\ x_4 = 5,\ x_5 = 6\}$
- The discriminant function is:



$$f(y) = \sum_{i=1}^{N} \lambda_i y_i K(x_i, y) + b = \sum_{i=1}^{N} \lambda_i y_i (xy + 1)^2 + b$$

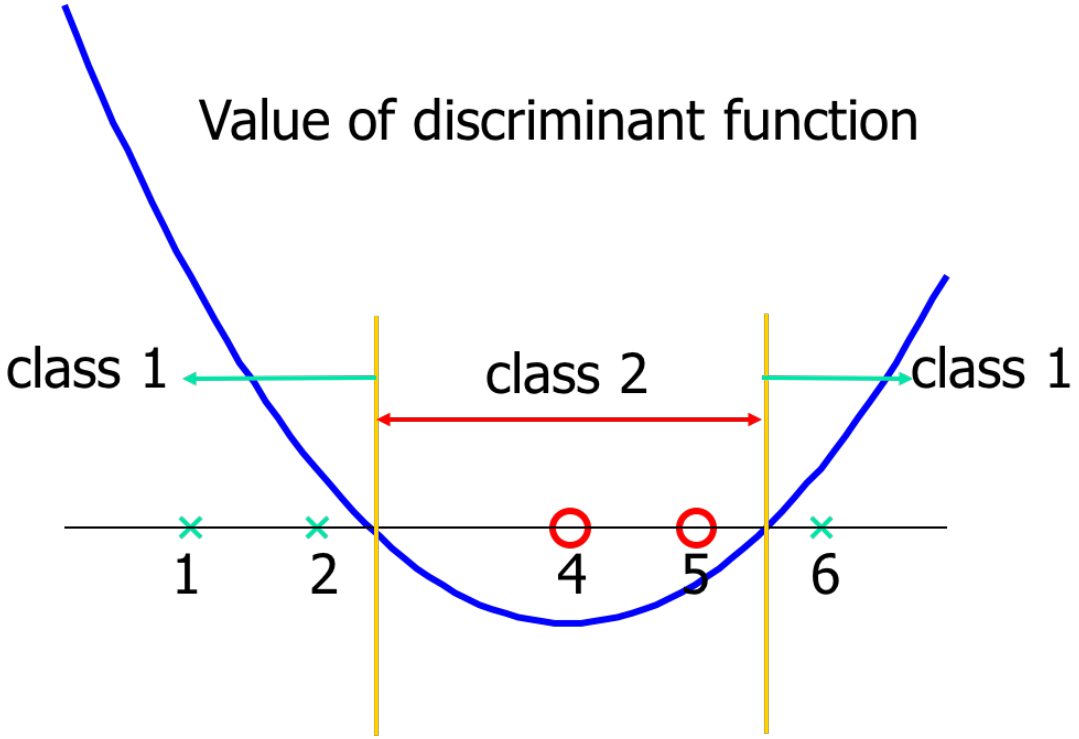
$$= 2.5(1)\times(2y+1)^2 + 7.333(-1)\times(5y+1)^2 + 4.833(1)\times(6y+1)^2 + b$$

$$\Rightarrow\ f(y) = 0.6667y^2 - 5.333y + b \quad \Rightarrow f(6) = 1 \Rightarrow\ \ b = 9$$

Nonlinear SVM (14)

- We obtain the discriminant function:

$$f(y) = 0.6667y^2 - 5.333y + 9$$



Nonlinear SVM (15)

- SVM properties:
  - Flexibility in choosing a similarity function
  - Sparseness of solution when dealing with large data sets
    - Only support vectors are used to specify the separating hyperplane
  - Ability to handle large feature spaces
    - Complexity does not depend on the dimensionality of the feature space
  - Overfitting can be controlled by soft margin approach

Nonlinear SVM (16)

- SVM properties (Cont.)
  - Nice path property: a simple convex optimization problem
    - Guarantee to converge to a single global solution
  - Feature selection
- SVM applications
  - hand-written character recognition
  - image classification
  - Bioinformatics
    - Protein classification
    - Cancer classification

Outline

- Motivation
- SVM
- Soft Margin Classification
- Nonlinear SVM
- Conclusion

Conclusion (1)

- SVM
  - SVMs use a single hyperplane via maximizing the margin $\rho = \frac{2}{\|w\|}$ to distinguish two classes
  - Support vectors are training samples that are closest to and to specify the hyperplane
  - The quadratic optimization problem subject to linear constraints and has a unique minimum
  - Solve the quadratic optimization problem by transforming it into its dual problem

Conclusion (2)

- Soft Margin Classification
  - Introducing slack variables $\xi_i \geq 0$ to allow misclassification of difficult or noisy samples
  - Soft margin classification is also a quadratic optimization problem and has a unique minimum as well.
  - The only parameter C is a user-selected regularization parameter and controls the tradeoff between the error and margin
  - The dual problem is identical to the separable cases except for an upper bound C on λ_i

Conclusion (3)

- Nonlinear SVM
  - Dealing with the data that is too hard for linear separation
  - Using a kernel function Φ to map the data to a higher-dimensional feature space H for linear separation
  - A kernel function implicitly maps data to a high-dimensional space without computing each $\varphi(x)$ explicitly.
  - Kernels are semi-positive definite symmetric Gram matrices