Neural Network

Chapter 3 Learning & Perceptron

西安电子科技大学
XIDIAN UNIVERSITY

Outline

01 Artificial Neural Network

02 Loss Function

03 Rosenblatt Perceptron

04 Learning

01

Artificial Neural Network

Artificial Neural Networks

- An ANN can:
  1. compute *any computable* function, by the appropriate selection of the network topology and weights values.
  2. learn from experience!
     - Specifically, by trial-and-error

Learning by trial-and-error

Continuous process of:

- Trial:

Processing an input to produce an output (In terms of ANN: Compute the output function of a given input)

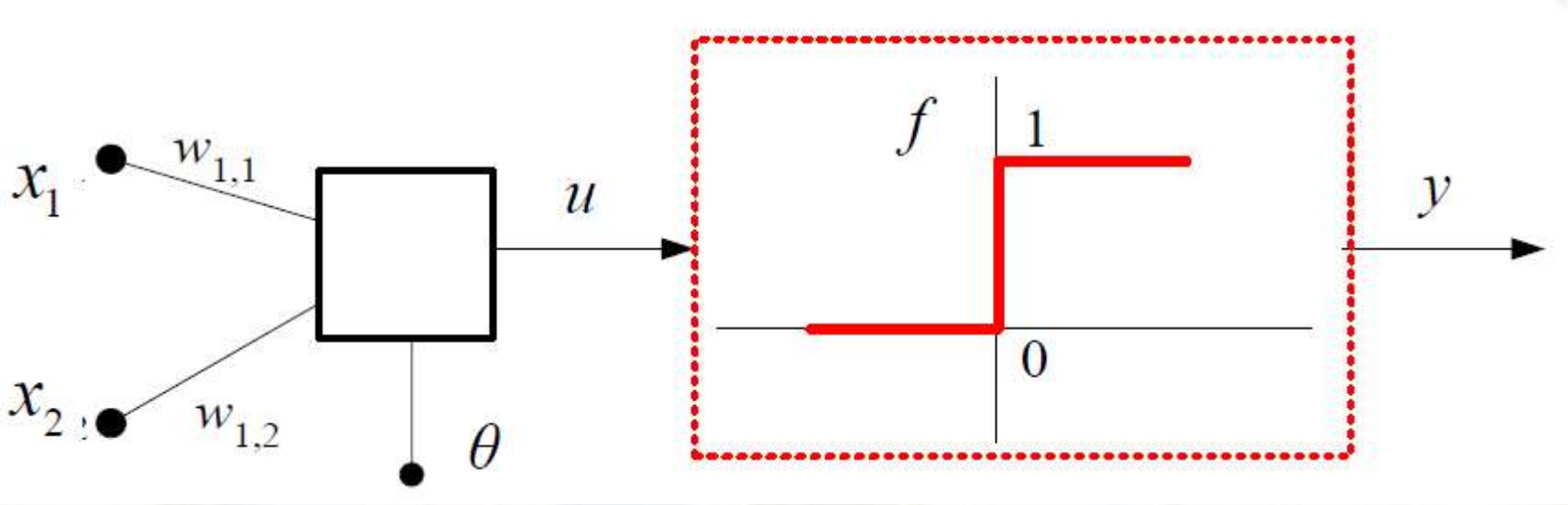
- Evaluate:

Evaluating this output by **comparing the actual output with the expected output**.

- Adjust:

Adjust the *weights*.

Perceptron (M-P)

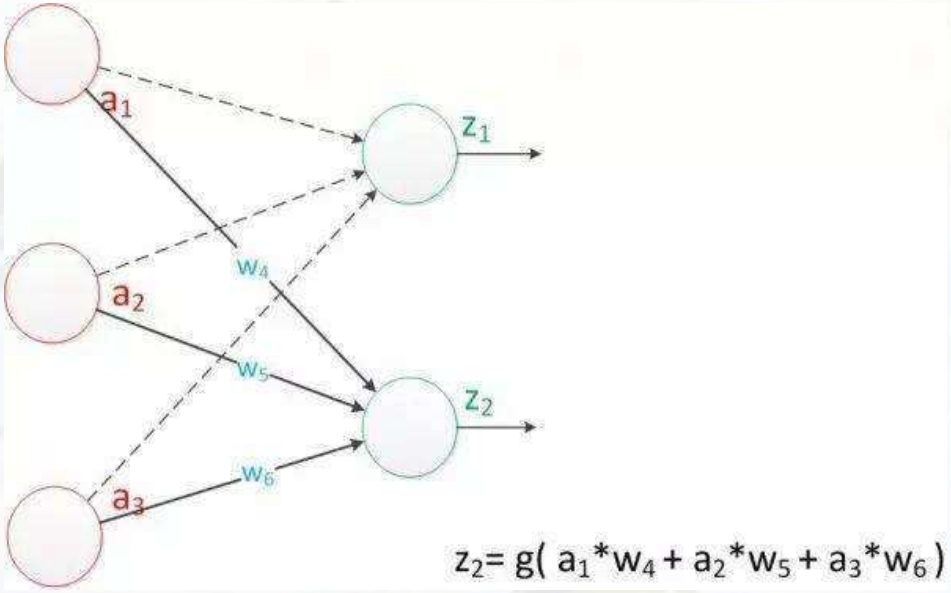


$$y = f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \qquad \text{sgn}(x) = \begin{cases} 1 & x \geq 0 \\ -1, & x < 0 \end{cases}$$

$$y = hard\lim(u) = \text{hardlim}(WX + \theta) = \text{hardlim}(w_{1,1}x_1 + w_{1,2}x_2 + \theta)$$

Single Layer Neural Network

- The neuron on the left column (a1,a2,a3) only have the input function。
- The neuron on the right column is the calculation function and output。



Limitations of Simple Neural Networks

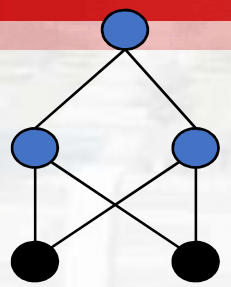
The Limitations of Perceptrons

(Minsky and Papert, 1969)

- Able to form only *linear discriminate functions*; *i.e.* classes which can be divided by a line or hyper-plane
- Most functions are more complex; *i.e.* they are *non-linear* or not *linearly separable*

EXAMPLE

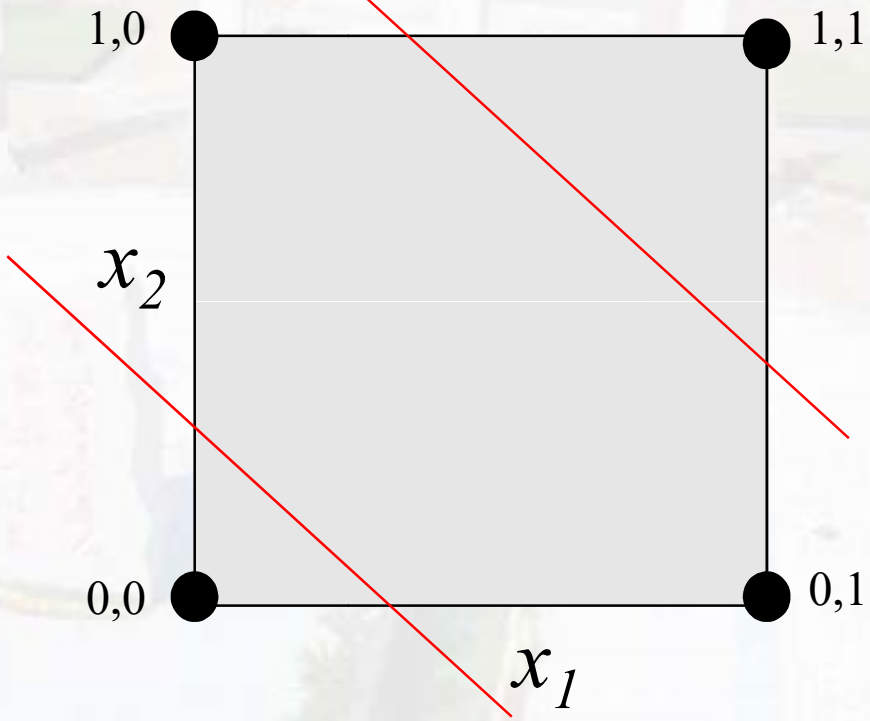
Multi-layer
Neural Network



Hidden layer
of neurons

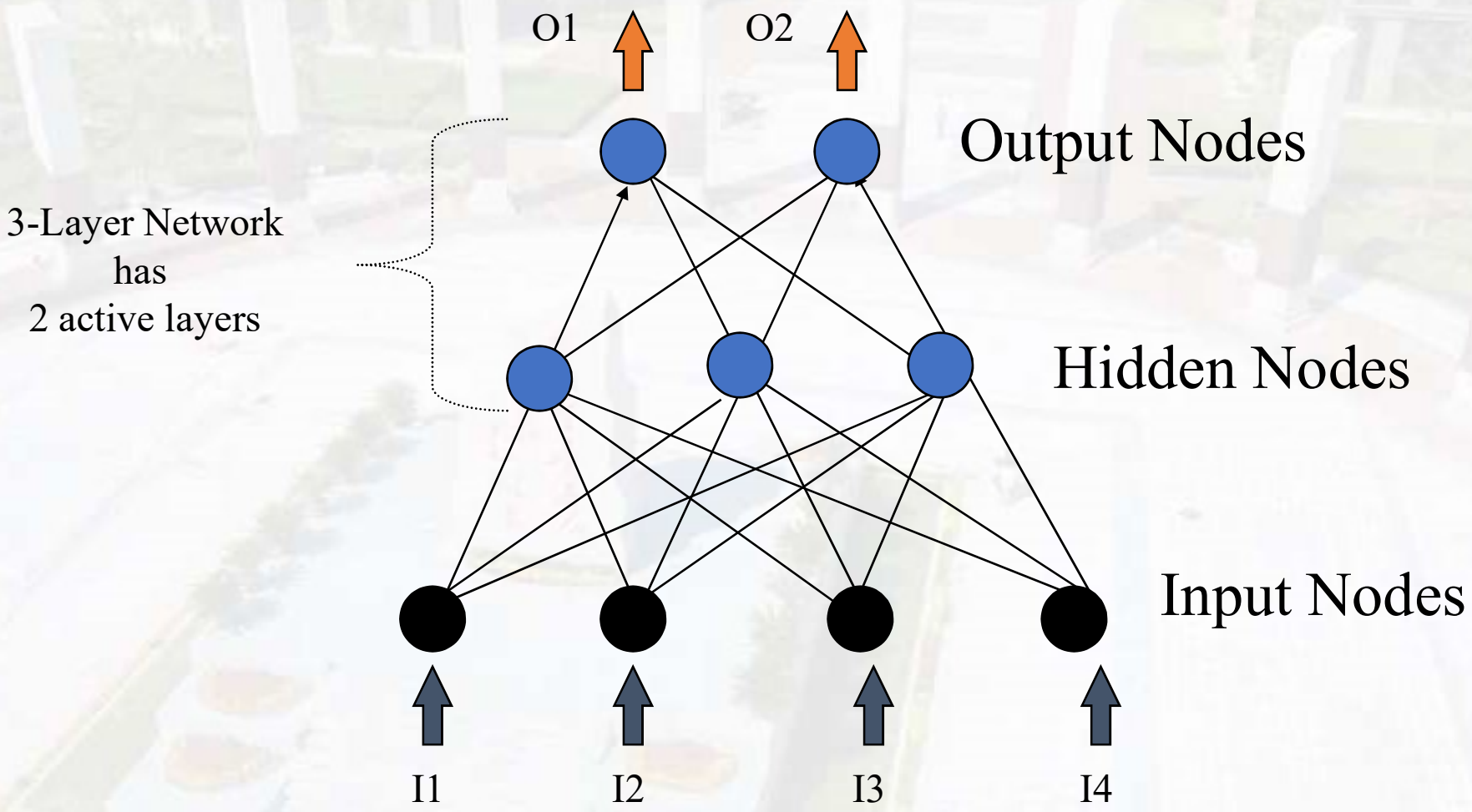
Logical XOR
Function

x1	*x2*	*y*
0	0	0
0	1	1
1	0	1
1	1	0

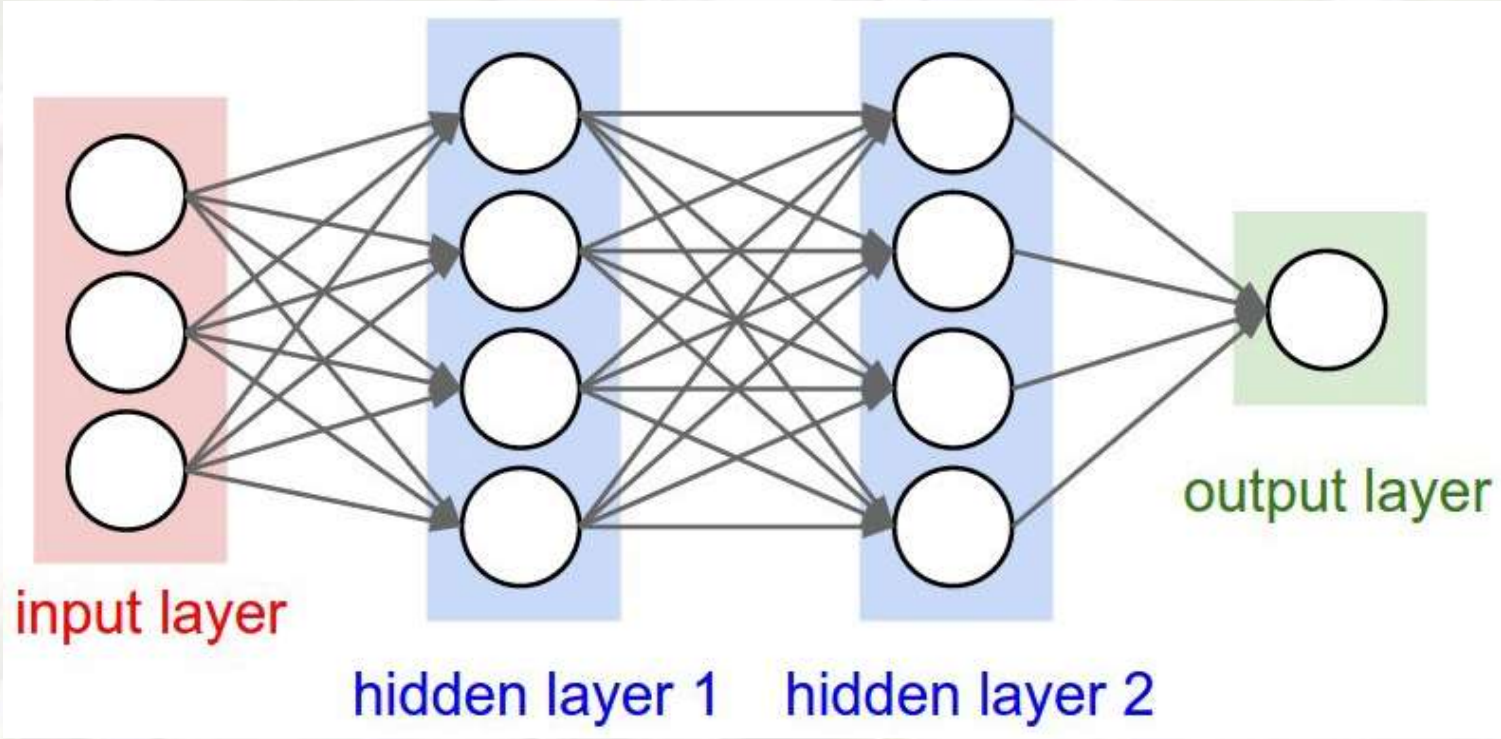


Two neurons are need! Their combined results can produce good classification.

Two-Layer Neural Network

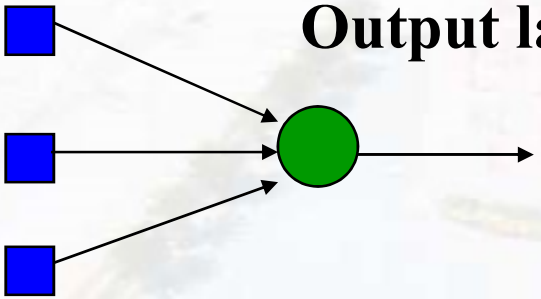


Three-Layer Neural Network

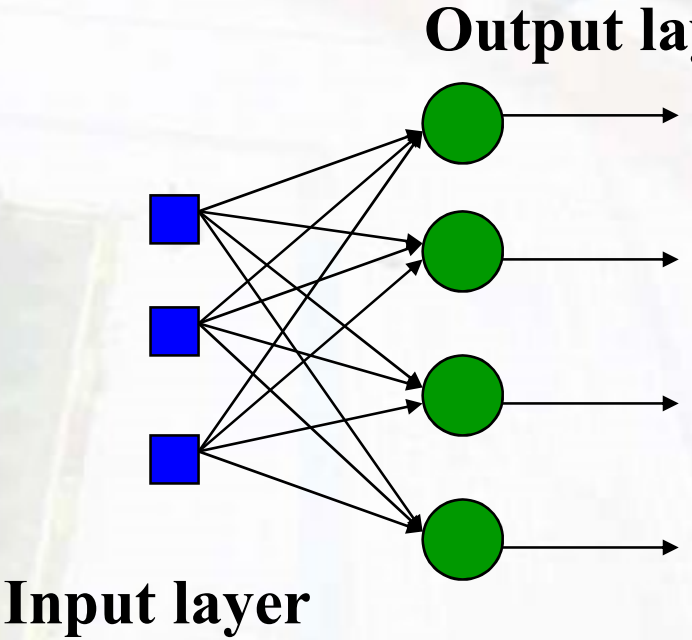


Multi-layer Feed-forward ANNs

- Single Output Perceptron (M-P)

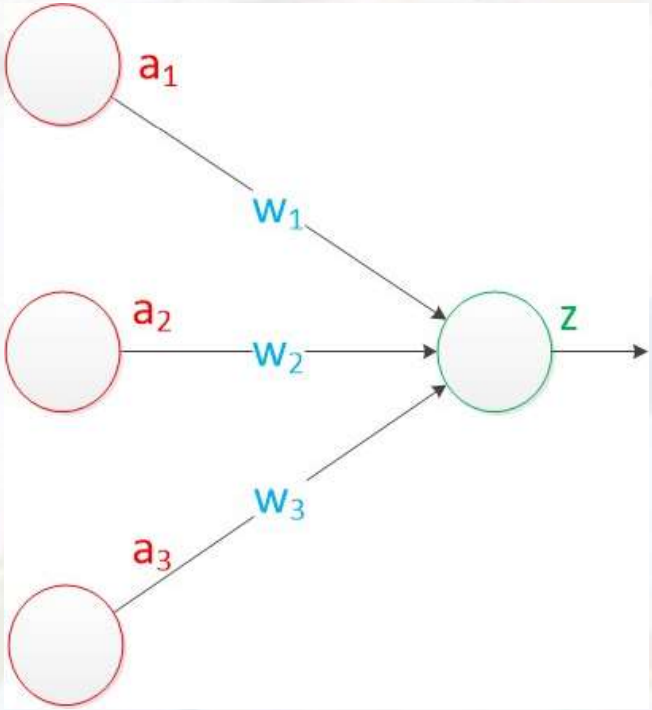


- Multi Output Perceptron (Rosenblatt)

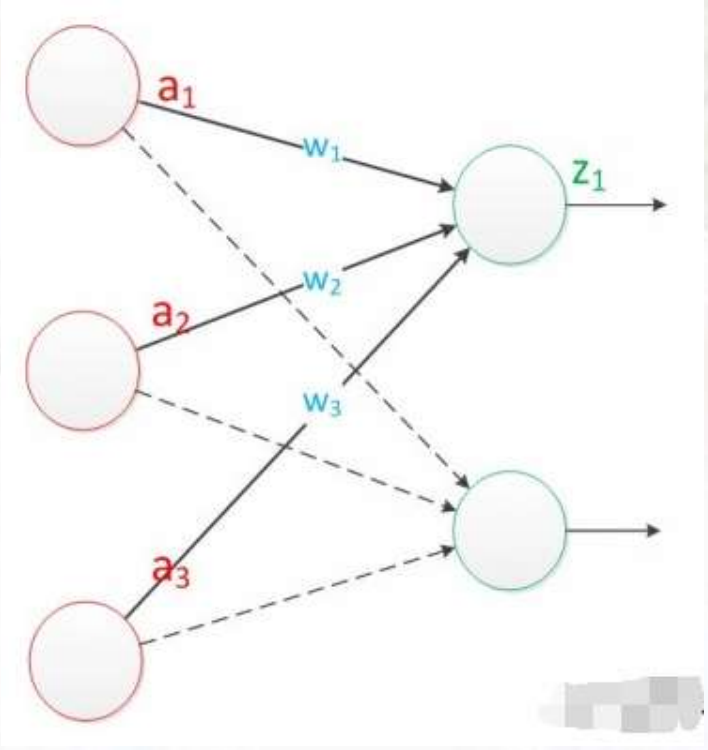


Multi-layer Feed-forward ANNs

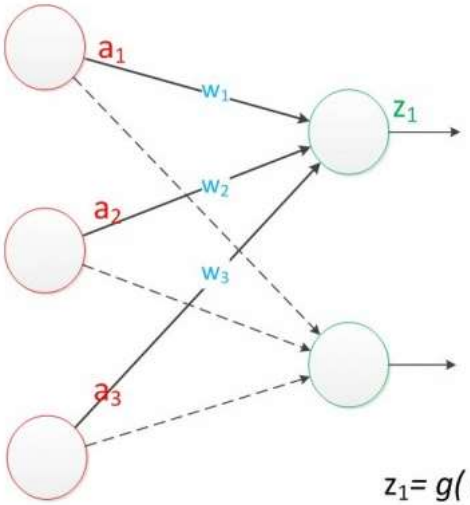
- Single Output Perceptron (M-P)



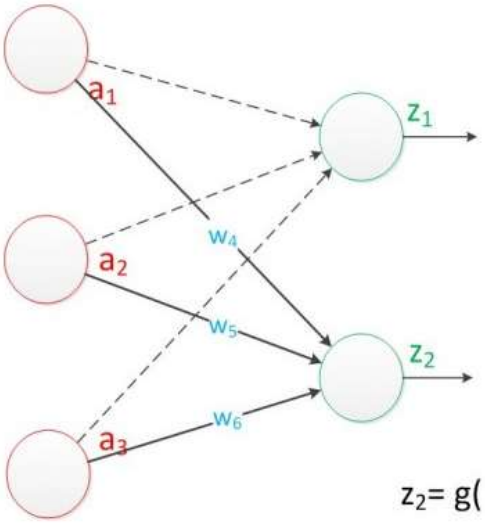
- Multi Output Perceptron (Rosenblatt)



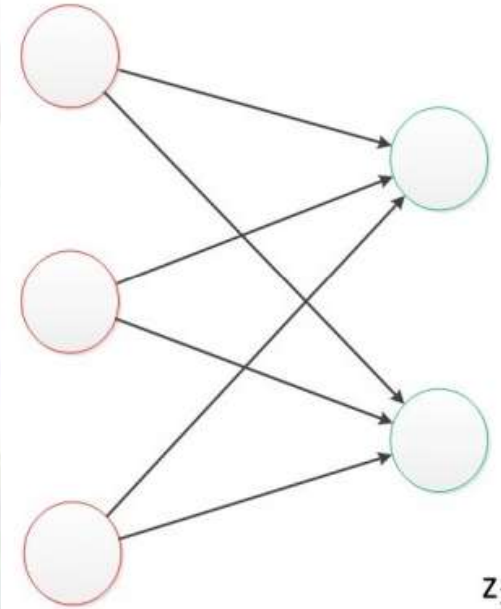
Multi-layer Feed-forward ANNs



$z_1 = g(a_1*w_1 + a_2*w_2 + a_3*w_3)$



$z_2 = g(a_1*w_4 + a_2*w_5 + a_3*w_6)$



$z_1 = g(a_1*w_1 + a_2*w_2 + a_3*w_3)$

$z_2 = g(a_1*w_4 + a_2*w_5 + a_3*w_6)$

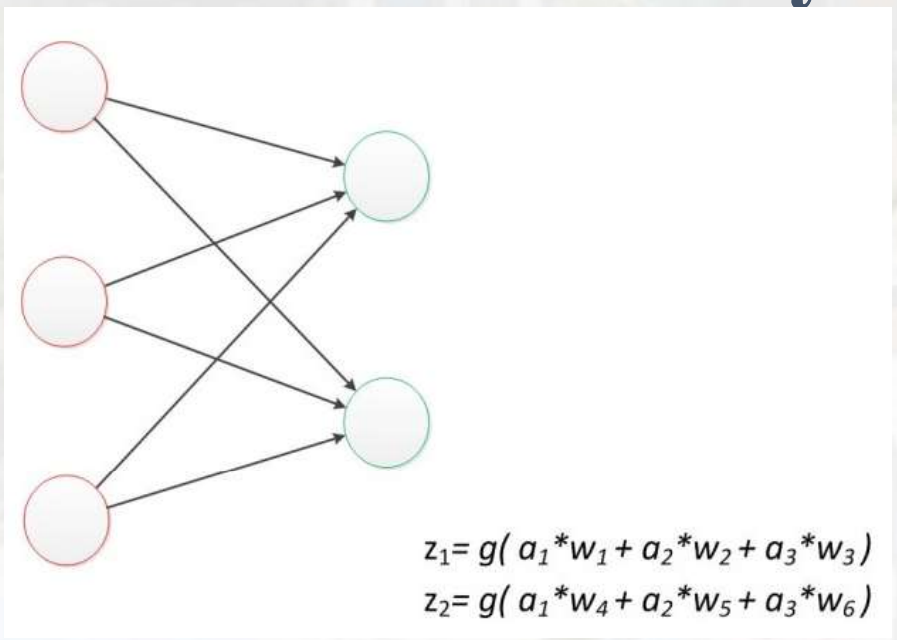
Multi-layer Feed-forward ANNs

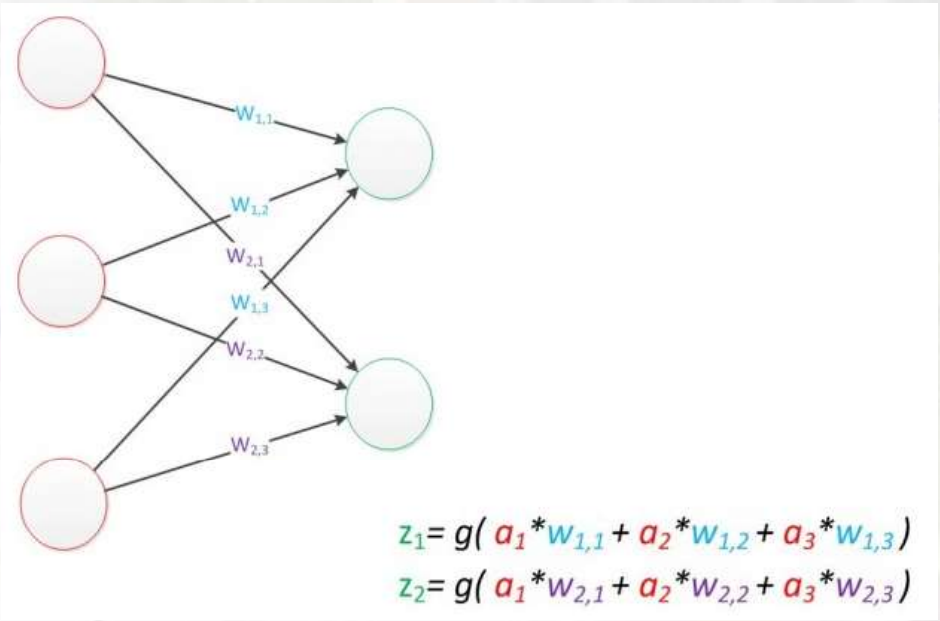
Over the 15 years (1969-1984) some research continued ...

- *hidden layer* of nodes allowed combinations of linear functions
- *non-linear activation functions* displayed properties closer to real neurons:
  - output varies continuously but not linearly
  - differentiable .... *sigmoid* $f(a)=1/(1+e^{-a})$

  non-linear ANN classifier was possible

Multi-layer Feed-forward ANNs





We use $W_{x,y}$ to represent the weight.

Specifically, ***X*** represent the ***Previous layer*** neuron number.

Y represent the ***Next layer*** neuron number.

The sequence of the neuron number is from top to bottom.

02

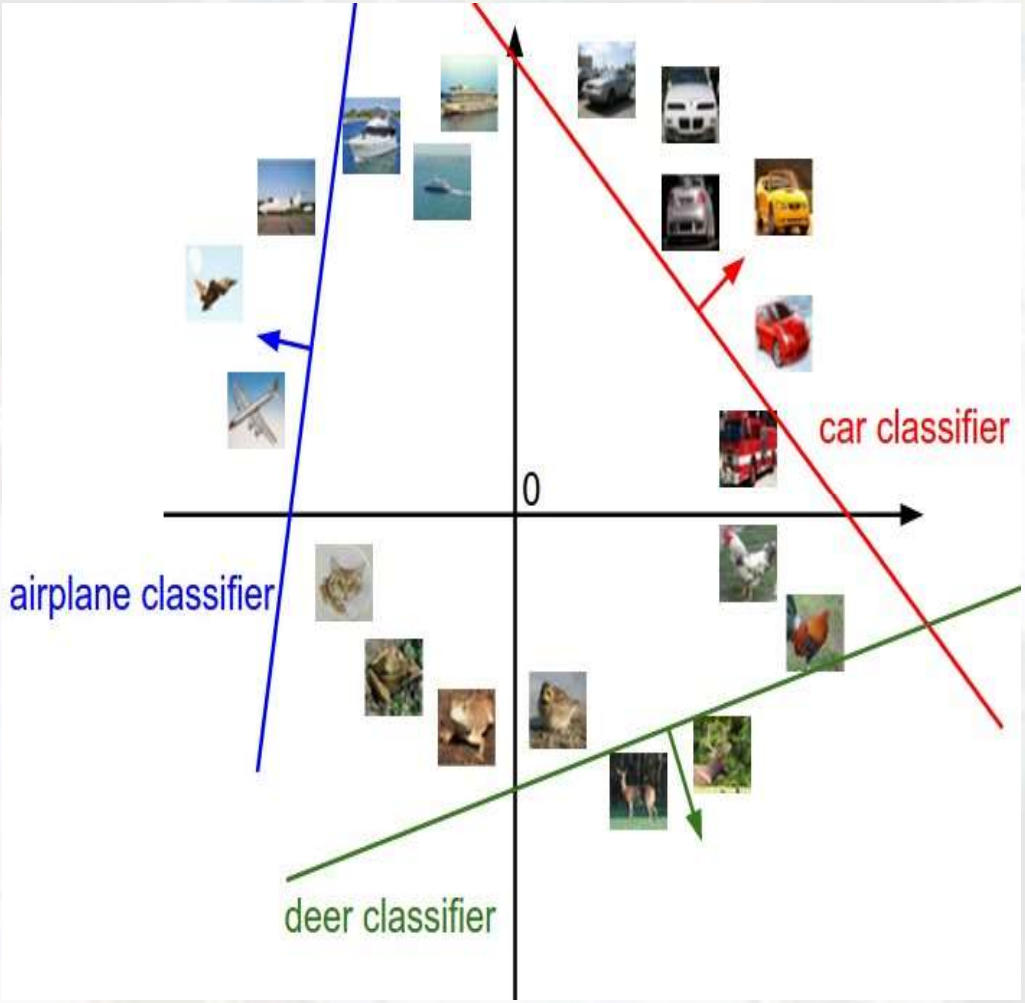
Loss Function

Loss Function

$$f(x_i, W, b) = Wx_i + b$$

Example trained weights of a linear classifier trained on CIFAR-10:

Loss Function



$$f(x_i, W, b) = Wx_i + b$$

[32x32x3]
array of numbers 0...1
(3072 numbers total)

Loss Function(损失函数):

Definition: is the error based on single example

(1) 0-1损失函数（0-1 loss function）

$$L(f(x),y)=\begin{cases}1, & y\neq f(x)\\ 0, & y=f(x)\end{cases}$$

(2) 平方损失函数（quadratic loss function）

$$L(f(x),y)=(f(x)-y)^2$$

(3) 绝对值损失函数（absolute loss function）

$$L(f(x),y)=|f(x)-y|$$

Cost Function(代价函数):

Definition: is the average error based on whole train examples, as well as the average of all lose function.

(1)均方误差 (Mean Squared Error)

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - f(x^{(i)}))^2$$

(2)平均绝对误差 (Mean Absolute Error)

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|y^{(i)} - f(x^{(i)}\right|$$

⟸ **Common use in Regression**

(3)交叉熵代价函数 (Cross-entropy)

$$H(p,q) = -\sum_{i=1}^{N} p(x^{(i)})\log q(x^{(-i)})$$

⟸ **Common use in Classification**

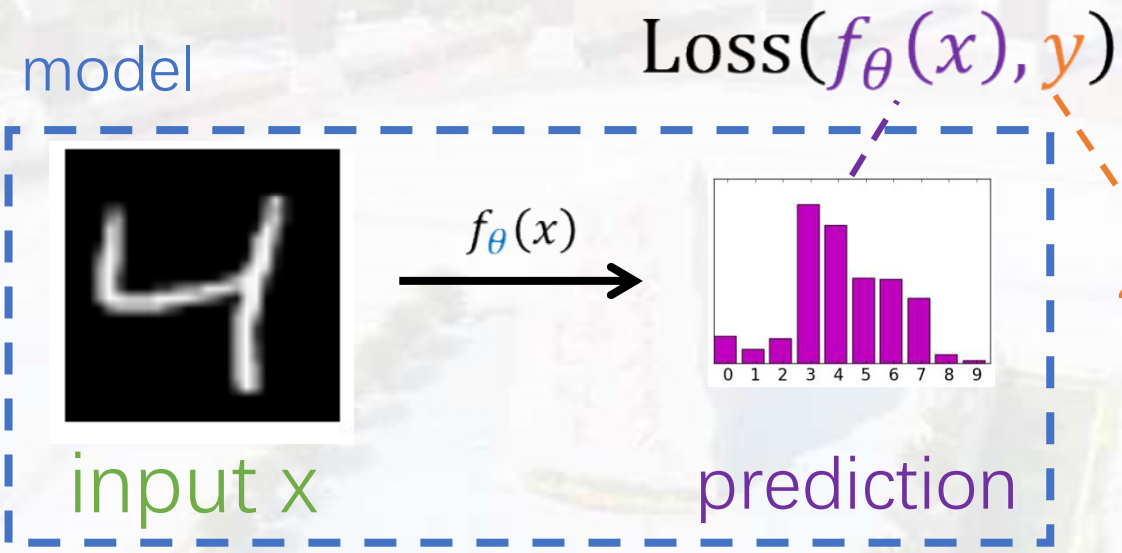
Turth distribution of probobality

Trained distribution of probobality

Objective Function(目标函数):

Definition: is the average error based on whole train examples, as well as the average of all lose function.

- Objective: minimize



θ: paramet

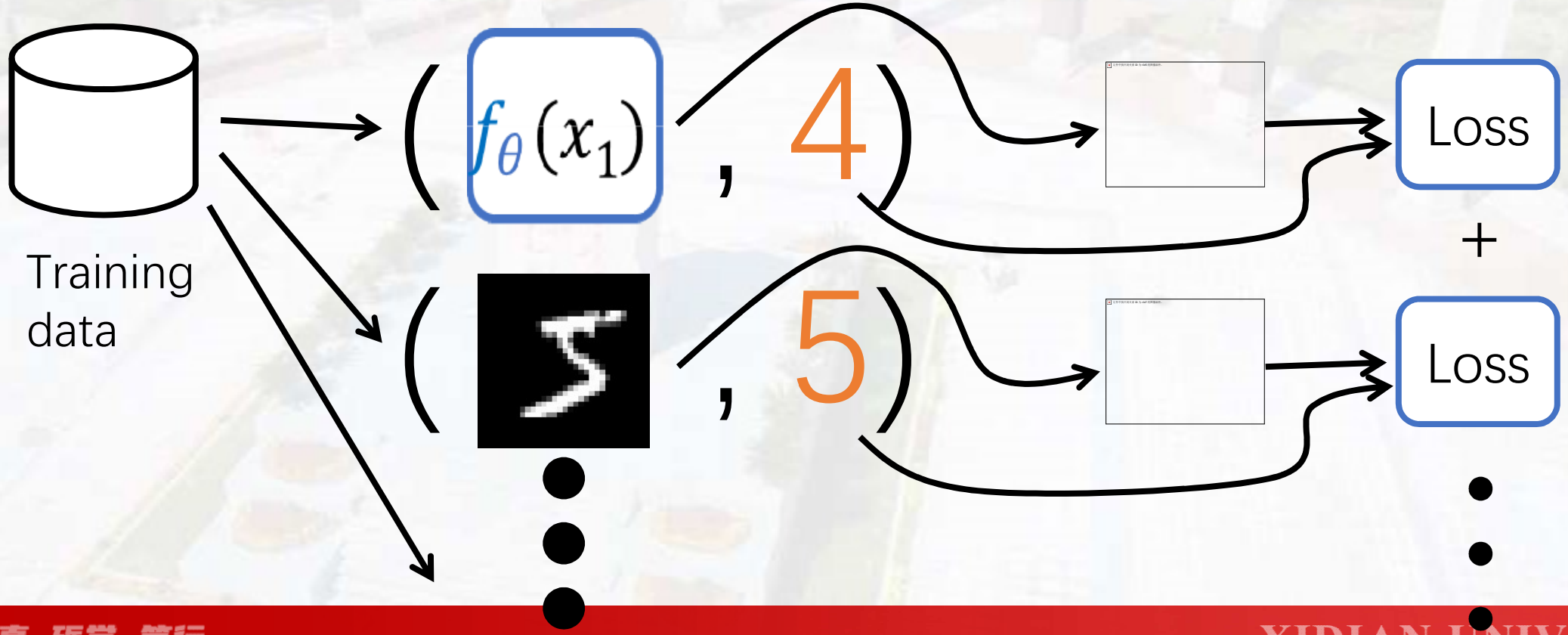
for a randomly chosen *(x,y)* from some distribution *D*.

- We don't know the distribution *D*.
- We only have access to (training) samples from *D*.

Loss Function

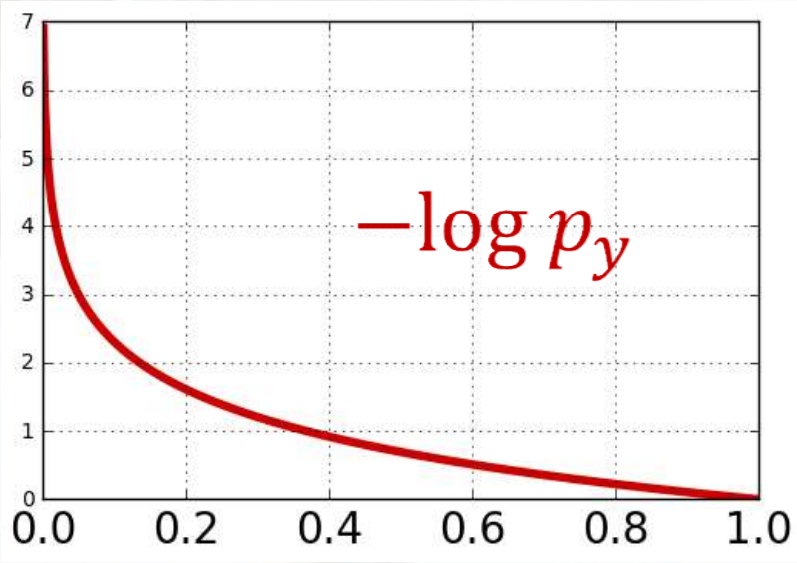
Objective function: $L(\theta) = \frac{1}{n}\sum_{i=1}^{n} Loss(f_\theta(x_i), y_i)$

Approximate the unknown distribution D with the training data average



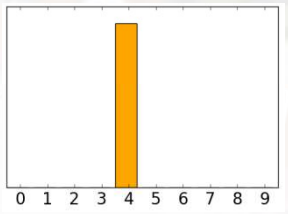
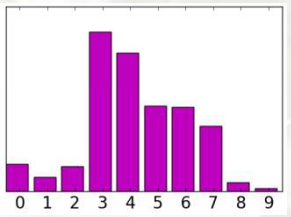
Cross-entropy loss

- Interpretation 1: you pay penalty



where y is the correct label.

- Interpretation 2: Kullback-Leibler divergence

$D_{KL}($  , 

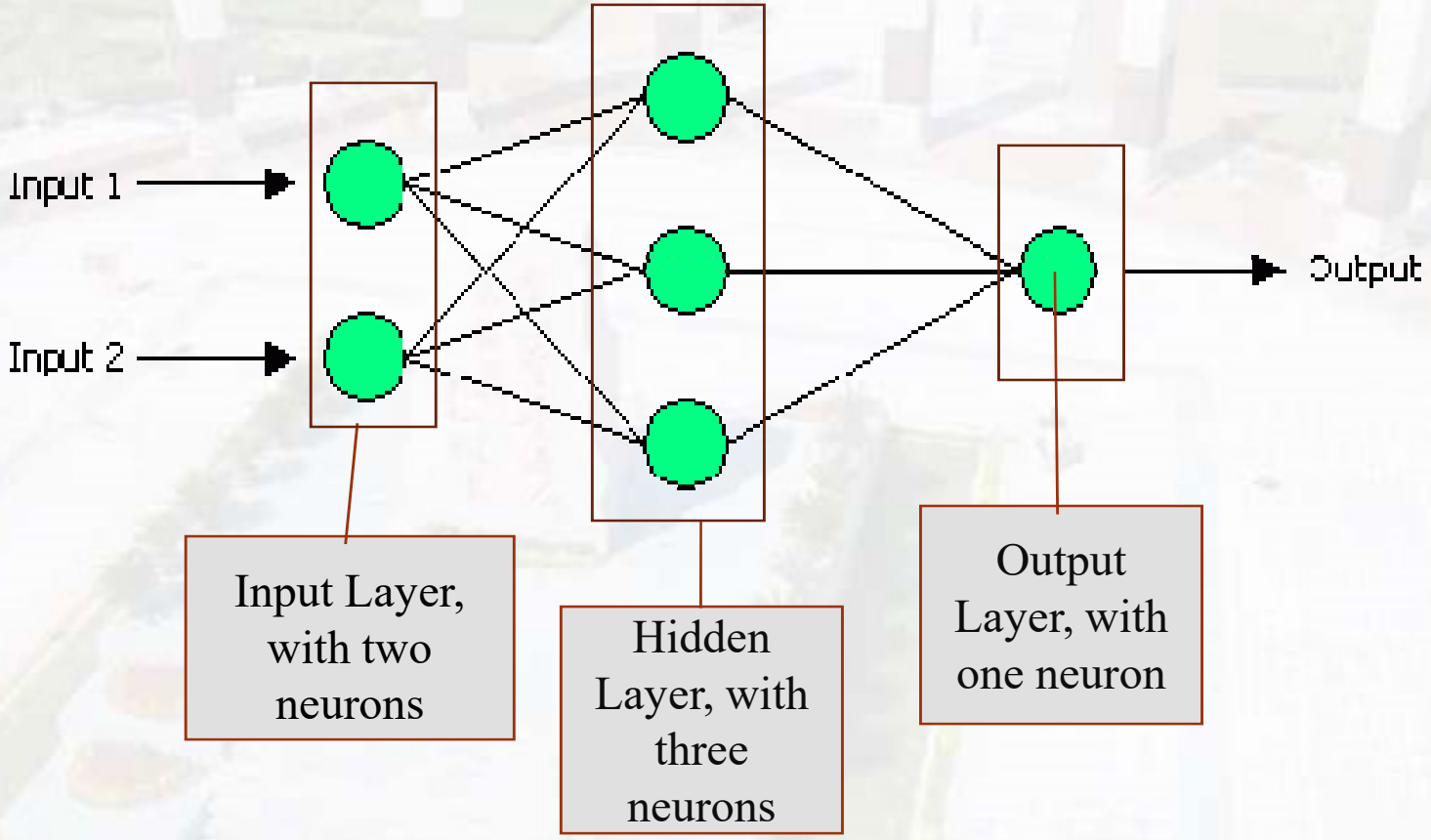
All the probability mass on the correct label ('4')

predic

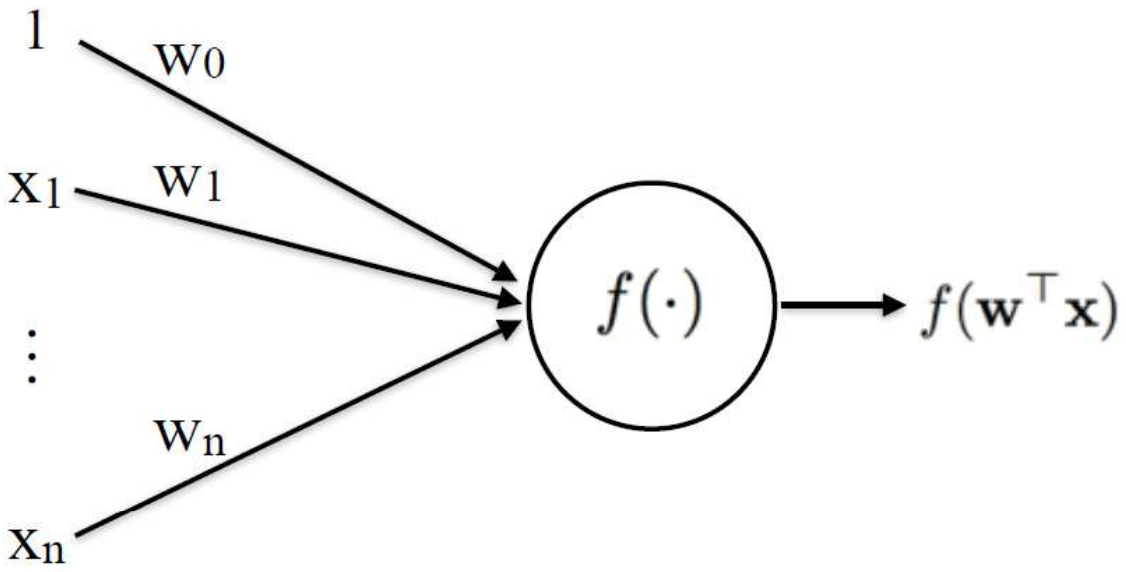
03

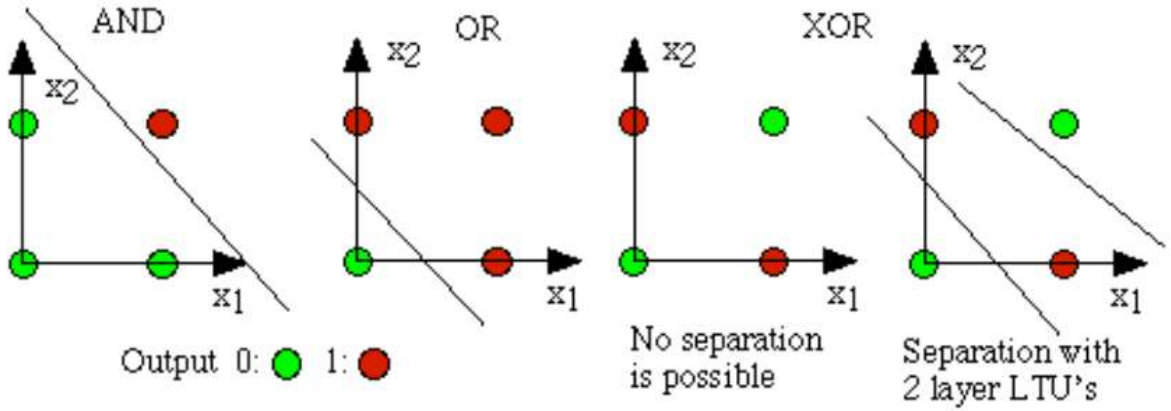
Rosenblatt Perceptron

Example: XOR
How it works?



Rosenblatt Perceptron





How it works?

- Set initial values of the weights randomly.
- Input: truth table of the XOR
- Do
  - Read input (e.g. 0, and 0)
  - Compute an output (e.g. 0.60543)
  - Compare it to the expected output. (Diff= 0.60543)
  - Modify the weights *accordingly.*
- Loop until a condition is met
  - Condition: certain number of iterations
  - Condition: error threshold

Rosenblatt Perceptron

(1)Input: $\{(X_i, y_i)\}$ i=1~n

(2)Random: $(w,\ b)$

(3)Pick up: X_i

(a)If: $wx + b \geq 0$ 且 $y = -1$, 则 $W = W + \eta yX,\ b = b + \eta y$

(4)repeat step (3), unstill all the classification are right:

Example One

If the **input/outputs** are as follows:

$$x_1 = \begin{pmatrix} 4 \\ 4 \end{pmatrix},\ y_1 = 1; x_2 = \begin{pmatrix} 4 \\ 5 \end{pmatrix},\ y_2 = 1;\ x_3 = \begin{pmatrix} 1 \\ 1 \end{pmatrix},\ y_3 = -1;$$

$$f(x) = hardlims(wx + b), \qquad x = (x^{(1)}, x^{(2)})^T;$$

Please calculate the Rosenblatt Perceptron

If learning rate η =1。

Answer: (1) if initial weight and bias are

$$\boldsymbol{W}(0) = [0 \quad 0], \; b(0) = 1$$

(2)then calculate the value of x_1、x_2、x_3 respectively,

if

$$y_1(\boldsymbol{W}_0\boldsymbol{x}_1 + b_0) = 1\left([0 \quad 0]\begin{bmatrix}4\\4\end{bmatrix}+1\right) = 1 > 0$$

Indicate x_1 is right classified。

if

$$y_2(\boldsymbol{W}_0\boldsymbol{x}_2 + b_0) = 1\left([0 \quad 0]\begin{bmatrix}4\\5\end{bmatrix}+1\right) = 1 > 0$$

Indicate x_2 is right classified

$$y_3(\boldsymbol{W}_0\boldsymbol{x}_3+b_0)=-1\left([0\quad 0]\begin{bmatrix}1\\1\end{bmatrix}+1\right)=-1<0$$

Indicate x_3 is wrong classified, therefore, we need to update the weight and bias now,

$$\boldsymbol{W}(1)=\boldsymbol{W}(0)+1\times(-1)\begin{bmatrix}1\\1\end{bmatrix}^{\mathrm{T}}=[-1\quad -1]$$

$$b(1)=b(0)+1\times(-1)=0$$

After the updated weight, we calculate the value of x_3 again,

$$y_3(\boldsymbol{W}_1\boldsymbol{x}_3+b_1)=-1\left([-1\quad -1]\begin{bmatrix}1\\1\end{bmatrix}\right)=2>0$$

Indicate x_3 is right classified。

We need to test x_1 again.

$$y_1(\boldsymbol{W}_1\boldsymbol{x}_1+b_1)=1\left([-1\quad -1]\begin{bmatrix}4\\4\end{bmatrix}+1\right)=-7<0$$

Indicate x_1 is classified wrong, therefore, we need to update the weight and bias now,

$$\boldsymbol{W}(2)=\boldsymbol{W}(1)+1\times 1\begin{bmatrix}4\\4\end{bmatrix}^{\mathrm{T}}=[3\quad 3]$$

$$b(2)=b(1)+1\times 1=1$$

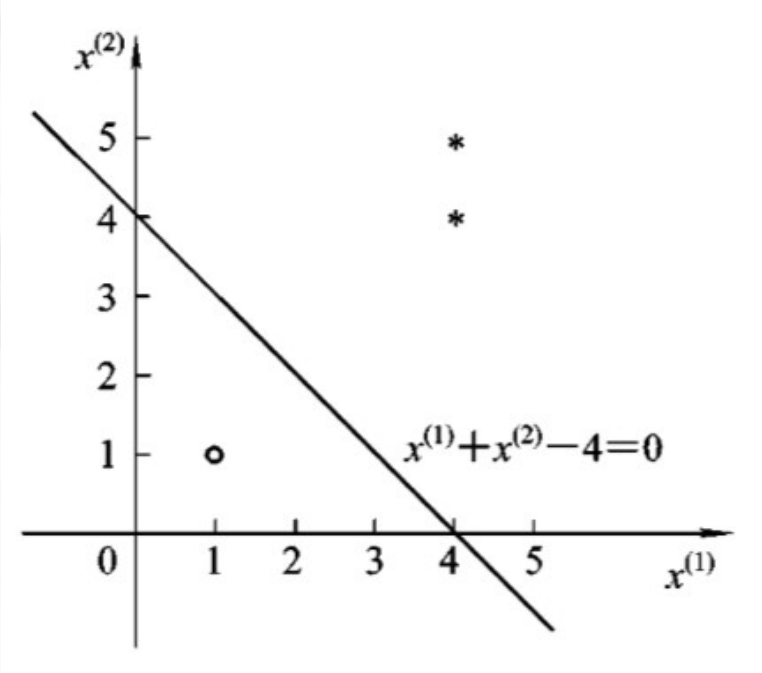
Right now, we can see that $y_1(\boldsymbol{W}_2\boldsymbol{x}_1+b_2)>0$、$y_2(\boldsymbol{W}_2\boldsymbol{x}_2+b_2)>0$、$y_3(\boldsymbol{W}_2\boldsymbol{x}_3+b_2)<0$ indicate not all of them is classified right.

After multiply iteration, until $\boldsymbol{W}(9)=[1\quad 1]$, $b(9)=-4$ a all the points are classified right.

Finally, the modal is

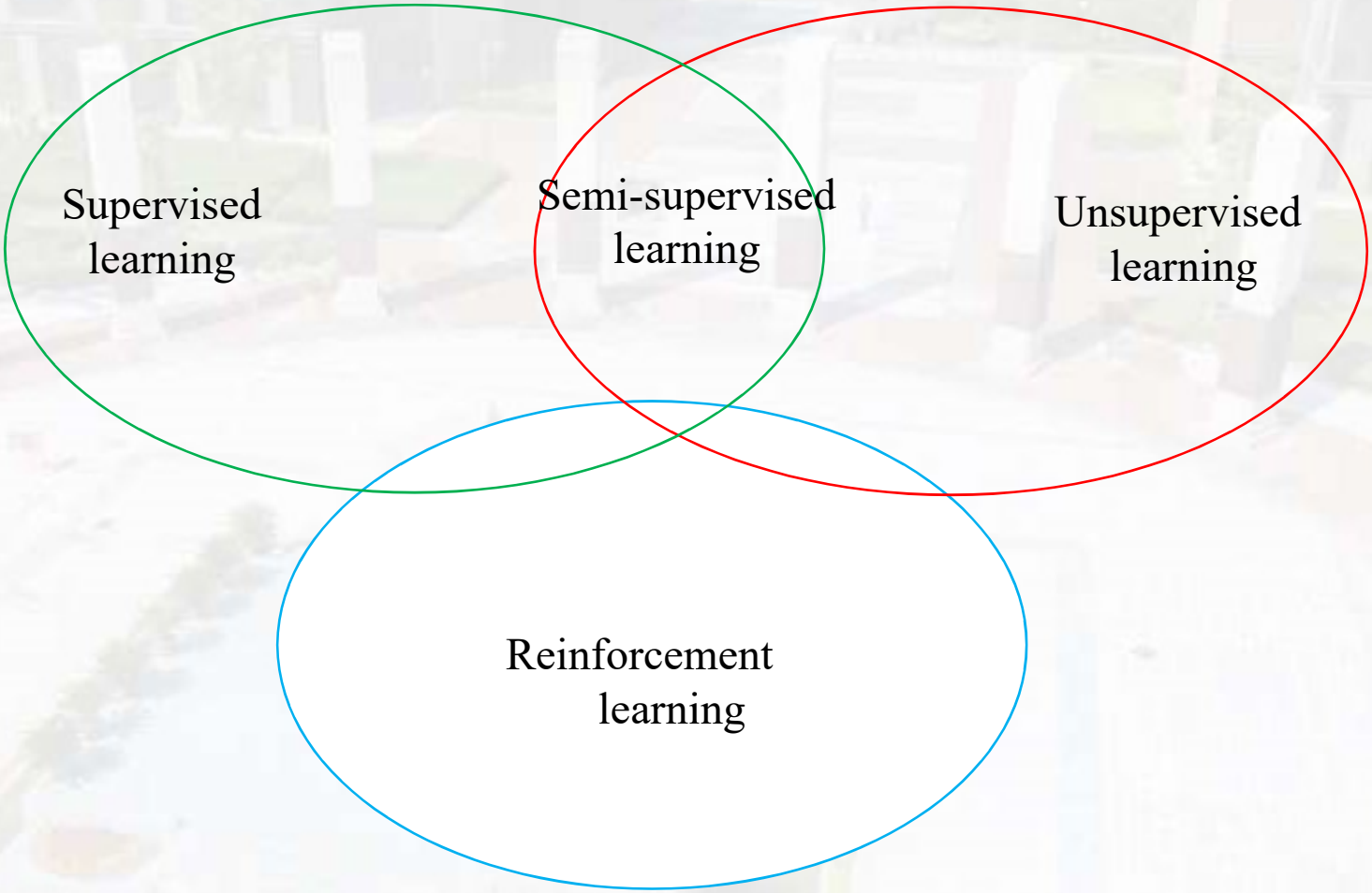
$$f(x) = \text{hardlims}(x^{(1)} + x^{(2)} - 4)$$

The classification result is as follows:

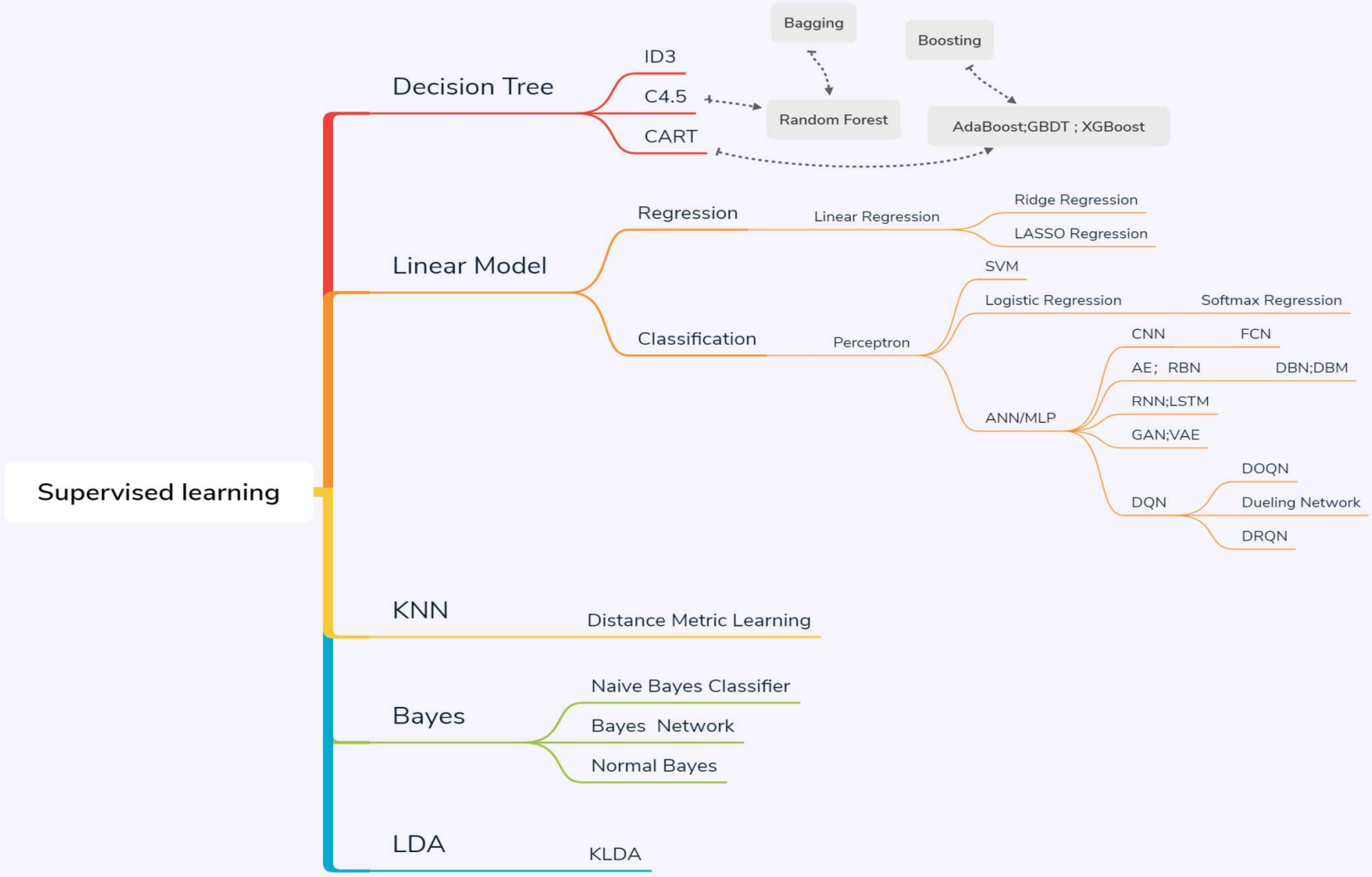



04
Learning

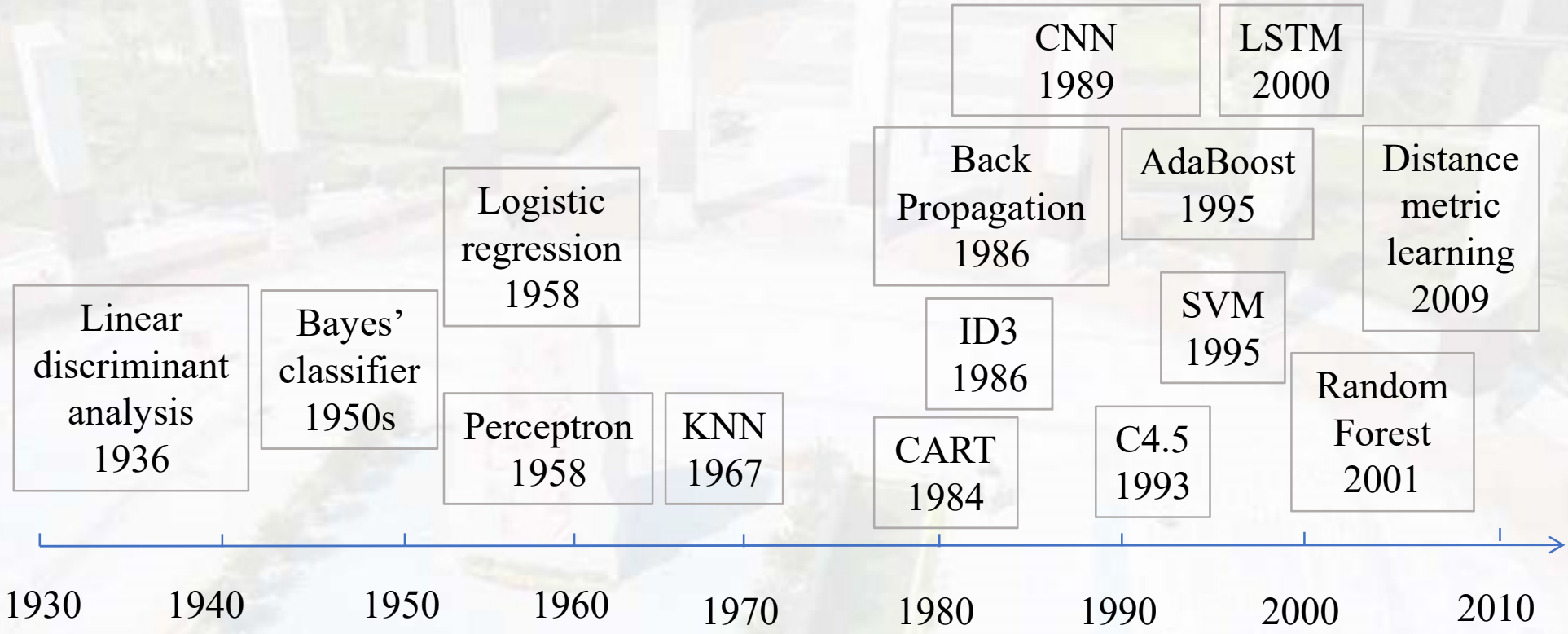
Learning


Supervised learning
Semi-supervised learning
Unsupervised learning
Reinforcement learning

Supervised Learning



Supervised Learning

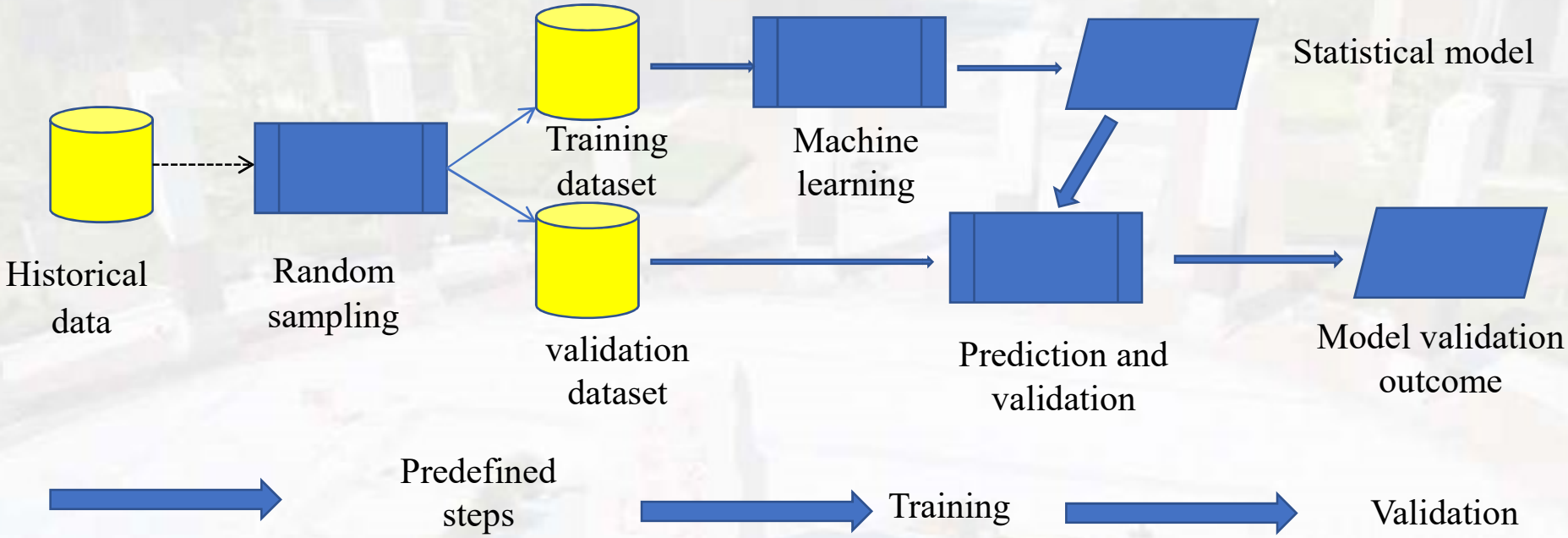


History development of the supervised learning

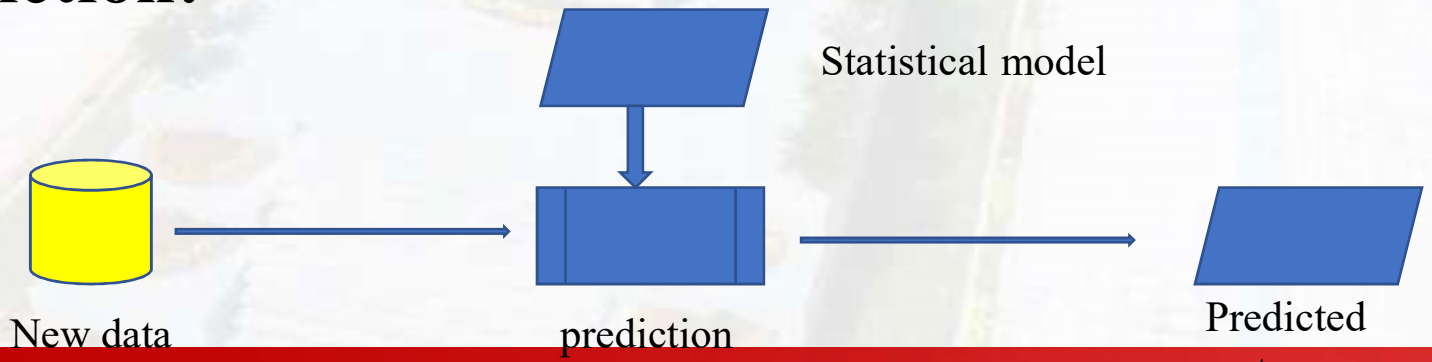
Definition:

- The supervised learning is one of the machine learning (ML) methods, and explores the study and construction of algorithms that can learn from data and make predictions.
- Such algorithms operate by building a model from example inputs in order to make data driven predictions or decisions, rather than following strictly static program instructions.
- In supervised learning , each example is a pair consisting of an input object and a desired output value. A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples.

Training and validation:



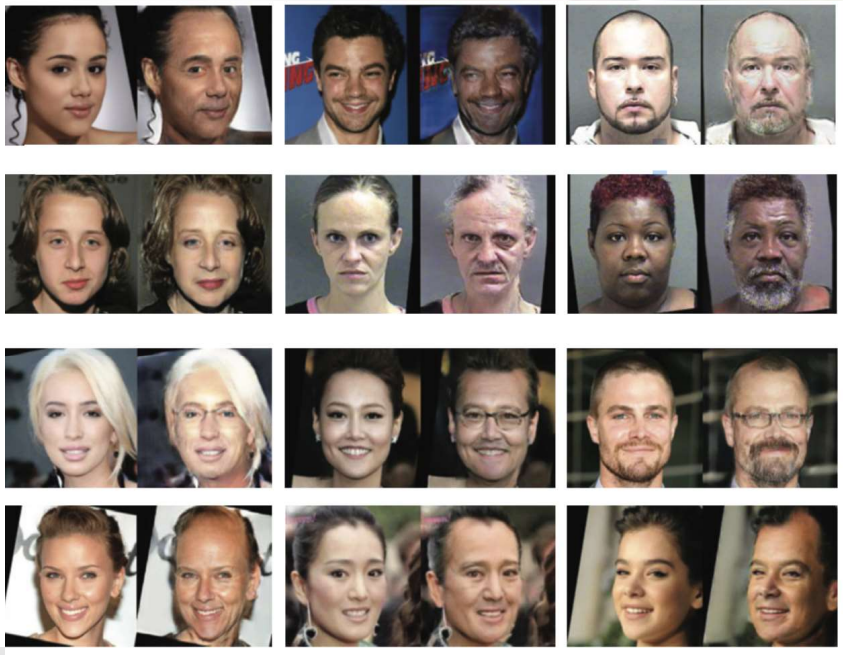
Prediction:



Supervised Learning

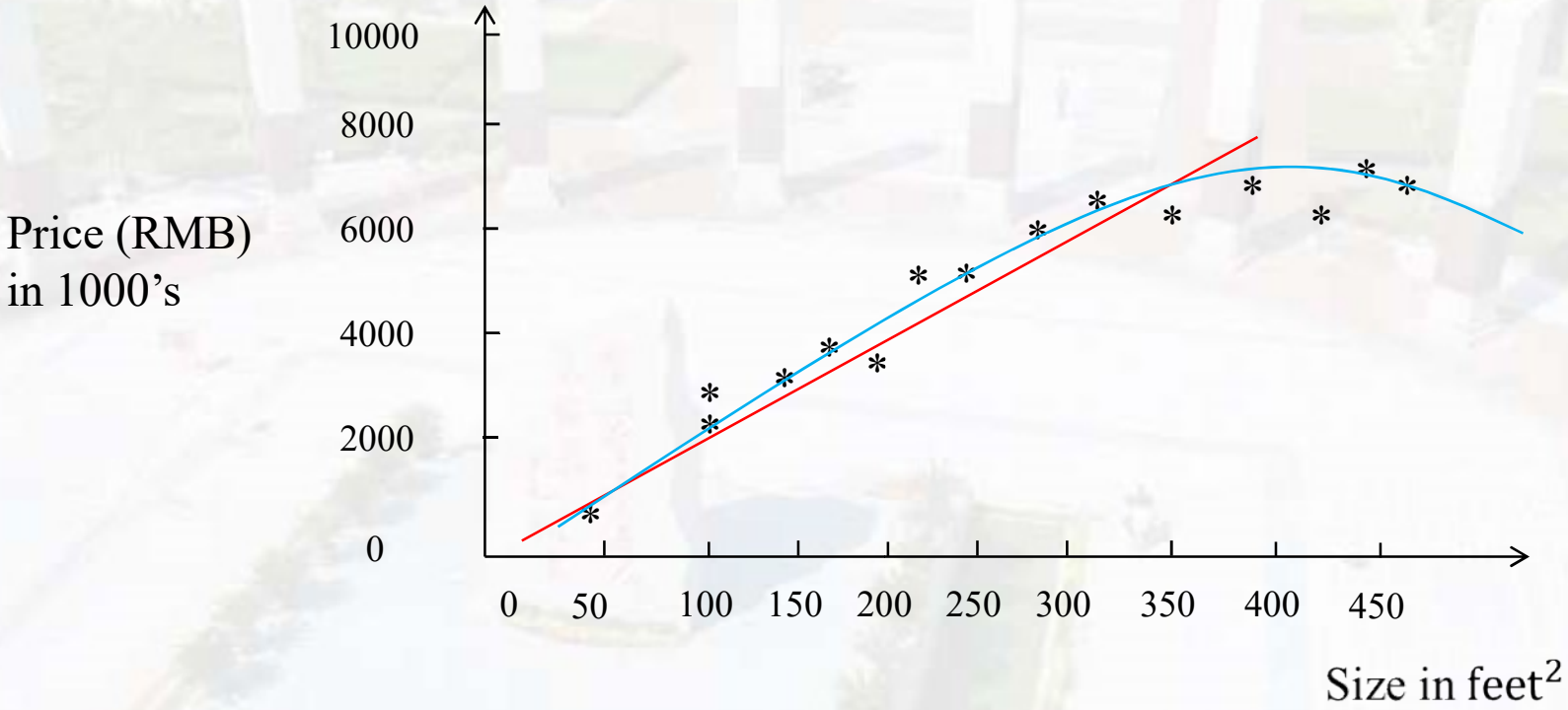
Regression and Classification:

- Regression means to predict the output value using training data.
  e.g. and use regression to predict the stock price from training data.
- Classification means to group the output into a class.
  e.g. we use classification to predict age of the test data sample.



Example I: Classification

Regression in housing price prediction:



And there are two methods of the regression to process the housing price, the linear function and the second linear function.

Supervised Learning

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \ldots, (x^{(m)}, y^{(m)})\}$

Hypothesis function: $f: x \rightarrow y$

Training set: $(x^{(i)}, y^{(i)})$

Training process: $\min imize \sum_{i=1}^{m} (y_h^{(i)} - y_r^{(i)})^2$

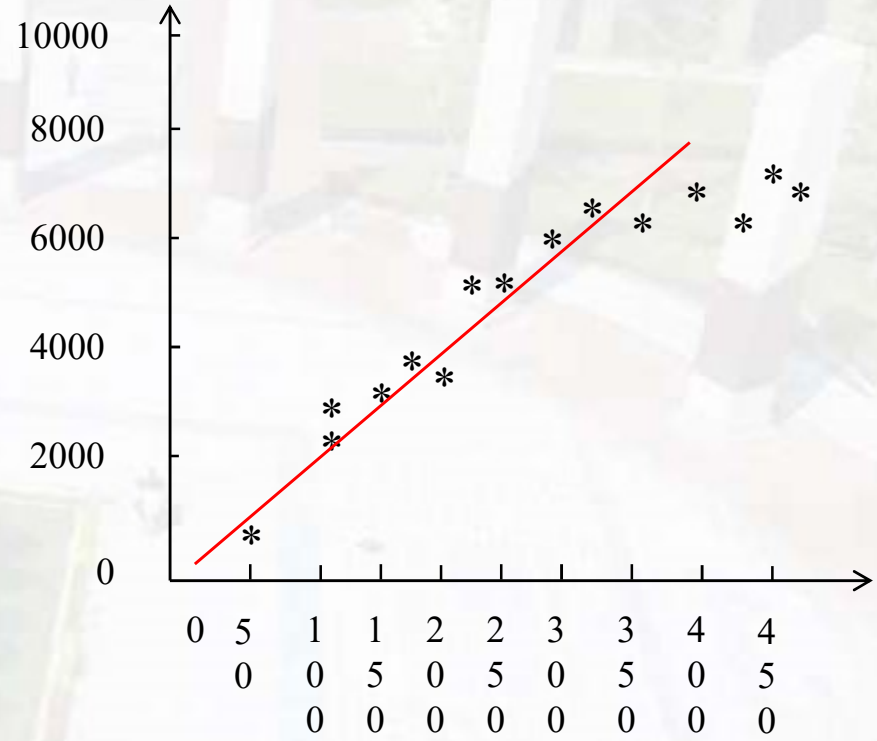
New data: $(x^{(j)}, y^{(j)})$

$y_{prediction}^{(j)}$

Example II: Regression

Training set :

Size	Price(RMB10000)
50	70
78	130
90	175
120	256
200	382
…	…



Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Choose appropriate parameters.

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Parameters: $\theta_0 \quad \theta_1$

Cost function: $J(\theta_0, \theta_1) = \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right)^2$

Goal: $minimize J(\theta_0, \theta_1)$ ⟸ **How?**

$$\frac{\partial J(\theta_0, \theta_1)}{\partial \theta_0} = 0 \qquad \frac{\partial J(\theta_0, \theta_1)}{\partial \theta_1} = 0$$

Repeat until convergence{

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \qquad \text{for j = 0 to j = 1}$$

}

⟹ $\theta_0 \quad \theta_1$ ⟹ $h_\theta(x) = \theta_0 + \theta_1 x$

Supervised Learning

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

$$= \theta^T X + \theta_0$$

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \cdots \\ \theta_n \end{bmatrix} \quad X = [x_1, x_2, x_3, \ldots, x_n]$$

Supervised Learning

Classification

Discriminative algorithms

Map x → y directly

e.g. using discriminative algorithms to classify the age of the person by face recognition

Logistic regression, SVM, Neural net works

Generative algorithms

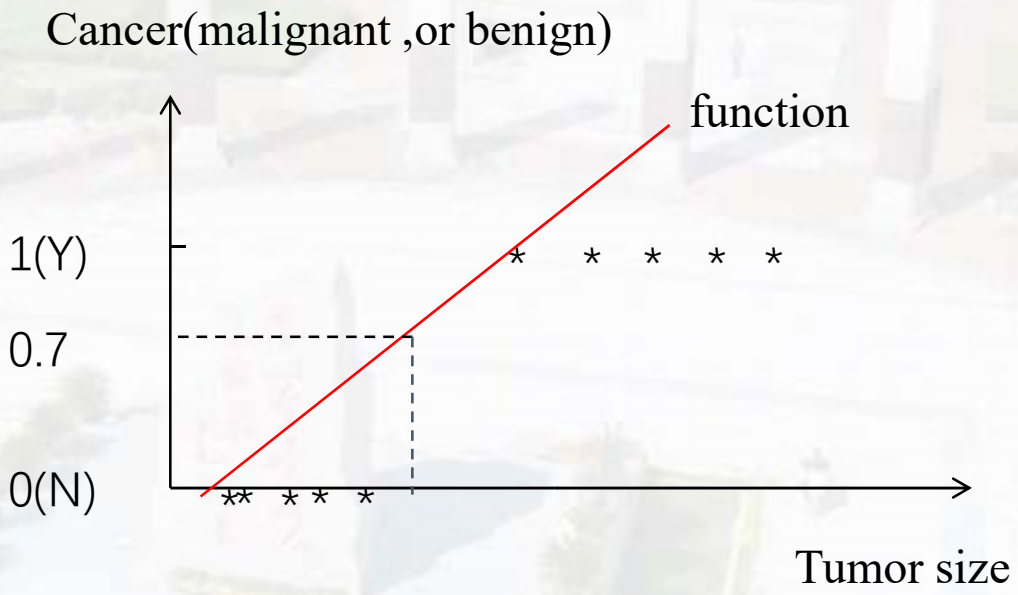
Models a more general problem: how the data was generated

e.g. you have a mail (the observation) and you want to infer whether the mail is spam or not (the cause).

Naïve bayes, bayesian network classifier,

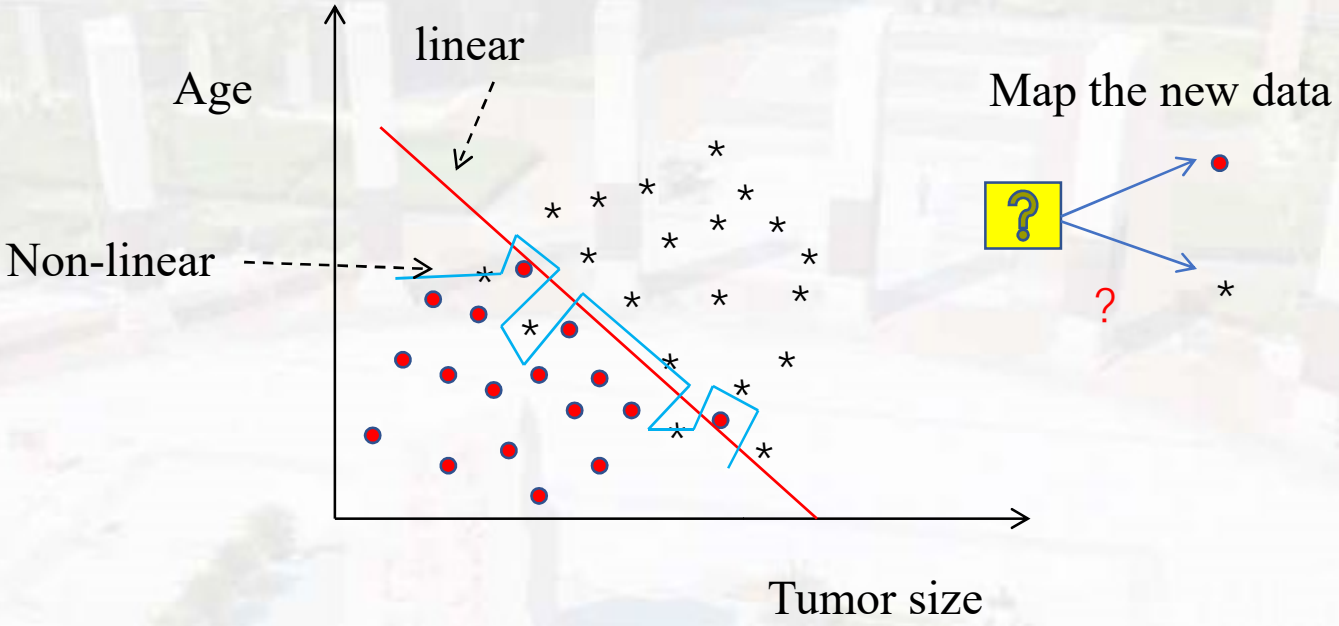
Example III:

Classification in the tumor



Though this is a binary classification problem, this has a method of the linear regression to process it.

Classification in the tumor

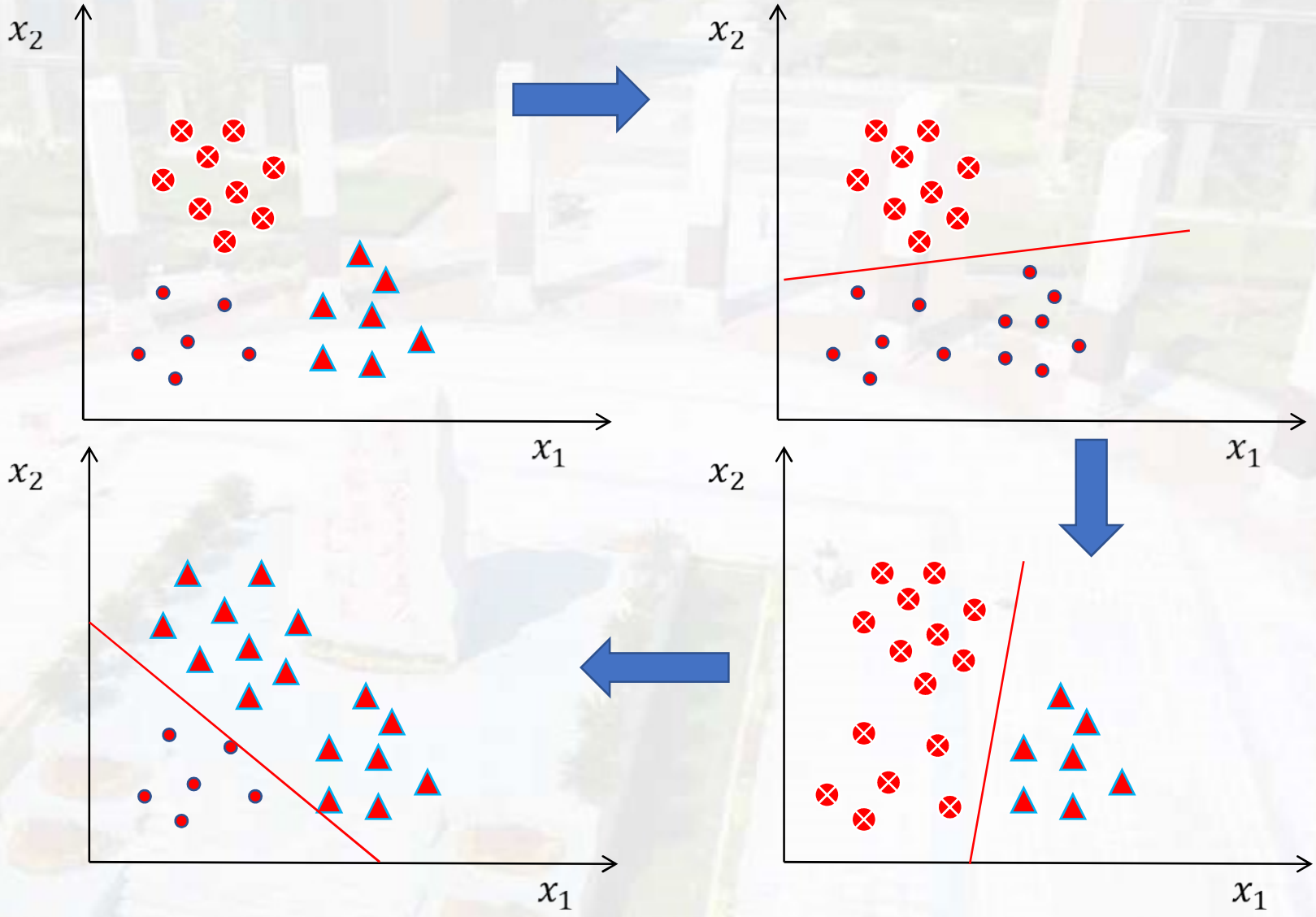


“ ● ” means the begin, “ *” means the malignant

Training set: $\left\{\left(x^{(1)},y^{(1)}\right),\left(x^{(2)},y^{(2)}\right),\left(x^{(3)},y^{(3)}\right),\ldots,\left(x^{(m)},y^{(m)}\right)\right\}$

This is also a binary classification problem. By the tumor size and age to confirm the one whether is the malignant cancer.

Supervised Learning



Logistic Regression

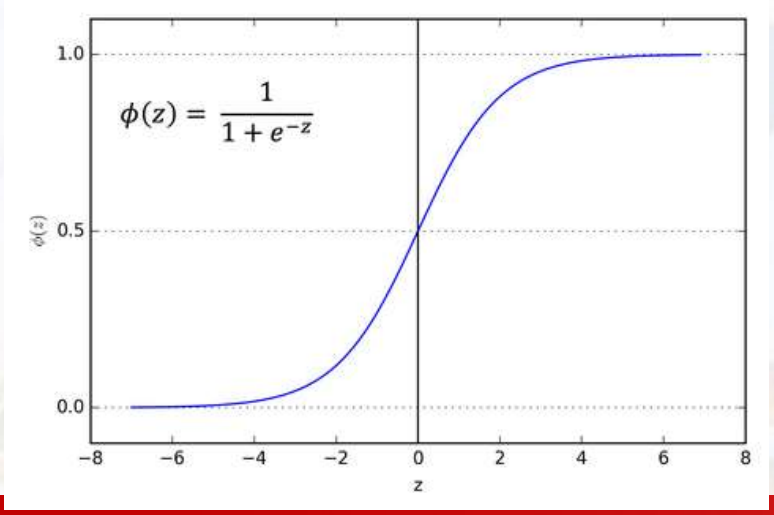
Logistic Regression (or logit regression) is a linear classification approach. It is used to model the probability of a certain class or event existing for the binary classification such as pass/fail, win/lose, alive/dead or healthy/sick.

Linear regression model:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = \theta^T x + \theta_0$$

$$z = \theta^T x + \theta_0$$

$$\begin{cases} 0 & \text{Negative class, (e.g. fail)} \\ 1 & \text{Positive class, (e.g. pass)} \end{cases}$$



Sigmoid function

$$\phi(z) = \frac{1}{1+e^{-z}} \quad \Rightarrow \quad \phi(z) = \begin{cases} 1, \\ 0.5 \\ 0 \end{cases}$$

Linear model → Classification model

$$z = \theta^T x + \theta_0 \qquad \phi(z) = \frac{1}{1+e^{-z}} \Rightarrow h_\theta(x) = \frac{1}{1+e^{-(\theta^T x+\theta_0)}} \Rightarrow y = \begin{cases} 0 \\ 1 \end{cases}$$

Logarithm

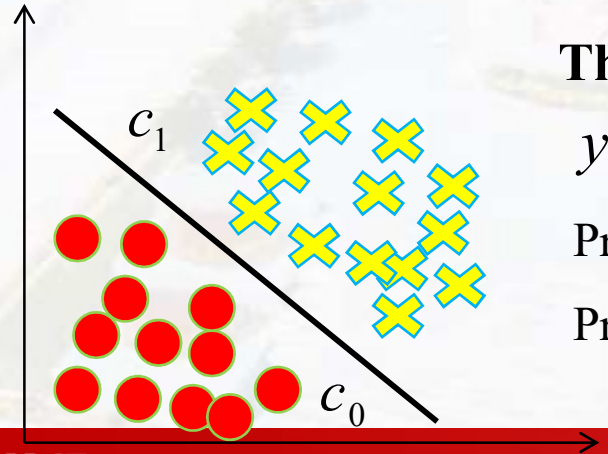
Log odds,(or Logit)

$$\ln \frac{h_\theta(x)}{1-h_\theta(x)} = \theta^T x + \theta_0$$

Logistic regression, or logit regression

Odds

The above equation use prediction result of the linear regression model to logarithmic probability of approximating the real token.



The new hypothesis function:

$$y = h_\theta(x) = \phi\left(\theta_0 + \theta_1 x_1 + \theta_2 x_2\right)$$

Predict " $y = h_\theta(x) = 1 \in c_1$ ", if the $\theta_0 + \theta_1 x_1 + \theta_2 x_2 > 0$

Predict " $y = h_\theta(x) = 0 \in c_0$ ", if the $\theta_0 + \theta_1 x_1 + \theta_2 x_2 < 0$

$$h_\theta(x) = \frac{1}{1+e^{-(\theta^T x+\theta_0)}}$$ → How to calculate the θ ?

Logistic regression cost function:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{cost}\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right)$$

$$\text{cost}\left(h_\theta(x), y\right) = \begin{cases} -\log\left(h_\theta(x)\right) & \text{if } y=1 \\ -\log\left(1-h_\theta(x)\right) & \text{if } y=0 \end{cases}$$

Cost $=0$ if $y=1, h_\theta(x)=1$
But as Cost $\to \infty$
$h_\theta(x) \to 0$

→ $p(y \mid x;\theta) = \left(h_\theta(x)\right)^y \left(1-h_\theta(x)\right)^{1-y}$

According to the logarithmic loss function:

→ $$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)\right]$$

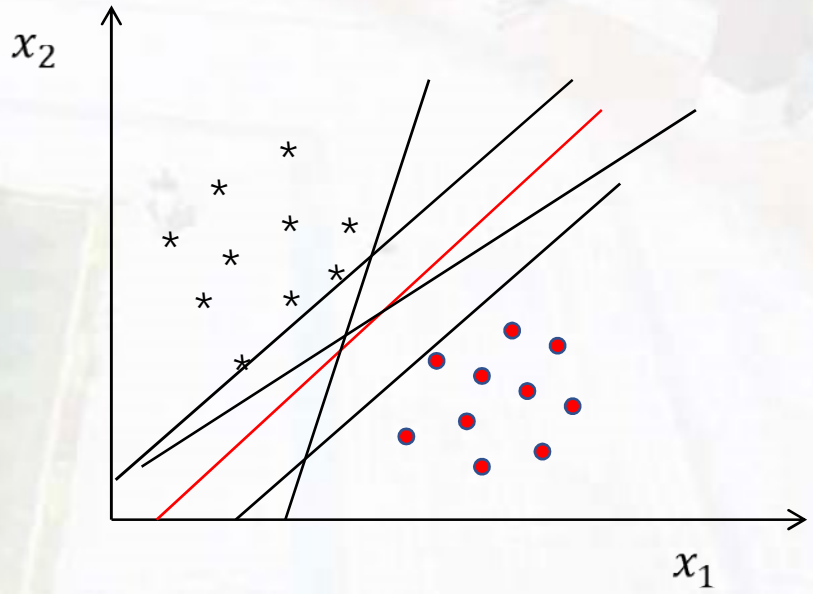
To fit the parameter θ : $\min_\theta J(\theta)$ ← Gradient Descent

Support Vector Machine(SVM)

Training set:

$$D=\left\{\left(x^{(1)},y^{(1)}\right),\left(x^{(2)},y^{(2)}\right),\left(x^{(3)},y^{(3)}\right),\ldots,\left(x^{(m)},y^{(m)}\right)\right\},\mathrm{y}_i\in\{-1,+1\}$$

The basic idea of class learning is to find a maximum margin partition hyperplane in the sample space based on training set D ,the maximum margin is calculated by the support vector , and to separate the different samples.



The partition hyperplane:

$$w^T x^{(i)} + b = 0$$

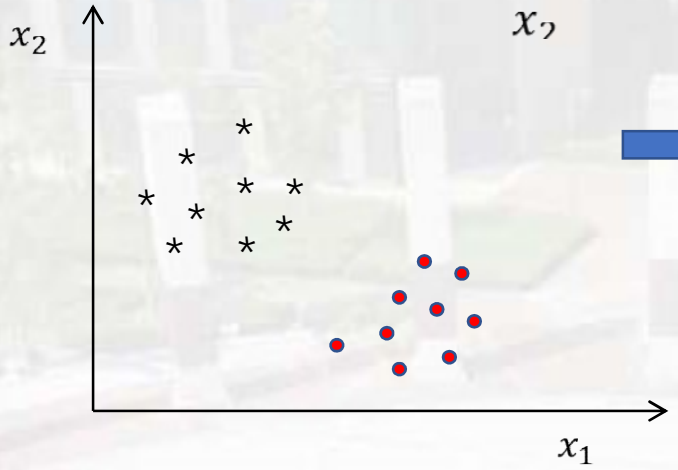
Among the equation:

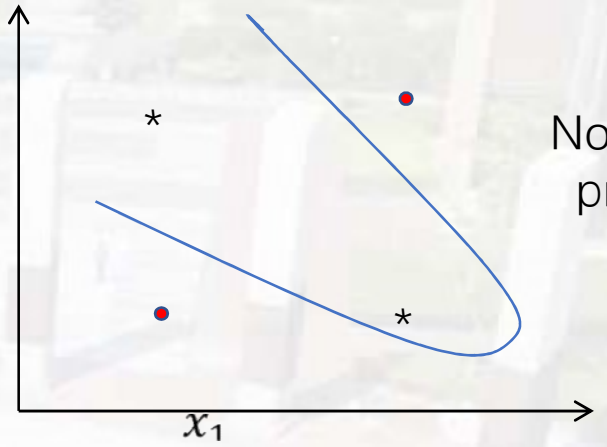
$w = (w_1; w_2; \ldots; w_d)$ is the normal vector, which determine the orientation of the partition hyperplane.

b is the displacement item, which determine the distance of between the partition hyperplane and original point.

Hypothesis, the partition hyperplane (w, b) can separate correctly the training set, i.e. for the $(x^{(i)}, y^{(i)}) \in D$, if the $y^{(i)} = +1$, $w^T x^{(i)} + b > 0$; or if the $y^{(i)} = -1$, $w^T x^{(i)} + b < 0$

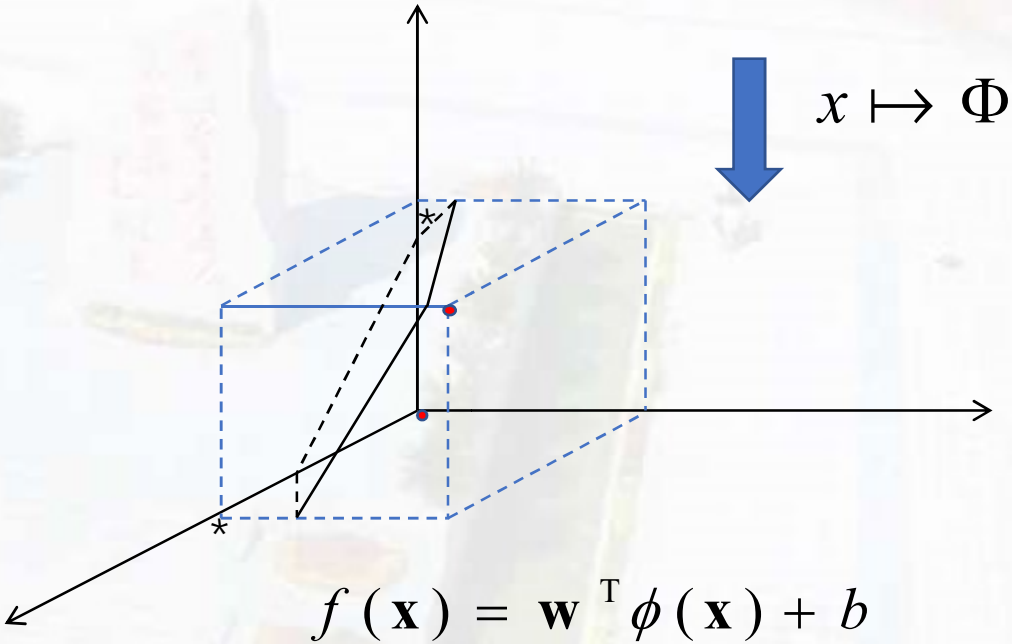
$$\begin{cases} w^T x^{(i)} + b >= +1, y^{(i)} = +1 \\ w^T x^{(i)} + b <= -1, y^{(i)} = -1 \end{cases}$$





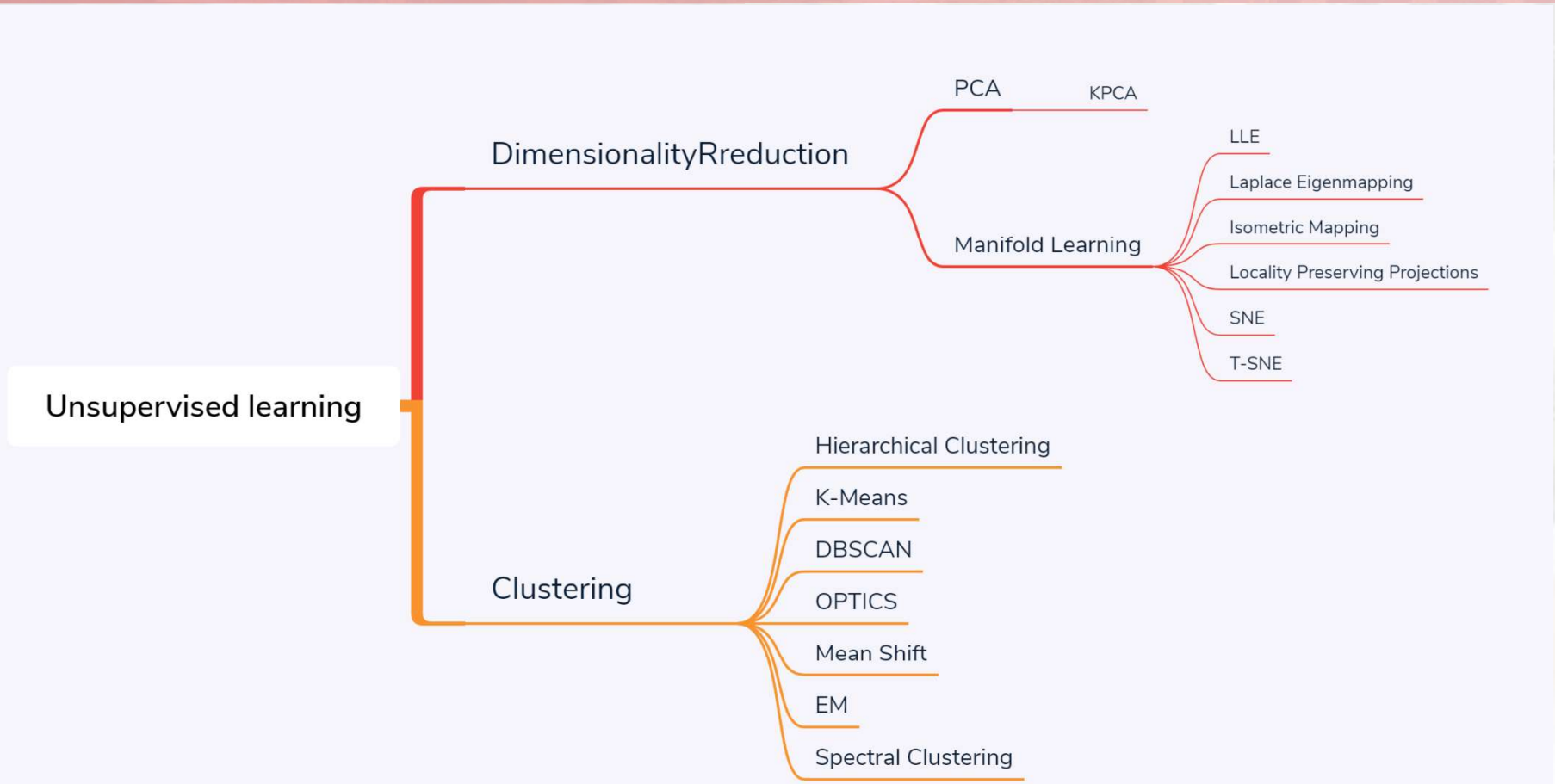
Non-linear problem

$x \mapsto \Phi(x)$

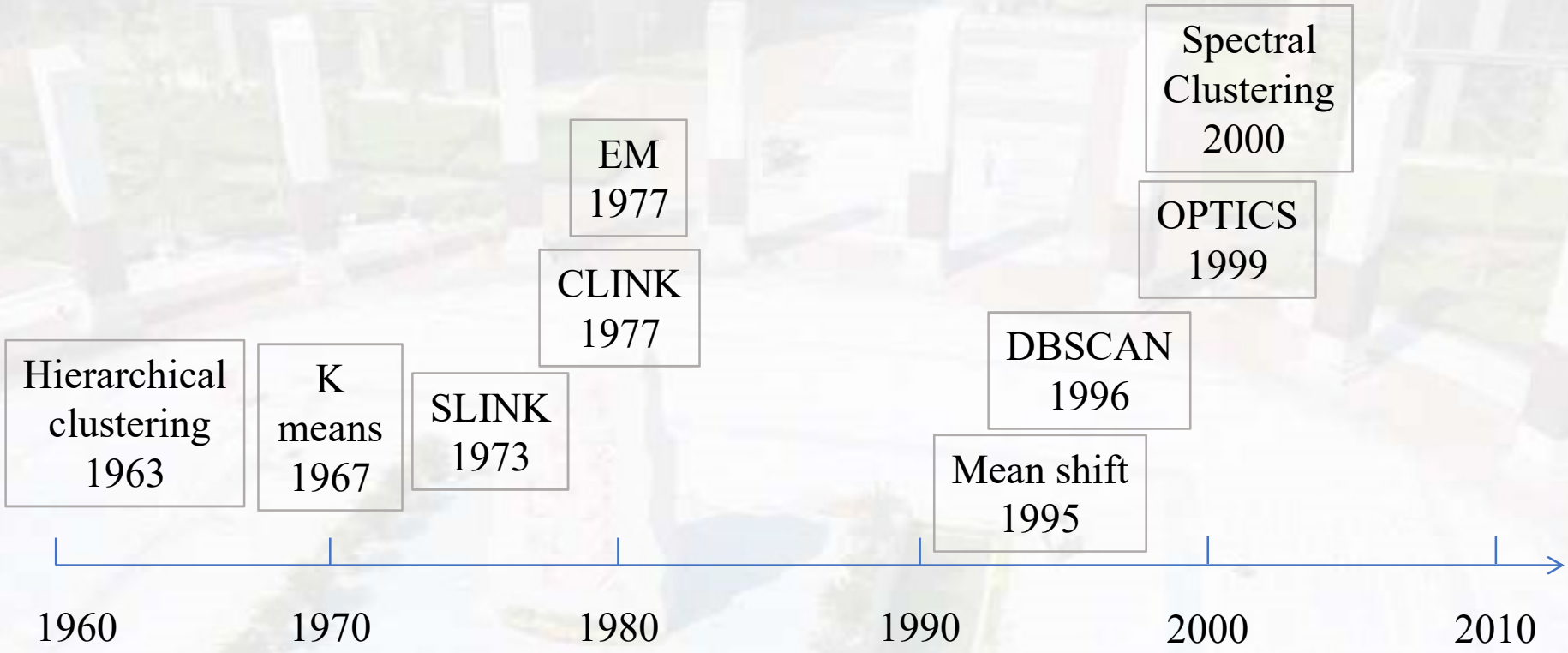


$$f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + b$$

Unsupervised Learning

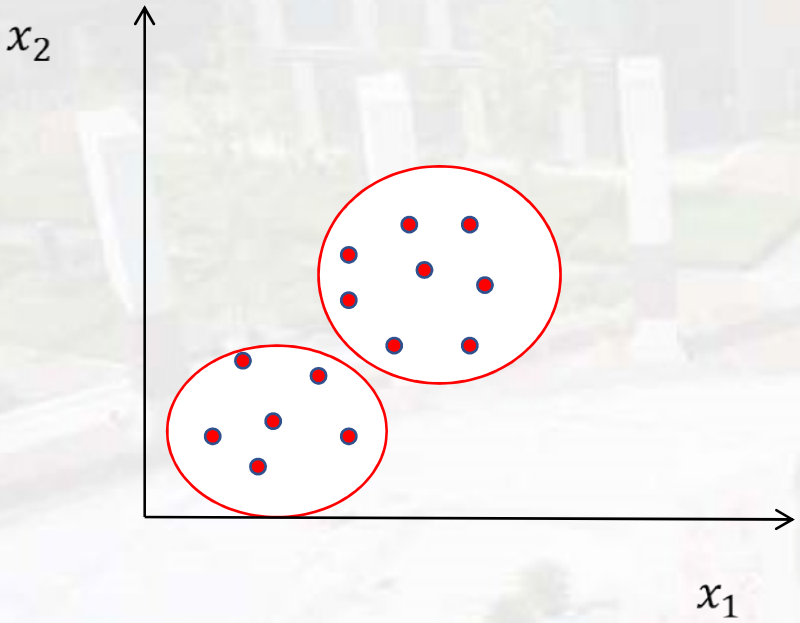


If instances are given with known labels (the corresponding correct outputs) then the learning is calle
supervised, in contrast to unsupervised learning, where instances are unlabeled.

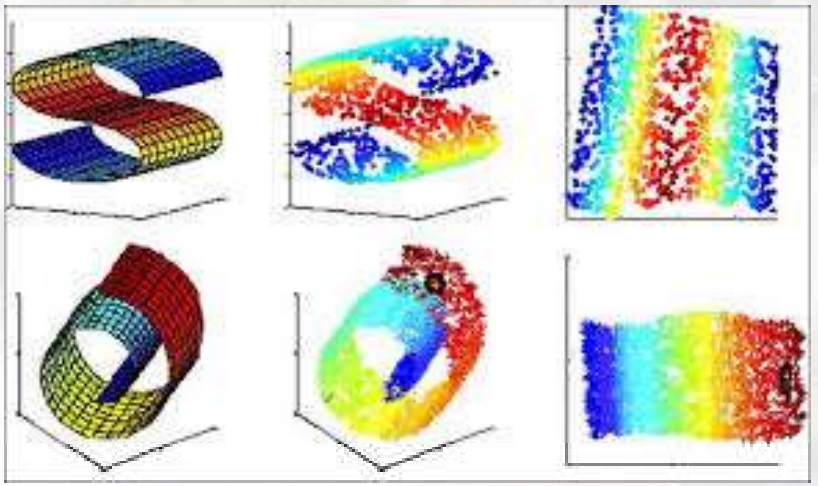


History development of the clustering of unsupervised learning

Unsupervised Learning



Unsupervised learning--clustering

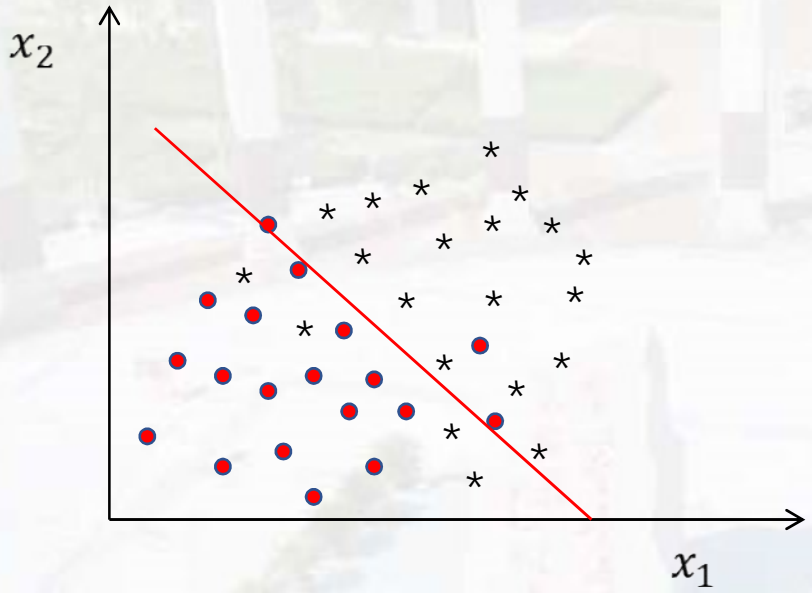


Unsupervised learning--dimensionality reduction

Training set: $\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(m)}\}$

Applications: web search, cognitive discipline, neurology, genetics.

Comparing the supervised learning and unsupervised learning.



Supervised learning--classifying

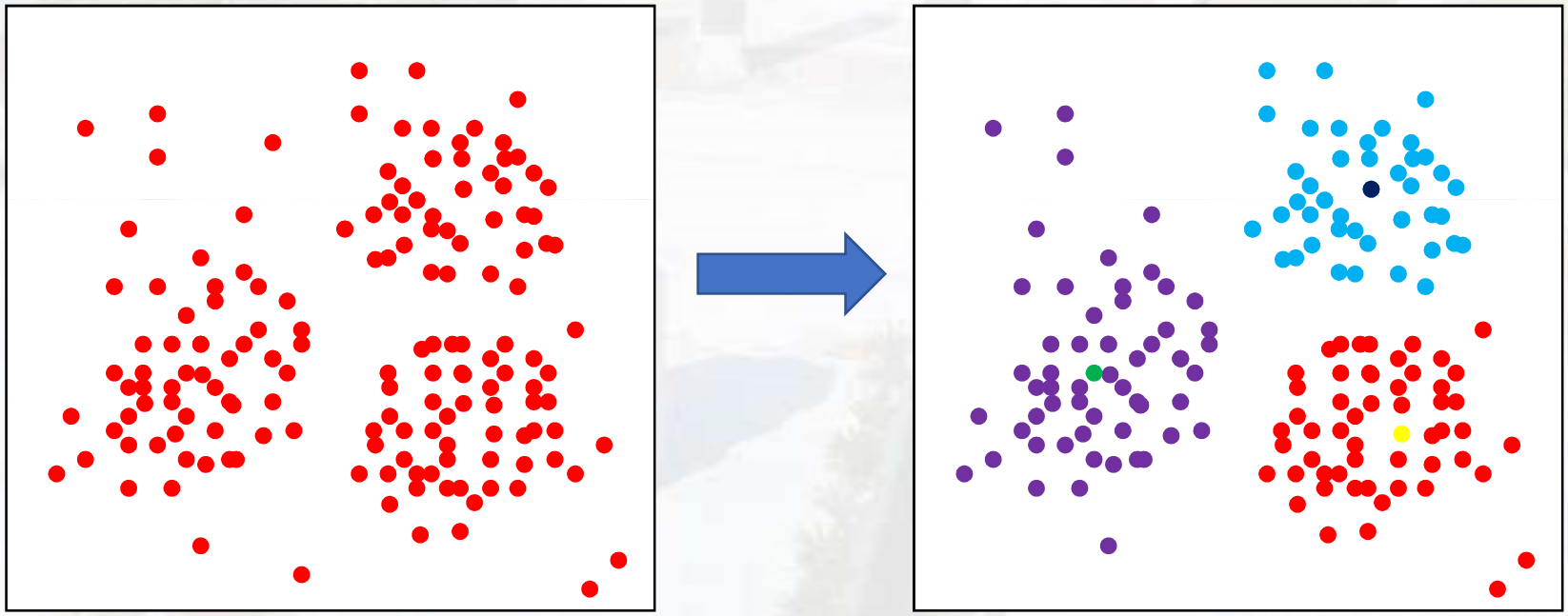
The data have the different labels, or different features. We know the correct answer.

Unsupervised learning--clustering

The data have not the label, or have the same labels. We don't know the what is the correct answer.

K-means clustering algorithm

means clustering divides ***n*** points (an observation or a sample) into ***k*** clusters such that each point belong
e nearest mean (this is the cluster center) corresponding clusters are used as criteria for clustering.



milarity measures (such as commonly used measures of distance: Euclidean distance, equine distanc
mming distance, cosine distance, etc.) are carried out on one dimension of the sample, and the similarit
gree is used to estimate the category of the sample.

For each cluster, we can select a center, so that the distance from all points in the cluster to the center is smaller than the distance to the center of other cluster.

Given sample set: $D = \{x_1, x_2, \ldots, x_m\}$
divided by the clustering: $C = \{C_1, C_2, \ldots, C_k\}$
k-means clustering algorithm make minimum square error:

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|_2^2 \qquad \mu_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} x$$

μ_i is the cluster C_i means vector. The above equation describe the closeness degree of the cluster inner sample around cluster means vector. Value E is smaller, similarity degree of the cluster inner sample is higher.

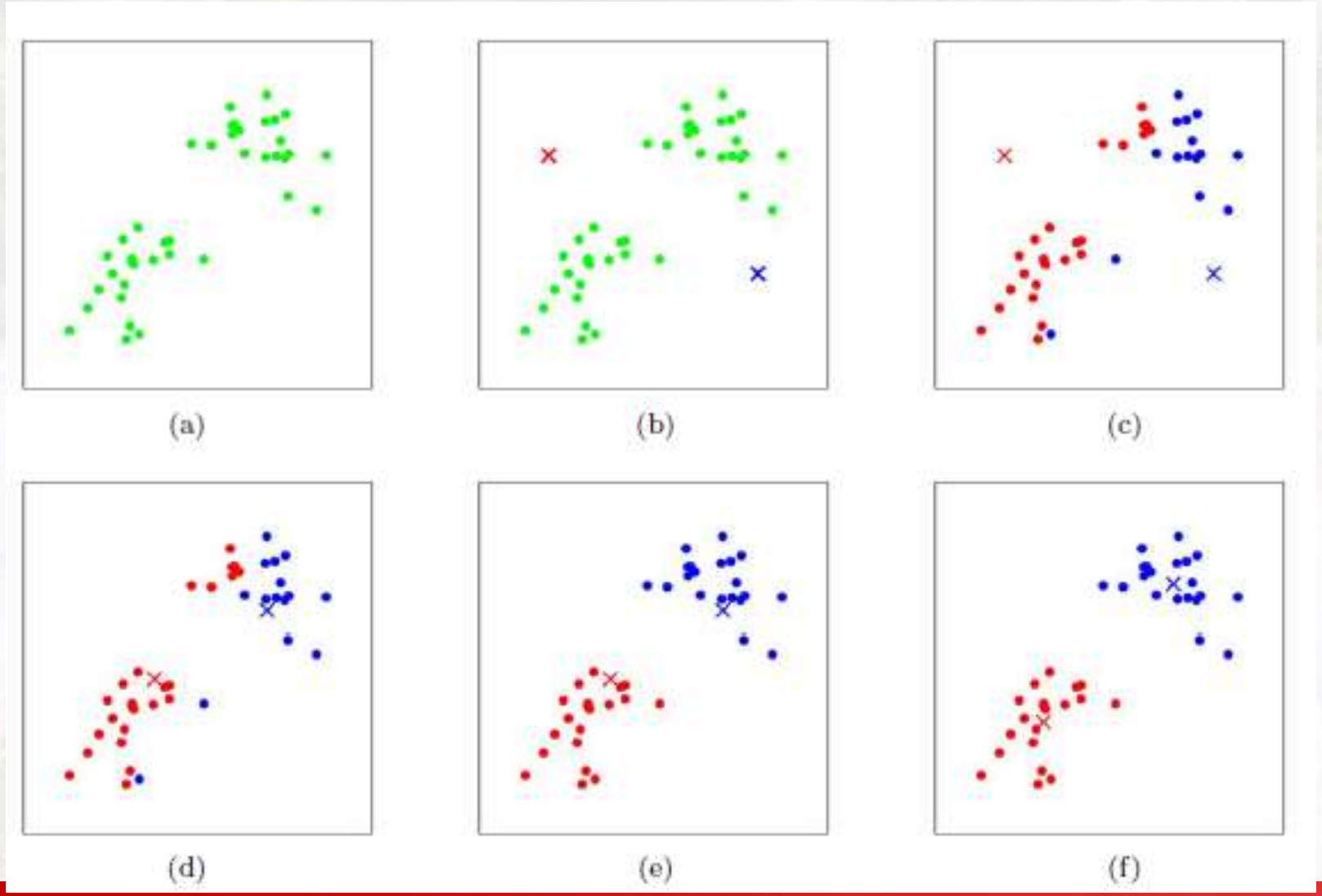
Steps for the *k*-means clustering:

1. selecting initial value of the k center μ_i
2. every point is classified nearest neighbor center point corresponding cluster.
3. computing new center point of the every cluster by function

$$\mu_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} x$$

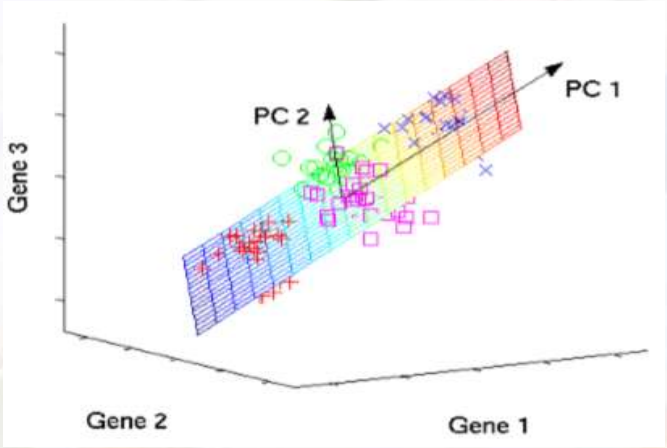
4. repeating the step 2, until iteration maximum steps or the difference value between former and later is less than a threshold.

Example I: K-means clustering algorithm



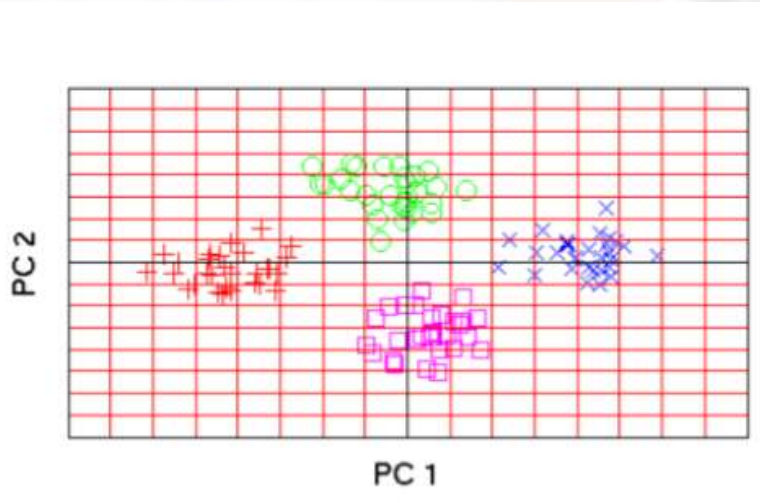
Principal Component Analysis (PCA)

Principal Component Analysis(PCA) is the often used to dimensionality reduction, while preserving as much of the randomness in the high-dimensional space as possible. PCA is mainly concerned with identifying correlations in the data.



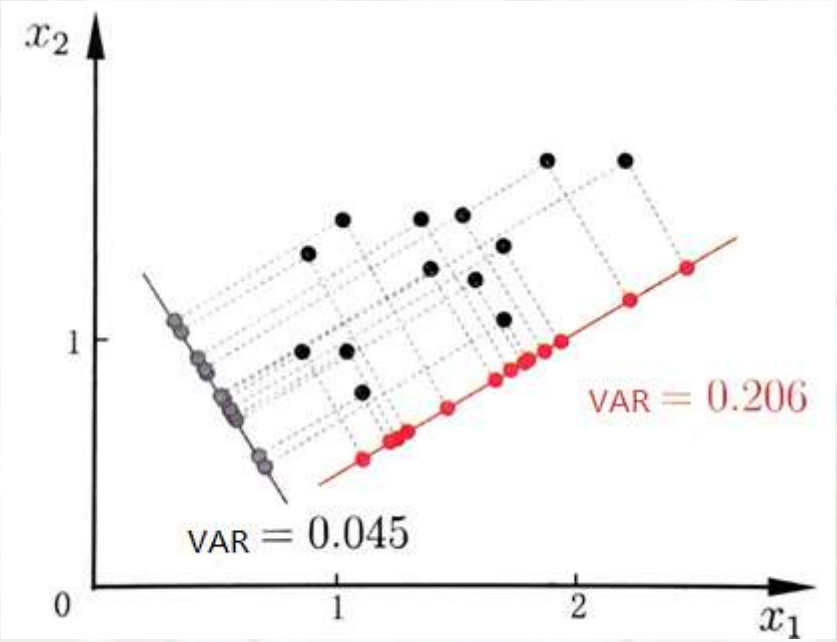
PCA

Component space

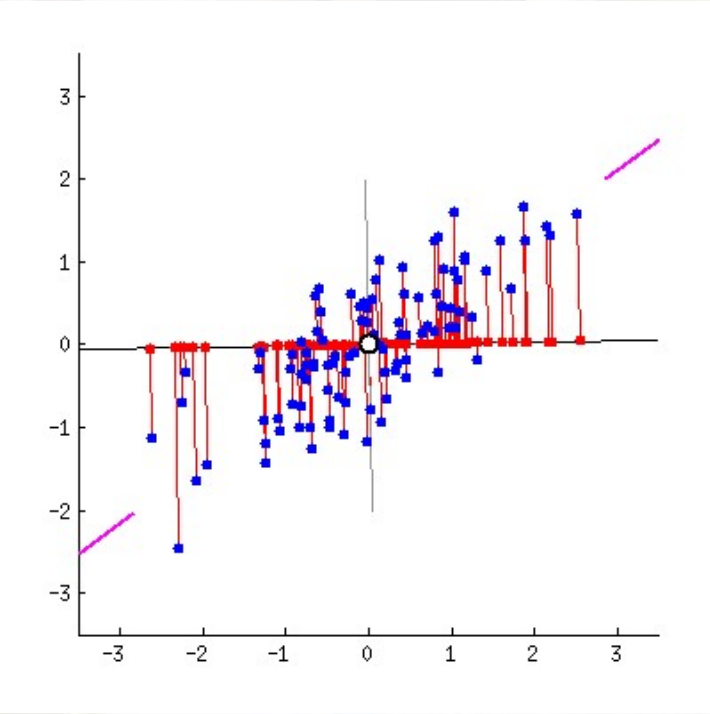


For sample points in the orthogonal attribute space, There is a hyperplane that contains the following two properties.

- **Recent reconfigurability:** the distance from the sample point to this hyperplane is close enough
- **Maximum separability:** the projection of the sample point on this hyperplane can be separated as much as possible.



Maximum separability



Recent reconfigurability

Steps for PCA

Input sample set $D = \{x_1, x_2, \ldots, x_m\}$;
Low dimensional space dimensions d'.
Process:

1. centralization for the all sample: $\mathbf{x}_i \leftarrow \mathbf{x}_i - \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i$

2. computing covariance matrix of the sample $\mathbf{X}\mathbf{X}^{\mathrm{T}}$.
3. making eigenvalue decomposition for the covariance matrix.
4. when maximum d' eigenvalue, computing corresponding eigenvector $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{d'}$.

output: projection matrix $\mathbf{W}^* = \left(\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{d'}\right)$

d' is the dimension of low-dimensional space of dimensionality reduction, Better d' values are selected by cross-validation of k-nearest neighbor classifier in low-dimensional space with different d' values.


THANK
YOU
谢谢！