

# 机器学习初步

——感知机与K近邻

李爽

助理教授，特聘副研究员

数据科学与知识工程研究所

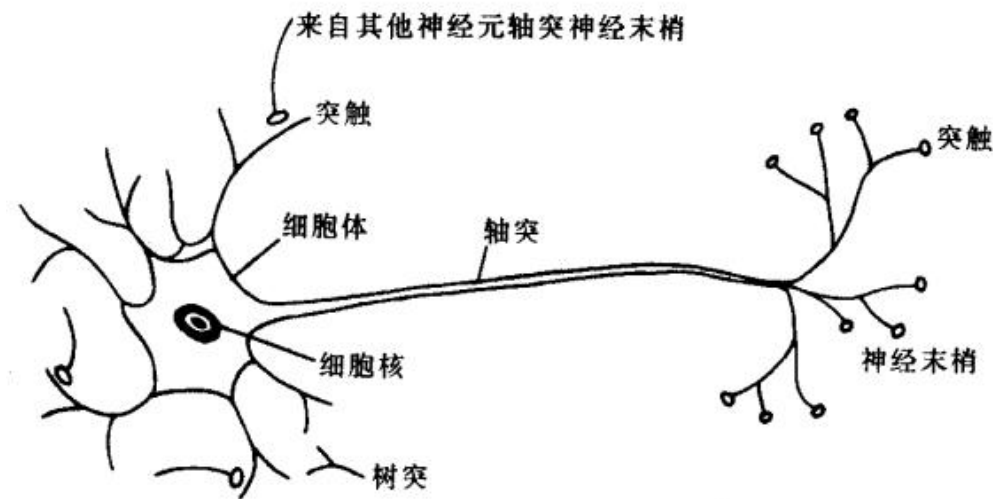
E-mail: [shuangli@bit.edu.cn](mailto:shuangli@bit.edu.cn)

Homepage: [shuangli.xyz](http://shuangli.xyz)

## 本节的主要内容—感知机

- 感知机的基本概念
- 感知机的模型构建
- 感知机的学习策略
- 感知机的学习算法

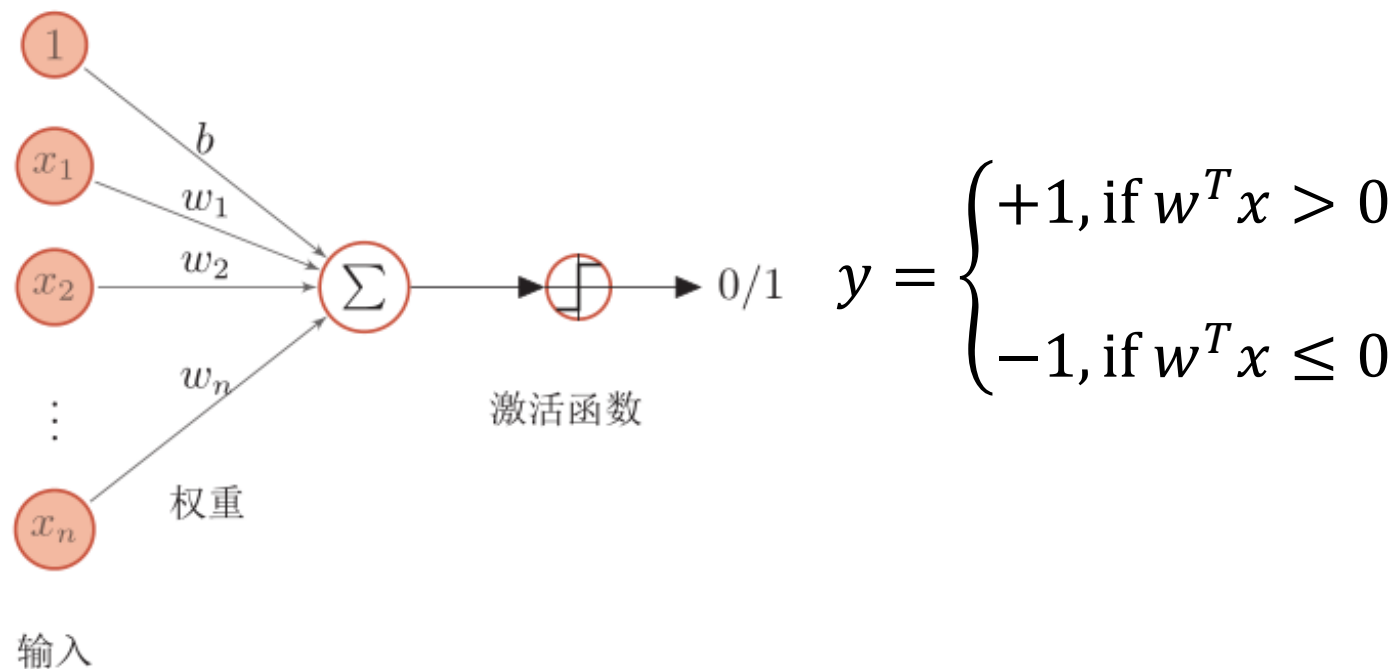
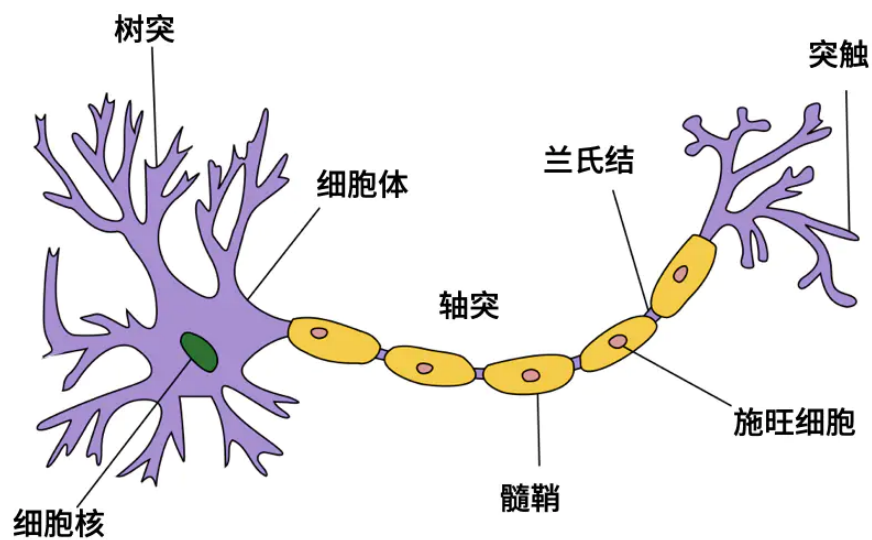
## 一、感知机的基本概念：神经元



- 神经网络中最基本的成分是**神经元(neuron)模型**。每个神经元与其他神经元相连，当它“兴奋”时，就会向相连的神经元发送化学物质，从而改变这些神经元内的电位；
- 如果某神经元的电位超过了一个**“阈值”(threshold)**，那么它就会被激活，即“兴奋”起来，向其他神经元发送化学物质。

## 一、感知机的基本概念：神经元

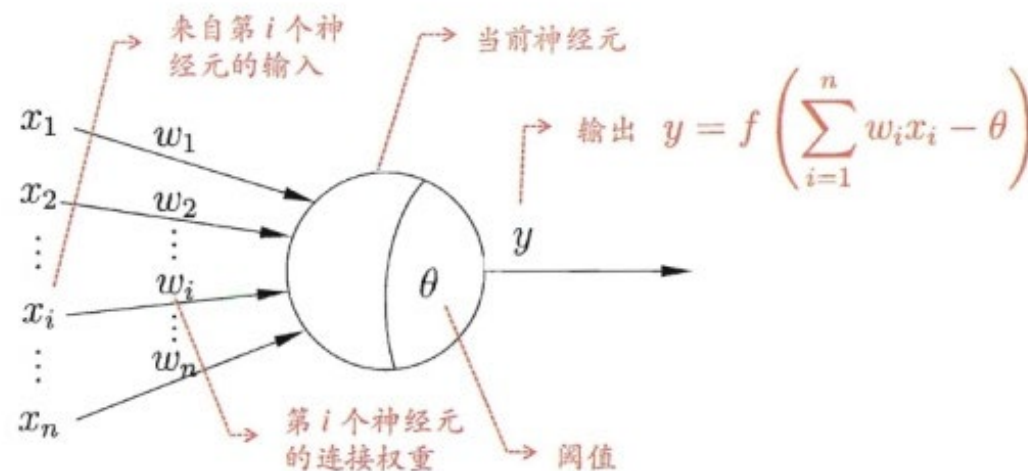
### 标准神经元结构



- 感知机是模拟生物神经元行为的机器，有与生物神经元相对应的部件，如**权重**（突触）、**偏置**（阈值）及**激活函数**（细胞体），输出为+1或-1。

## 一、感知机的基本概念：M-P 神经元模型

- 1943 年，McCulloch 麦克卡洛与 Pitts 皮茨将上述情形抽象为右图所示的简单模型，这就是一直沿用至今的“**M-P 神经元模型**”。

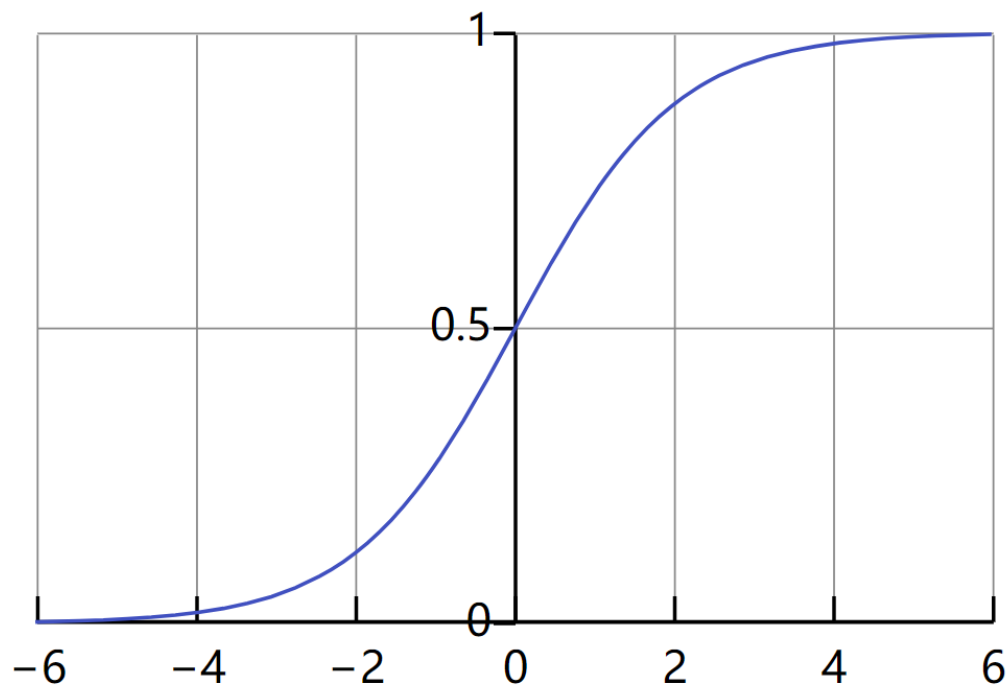


M-P 神经元模型

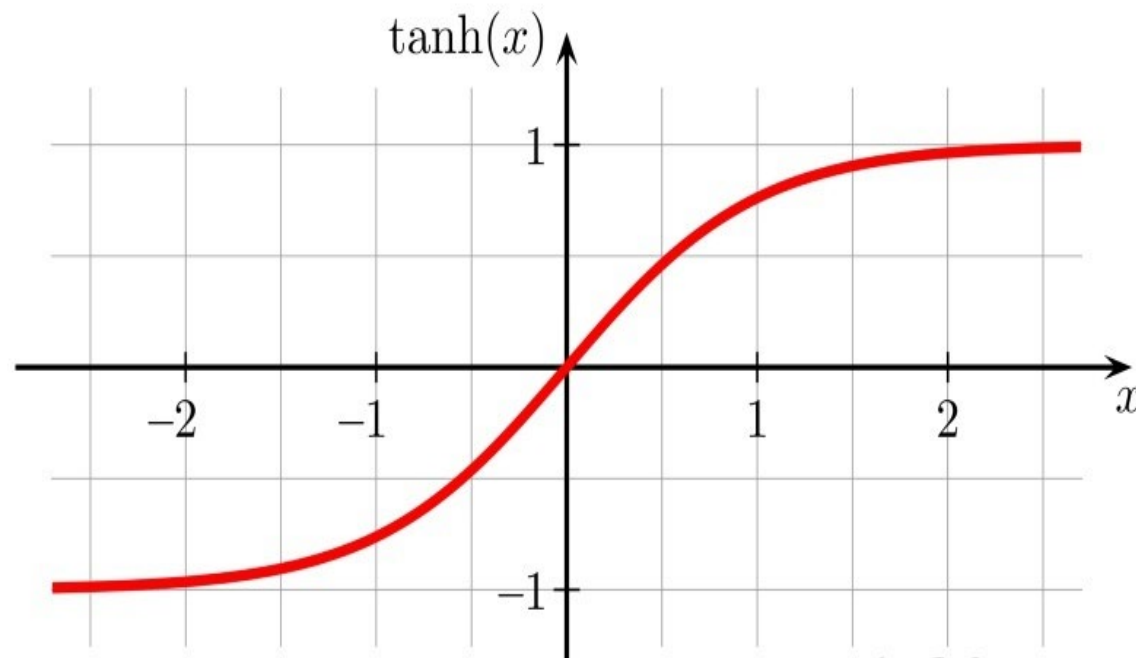
- 神经元接收到来自多个其他神经元传递过来的输入信号，这些输入信号通过带**权重**的**连接 (connection)**进行传递，神经元接收到的总输入值将与神经元的**阈值**进行比较，然后通过“**激活函数 (activation function)**”处理以产生神经元的输出。
- 多个这样的神经元按一定的**层次结构**连接起来，就得到了神经网络。

## 一、感知机的基本概念：激活函数

### 常见的神经元激活函数



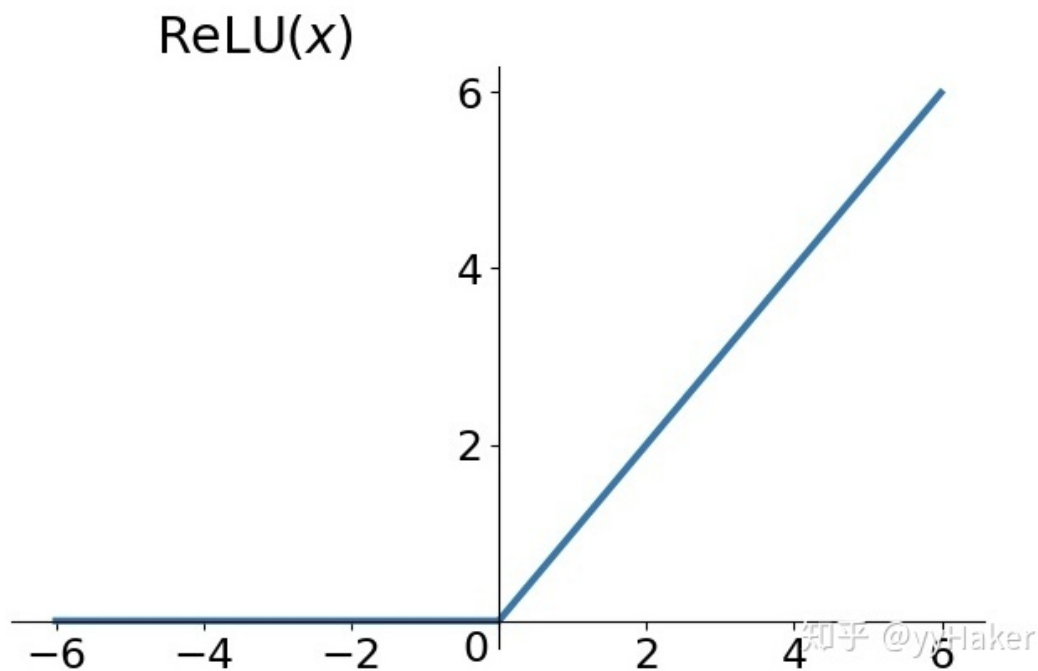
$$\text{Sigmoid: } f(x) = \frac{1}{1+e^{-x}}$$



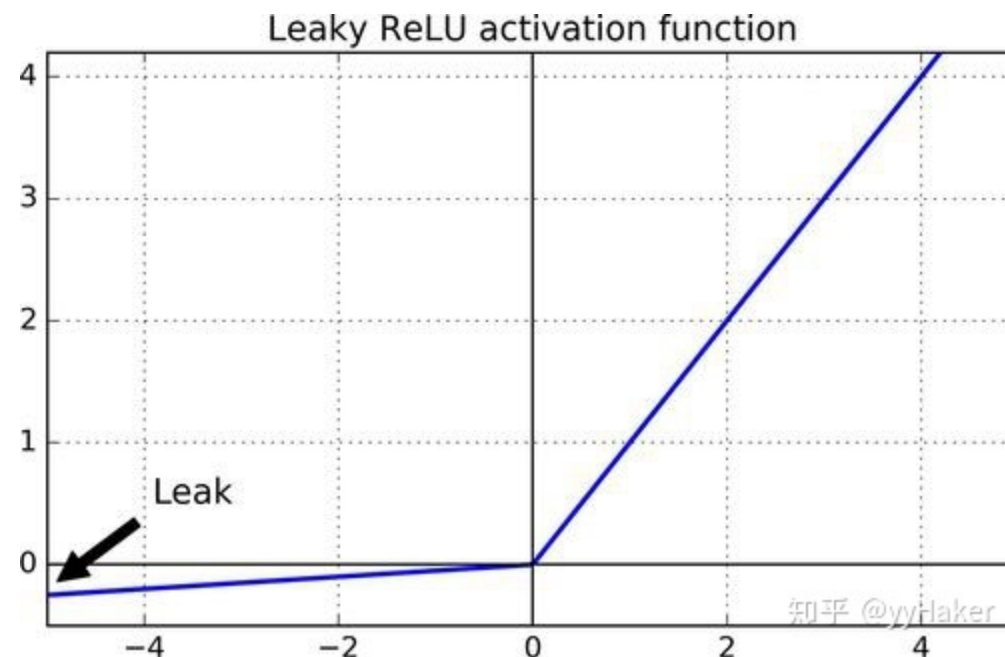
$$f(x) = \tanh(x)$$

## 一、感知机的基本概念：激活函数

### 常见的神经元激活函数




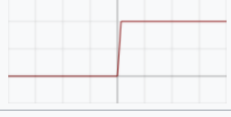
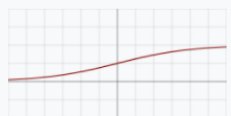



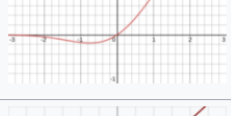
$$f(x) = \max(0, x)$$



$$f(x) = \begin{cases} ax, & x < 0 \\ x, & x \geq 0 \end{cases}$$


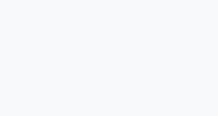

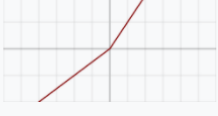
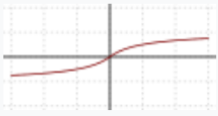



## 一、感知机的基本概念：激活函数

Name	Plot	Function, $f(x)$	Derivative of $f$ , $f'(x)$	Range
Identity		$x$	1	$(-\infty, \infty)$
Binary step		$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	$\begin{cases} 0 & \text{if } x \neq 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$	$\{0, 1\}$
Logistic, sigmoid, or soft step		$\sigma(x) = \frac{1}{1 + e^{-x}}$ <sup>[1]</sup>	$f(x)(1 - f(x))$	$(0, 1)$
tanh		$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$1 - f(x)^2$	$(-1, 1)$
Rectified linear unit (ReLU) <sup>[11]</sup>		$\begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ $= \max\{0, x\} = x\mathbf{1}_{x>0}$	$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$	$[0, \infty)$
Gaussian Error Linear Unit (GELU) <sup>[6]</sup>		$\frac{1}{2}x \left( 1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right) \right)$ $= x\Phi(x)$	$\Phi(x) + x\phi(x)$	$(-0.17\dots, \infty)$
Softplus <sup>[12]</sup>		$\ln(1 + e^x)$	$\frac{1}{1 + e^{-x}}$	$(0, \infty)$



## 一、感知机的基本概念：激活函数

Exponential linear unit (ELU) <sup>[13]</sup>		$\begin{cases} \alpha(e^x - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ with parameter $\alpha$	$\begin{cases} \alpha e^x & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ 1 & \text{if } x = 0 \text{ and } \alpha = 1 \end{cases}$	$(-\alpha, \infty)$
Scaled exponential linear unit (SELU) <sup>[14]</sup>		$\lambda \begin{cases} \alpha(e^x - 1) & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$ with parameters $\lambda = 1.0507$ and $\alpha = 1.67326$	$\lambda \begin{cases} \alpha e^x & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	$(-\lambda\alpha, \infty)$
Leaky rectified linear unit (Leaky ReLU) <sup>[15]</sup>		$\begin{cases} 0.01x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$	$\begin{cases} 0.01 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	$(-\infty, \infty)$
Parametric rectified linear unit (PReLU) <sup>[16]</sup>		$\begin{cases} \alpha x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$ with parameter $\alpha$	$\begin{cases} \alpha & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	$(-\infty, \infty)$ <sup>[2]</sup>
ElliotSig, <sup>[17][18]</sup> softsign <sup>[19][20]</sup>		$\frac{x}{1 +  x }$	$\frac{1}{(1 +  x )^2}$	$(-1, 1)$
Square nonlinearity (SQNL) <sup>[21]</sup>		$\begin{cases} 1 & \text{if } x > 2.0 \\ x - \frac{x^2}{4} & \text{if } 0 \leq x \leq 2.0 \\ x + \frac{x^2}{4} & \text{if } -2.0 \leq x < 0 \\ -1 & \text{if } x < -2.0 \end{cases}$	$1 \mp \frac{x}{2}$	$(-1, 1)$

## 一、感知机的基本概念：感知机

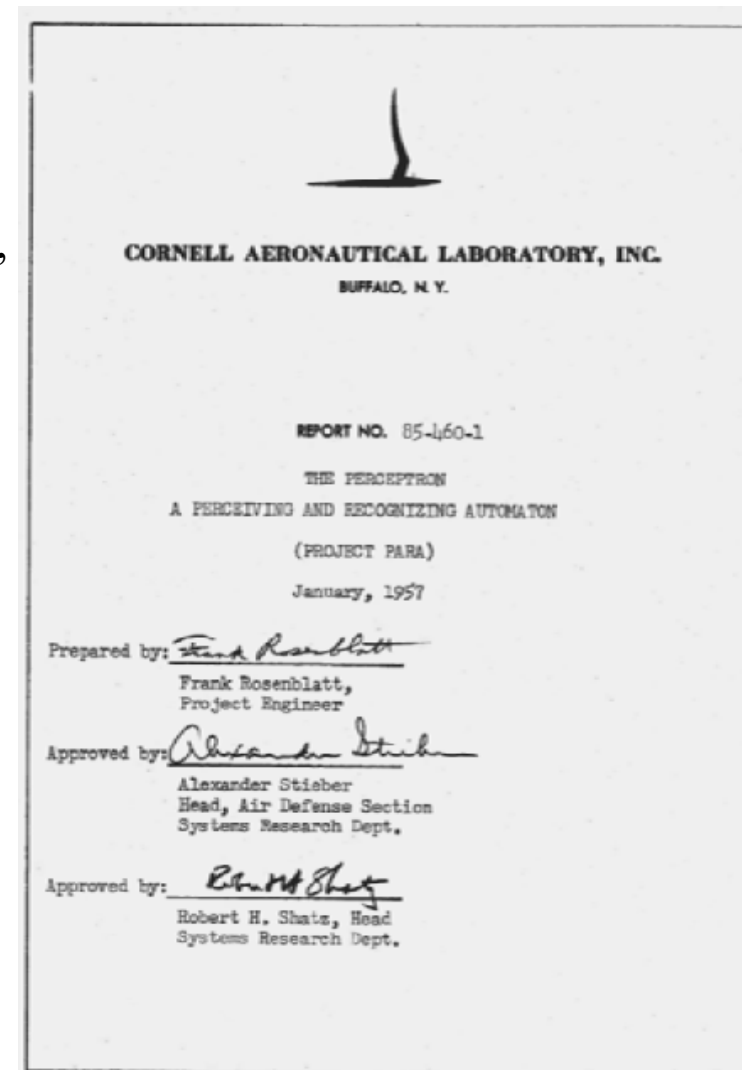
### 感知机（Perceptron）

Frank Rosenblatt, *The Perceptron – a perceiving and recognizing automaton*,  
Report 85-460-1, Cornell Aeronautical Laboratory, Jan. 1957

- 感知器：第一台学习机器

$$y = \text{sign} \left( \sum_{i=1} w_i x + w_0 \right)$$

- 为什么把它叫做学习机器？

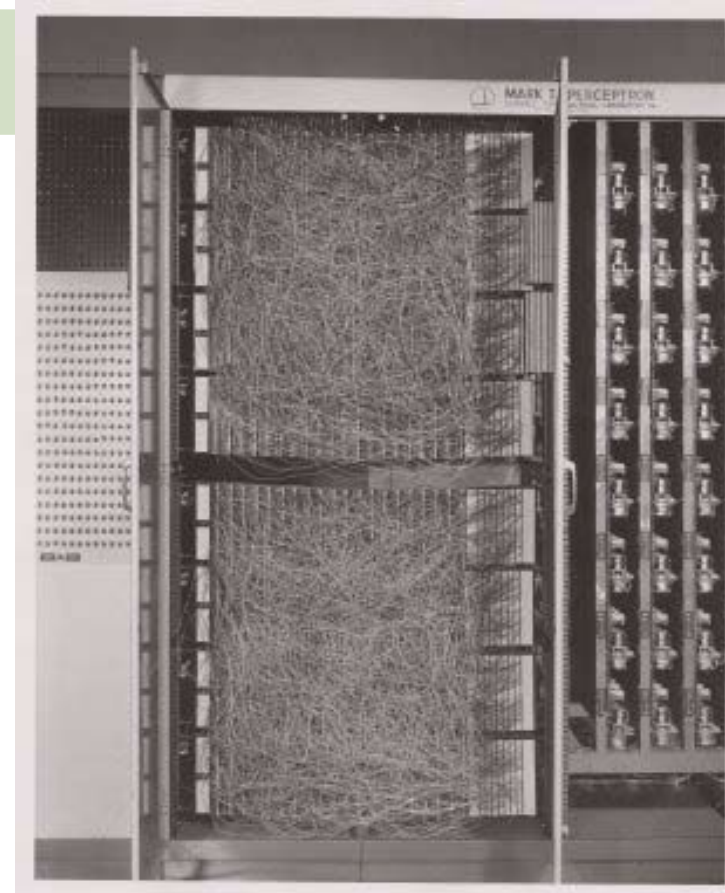
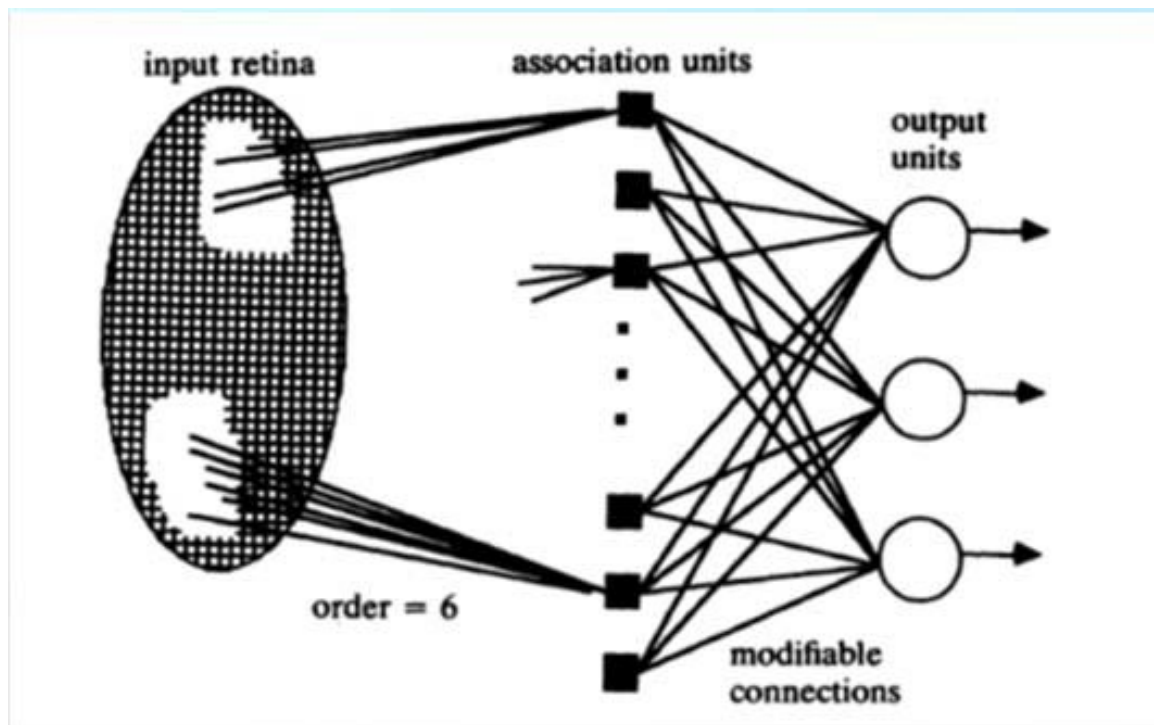


# 感知机与K近邻

## 一、感知机的基本概念：感知机

为什么把它叫做学习机器？

- 因为它是一台机器
- 因为它会学习



## 一、感知机的基本概念：感知机模型

- 感知机(Perceptron) 1957年由Rosenblatt提出，是神经网络与支持向量机的基础。
- 输入为实例的特征向量，输出为实例的类别，取+1和-1；
- 感知机对应于输入空间中将实例划分为正负两类的分离超平面，属于判别模型；
- 设计基于误分类的损失函数；
- 利用梯度下降法对损失函数进行极小化；
- 感知机学习算法具有简单而易于实现的优点，分为原始形式和对偶形式；

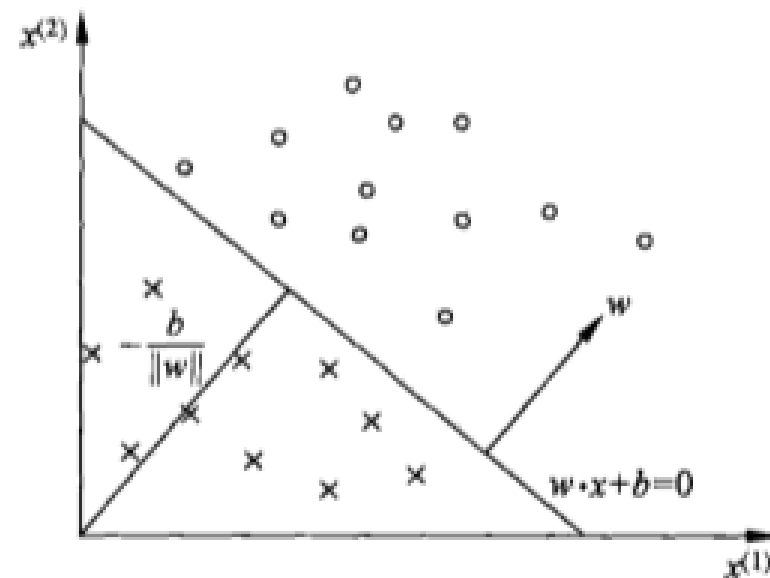


图 2.1 感知机模型

## 二、感知机的模型：感知机定义

### 定义(感知机):

- 假设输入空间(特征空间)是  $x \subseteq R^n$ ，输出空间是  $y = \{+1, -1\}$
- 输入  $x \in X$  表示实例的特征向量，对应于输入空间（特征空间）的点，输出  $y \in Y$  表示实例的类别，输入空间到输出空间的函数：

$$f(x) = \text{sign}(w \cdot x + b)$$

- 感知机中，模型参数： $w, b$ ，分别为模型权值向量与偏置
- 符号函数： $\text{sign}(x) = \begin{cases} +1, & x \geq 0 \\ -1, & x < 0 \end{cases}$

## 二、感知机的模型：几何解释

### 感知机几何解释：

- 线性方程：  $w \cdot x + b = 0$
- 对应于超平面 $S$ ，  $w$ 为法向量，  $b$ 为截距，分离正、负类：
- 分离超平面：  $w \cdot x + b = 0$

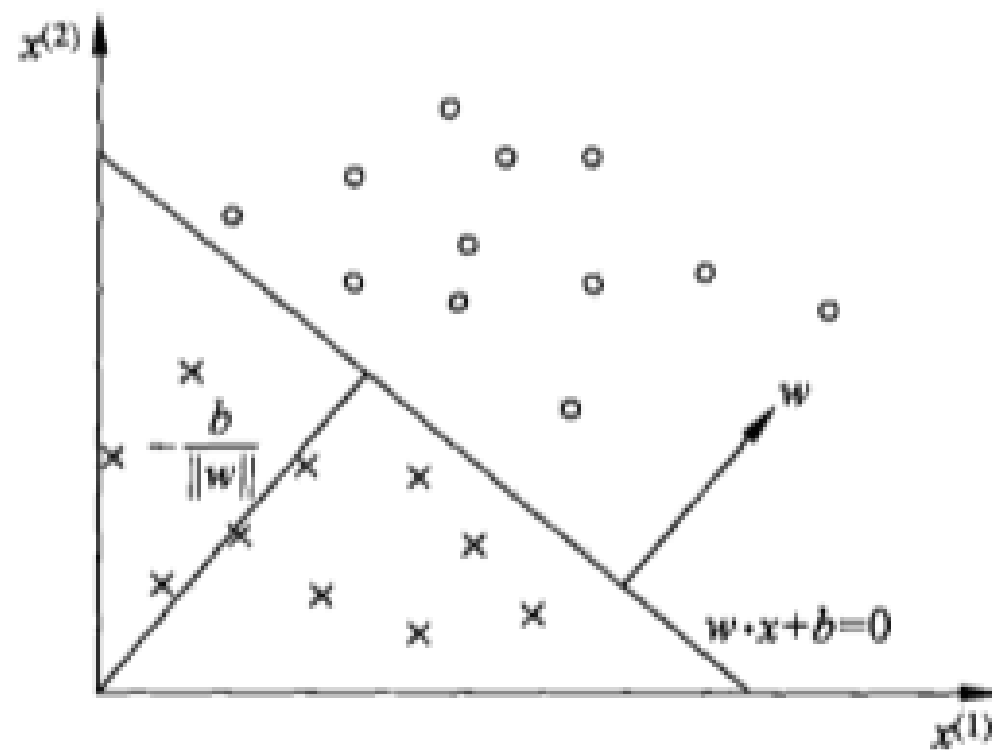


图 2.1 感知机模型

## 二、感知机的学习策略：数据集的线性可分性

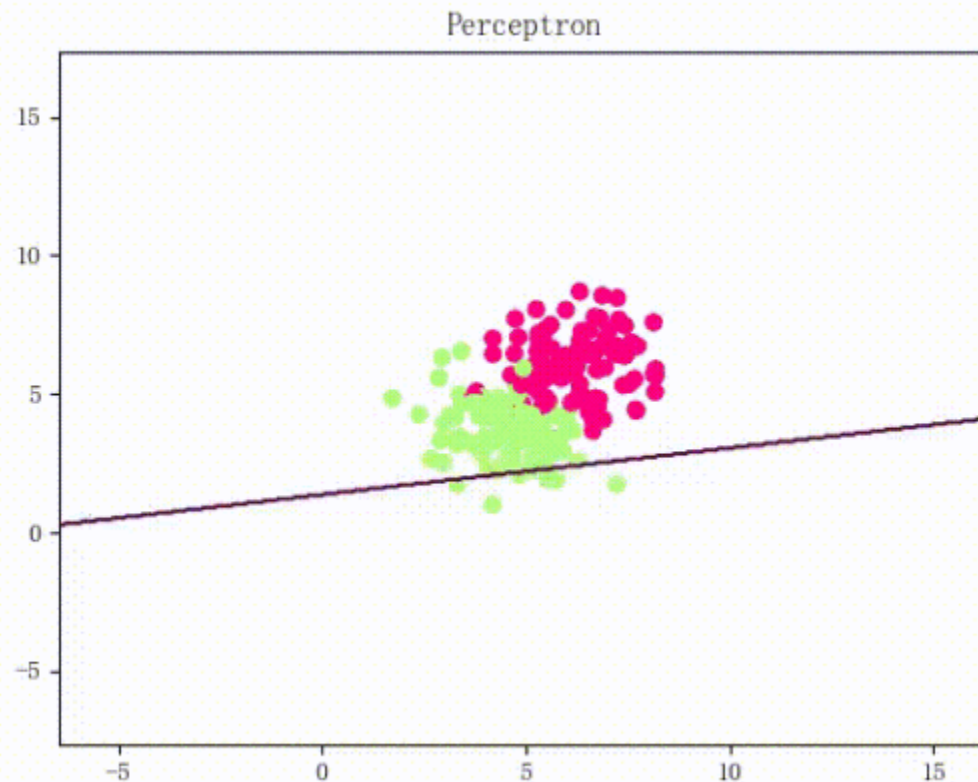
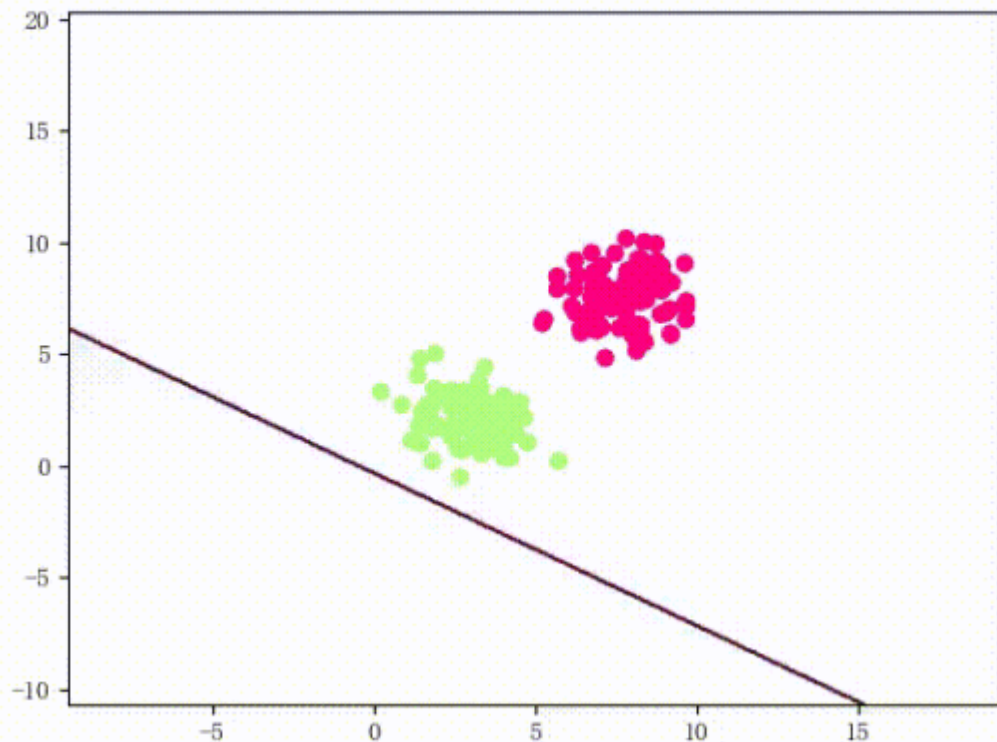
### 定义（数据集的线性可分性）

- 给定一个数据集  $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ ，其中， $x_i \subseteq R^n$ ， $y_i = \{+1, -1\}$ ， $i = 1, 2, \dots, N$ ，如果存在某个超平面  $w \cdot x + b = 0$ ，能够将数据集的正实例点和负实例点完全正确地划分到超平面的两侧；
- 即对所有  $y_i = +1$  的实例  $i$ ，有  $w \cdot x_i + b > 0$   
对所有  $y_i = -1$  的实例  $i$ ，有  $w \cdot x_i + b < 0$ ；
- 则称数据集  $T$  为线性可分数据集（linearly separable data set）；否则，称数据集  $T$  线性不可分。



## 三、感知机的学习策略

注：感知机只能处理数据集线性可分的情况



## 三、感知机的学习策略：损失函数

### 损失函数的定义：

- 假设训练数据集是线性可分的，感知机学习的目标是求得一个能够将训练集正实例点和负实例点完全正确分开的分离超平面。为了找出这样的超平面，即确定感知机模型参数 $w, b$ ，需要确定一个学习策略，即定义（经验）损失函数并将损失函数极小化。
- 损失函数的一个自然选择是误分类点的总数。但是，这样的损失函数不是参数 $w, b$ 的连续可导函数，不易优化。
- 损失函数的另一个选择是误分类点到超平面 $S$ 的总距离，这是感知机所采用的。

## 三、感知机的学习策略

- 为此，首先写出输入空间 $R^n$ 中任一点 $x_0$ 到超平面 $S$ 的距离：

$$\frac{1}{\|w\|} |w \cdot x_0 + b|$$

这里的 $\|w\|$ 是 $w$ 的L2范数

- 误分类点满足： $-y_i(w \cdot x_i + b) > 0$
- 误分类点到超平面距离： $-\frac{1}{\|w\|} y_i |w \cdot x_i + b|$
- 总距离： $-\frac{1}{\|w\|} \sum_{x_i \in M} y_i (w \cdot x_i + b)$ ， $M$ 为误分类点的集合

## 三、感知机的学习策略

求解最优化问题：  $\min_{w,b} L(w,b) = - \sum_{x_i \in M} y_i (w \cdot x_i + b)$

### 随机梯度下降法

- 首先任意选择一个超平面参数  $w, b$ ，然后不断最小化目标函数；
- 损失函数  $L$  的梯度：

$$\nabla_w L(w, b) = - \sum_{x_i \in M} y_i x_i$$

$$\nabla_b L(w, b) = - \sum_{x_i \in M} y_i$$

- 选取误分类点  $(x_i, y_i)$  更新  $w, b$ ：

$$w \leftarrow w + \eta y_i x_i, \quad b \leftarrow b + \eta y_i$$

其中， $\eta$  是步长，统计学习中又称为学习率 (learning rate)。

## 四、感知机的学习算法

### 感知机学习算法：

- 输入：  $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$

其中，  $x_i \subseteq R^n$ ，  $y_i = \{+1, -1\}$ ，  $i = 1, 2, \dots, N$

学习率  $\eta$  ( $0 < \eta \leq 1$ )

- 输出：  $w, b$  感知机模型  $f(x) = \text{sign}(w \cdot x + b)$

(1) 选取初值  $w_0, b_0$

(2) 在训练集中选取数据  $(x_i, y_i)$

(3) 如果  $y_i (w \cdot x + b) \leq 0$

$$w \leftarrow w + \eta y_i x_i \quad b \leftarrow b + \eta y_i$$

(4) 转至 (2)，直到训练集中没有误分类点

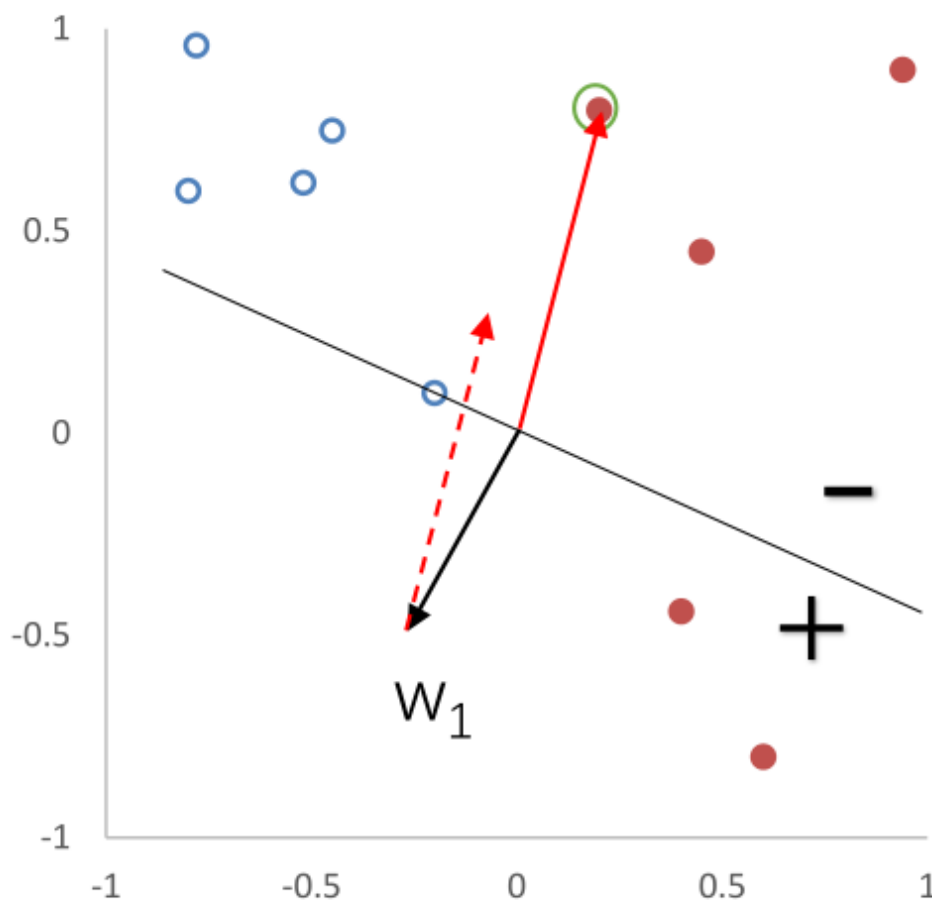
## 四、感知机的学习算法

感知机学习算法的直观的解释：

- 当一个实例点被**误分类**，即位于分离超平面的错误一侧时，则调整  $w, b$  的值，使分离超平面向该误分类点的一侧移动，以减少该误分类点与超平面间的距离，直至超平面**越过该误分类点**使其被正确分类；
- 上面的感知机学习的基本算法，对应于后面的对偶形式，称为原始形式，感知机学习算法简单且易于实现。

## 四、感知机的学习算法

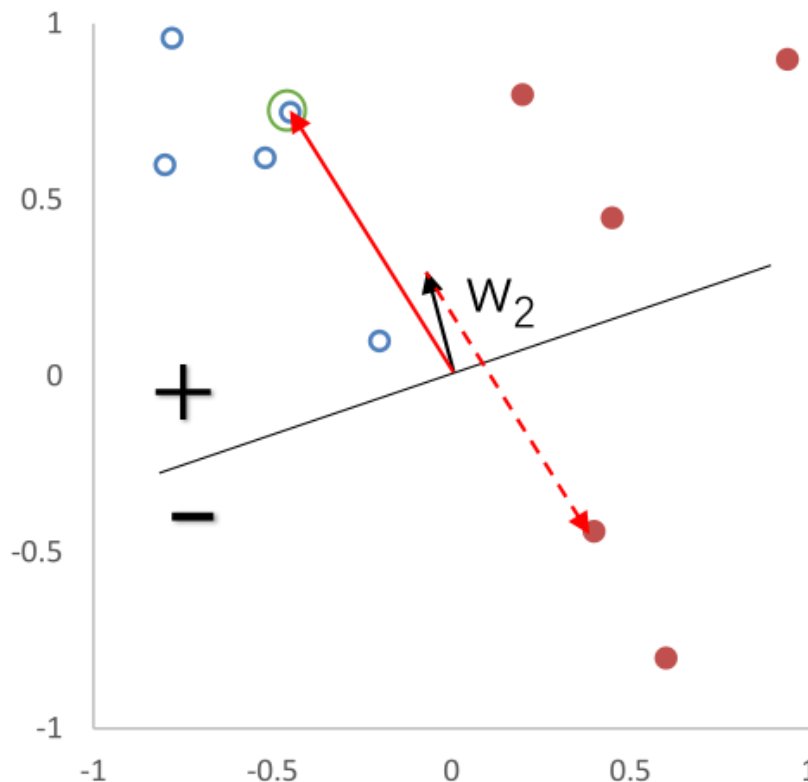
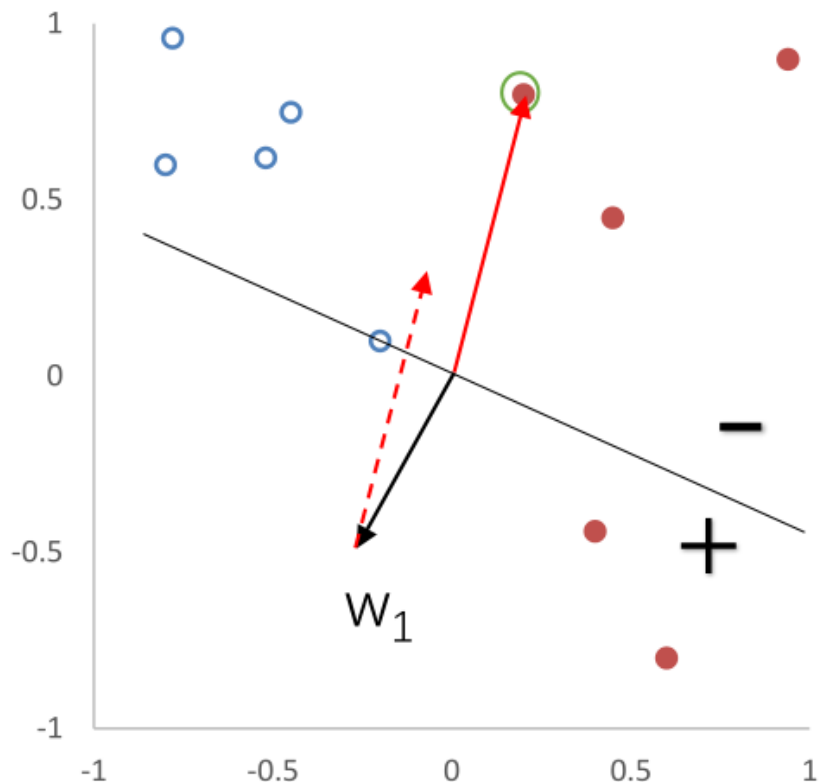
感知机学习算法是一种错误驱动的学习方法



- 红色实心点为正例
- 蓝色空心点为负例
- 黑色箭头表示权重向量
- 红色虚线箭头表示权重的更新方向

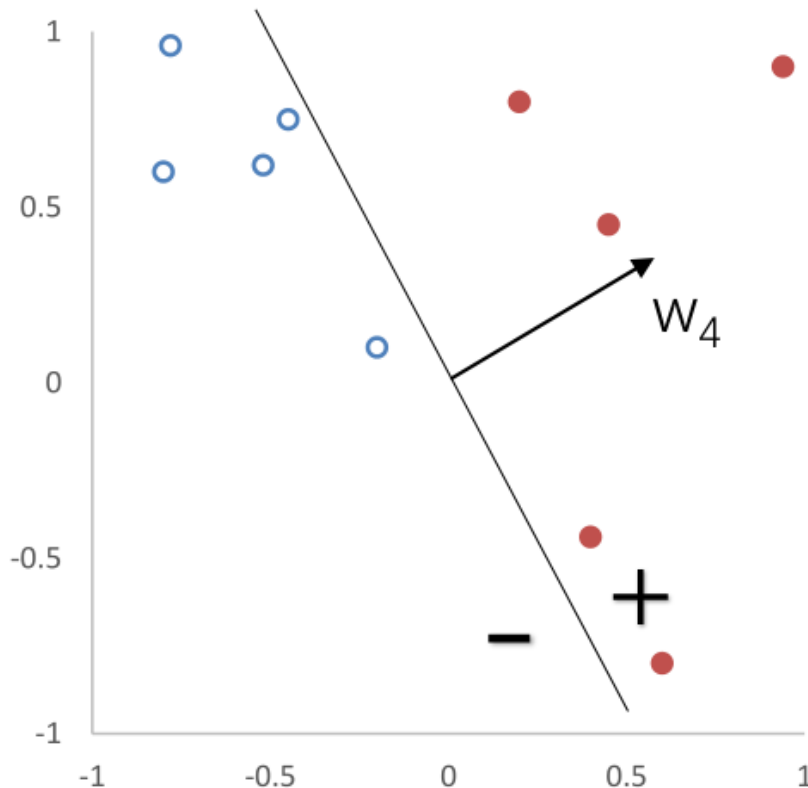
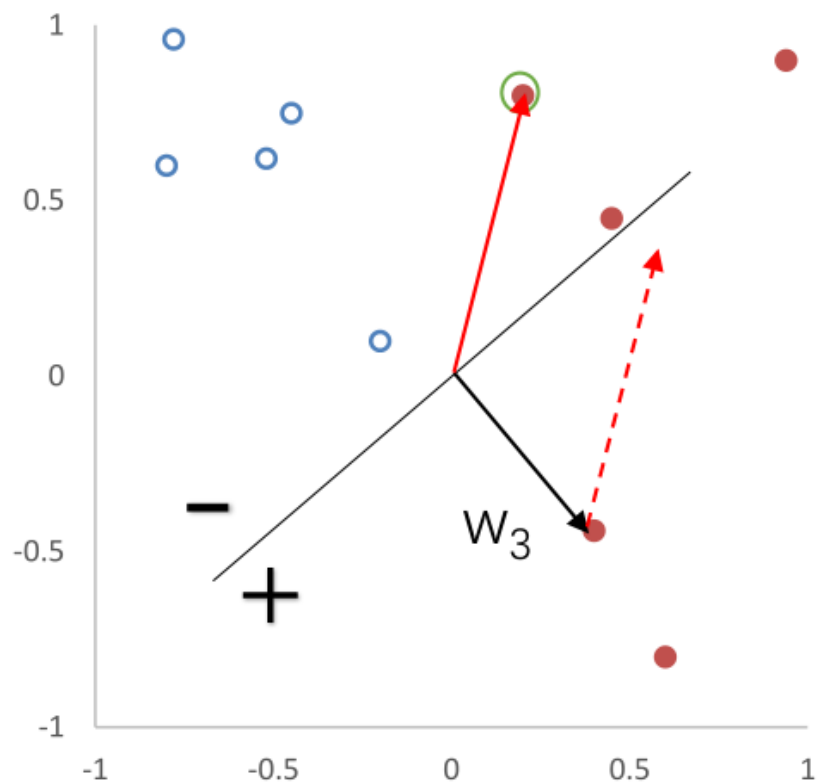


## 四、感知机的学习算法



- 红色实心点为正例
- 蓝色空心点为负例
- 黑色箭头表示权重向量
- 红色虚线箭头表示权重的更新方向

## 四、感知机的学习算法

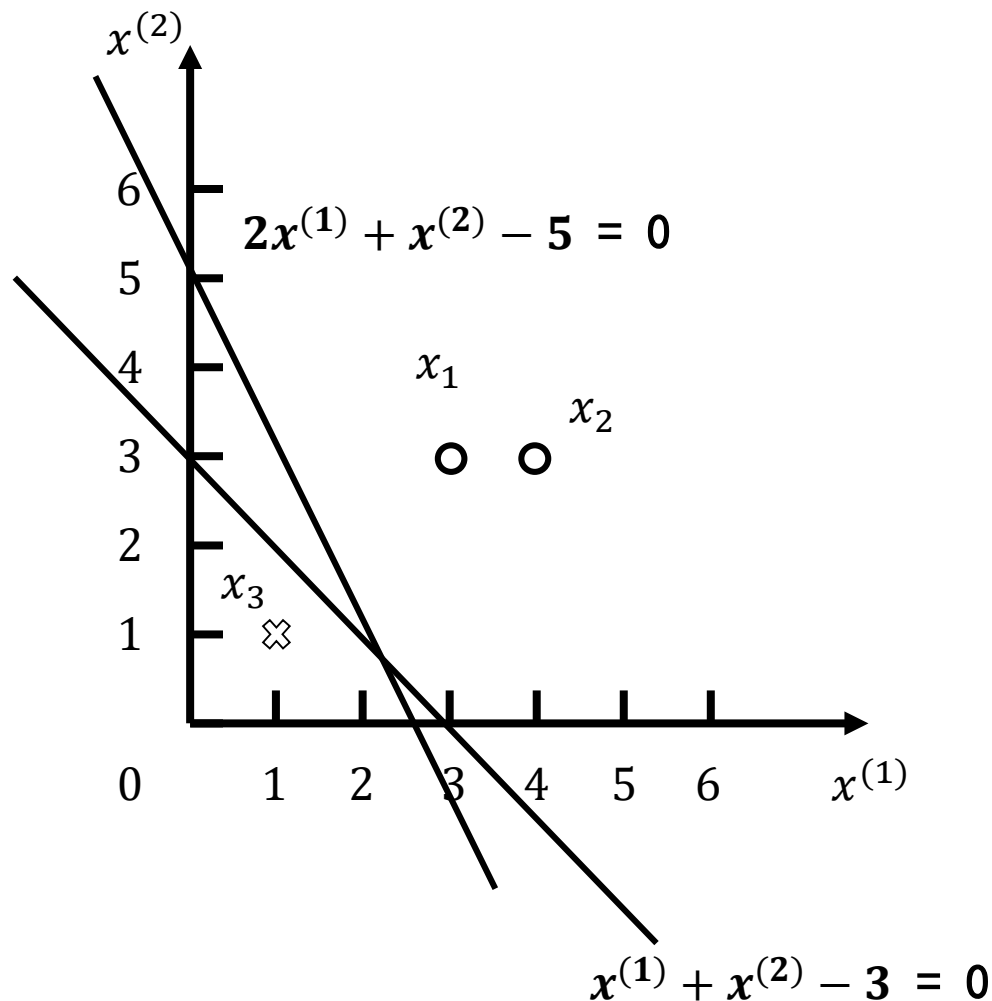


- 红色实心点为正例
- 蓝色空心点为负例
- 黑色箭头表示权重向量
- 红色虚线箭头表示权重的更新方向

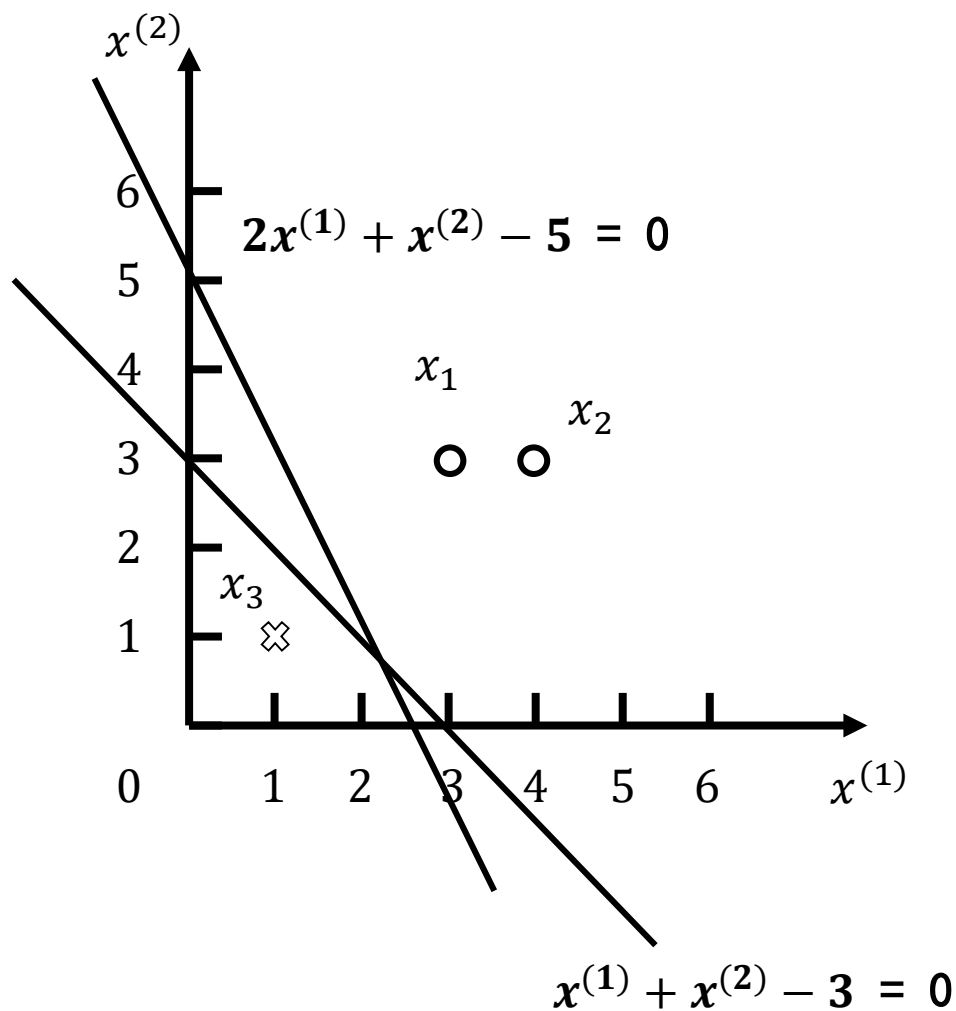
## 四、感知机的学习算法：例题

例题：

如图所示的训练数据集，其正实例点是  $x_1 = (3,3)^T$ ， $x_2 = (4,3)^T$ ，负实例点是  $x_3 = (1,1)^T$ ，试用感知机学习算法的原始形式求感知机模型。



## 四、感知机的学习算法：例题



解：构建最优化问题：

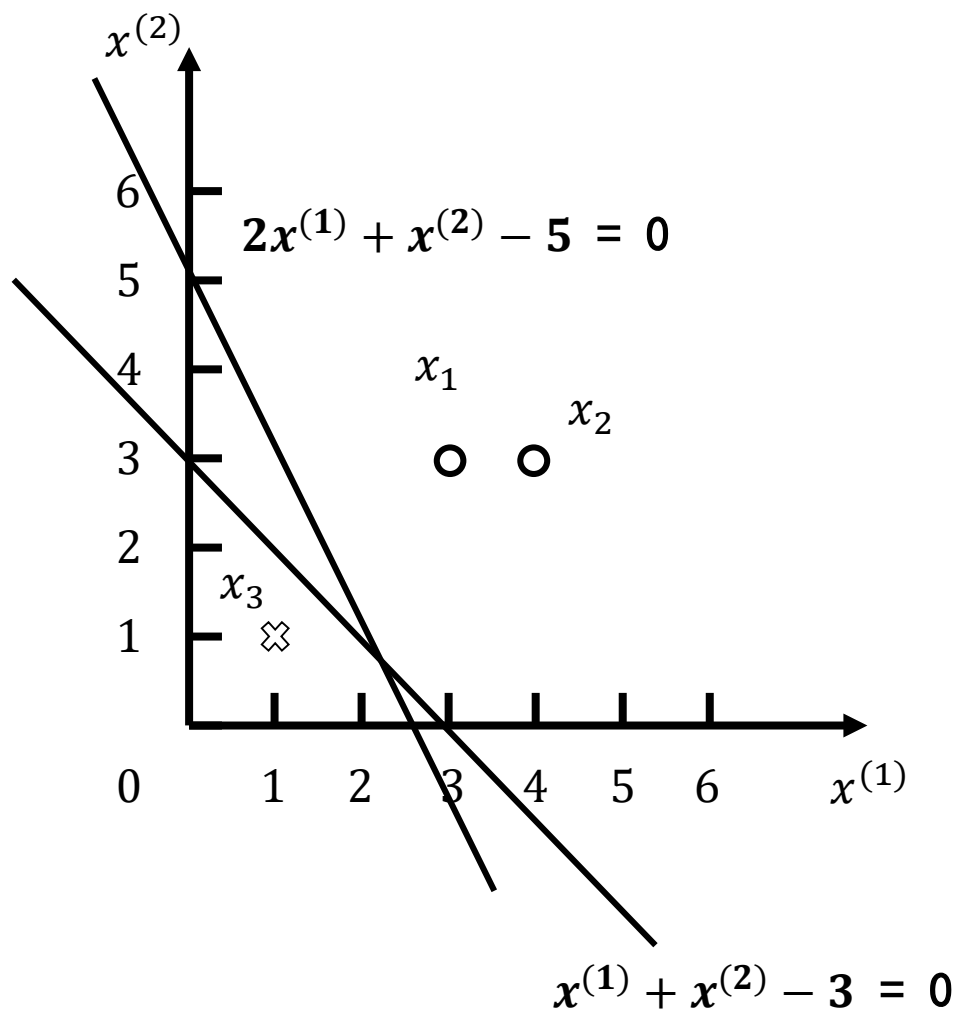
$$\min_{w, b} L(w, b) = - \sum_{x_i \in M} y_i (w \cdot x_i + b)$$

按照感知机学习算法的原始形式求感知机模型

设置学习率  $\eta = 1$ , 则迭代更新中：

$$w \leftarrow w + y_i x_i \quad b \leftarrow b + \eta y_i$$

## 四、感知机的学习算法：例题



(1) 选取初值  $w_0 = 0, b_0 = 0$

(2) 对于  $x_1 = (3, 3)^T$ :

$$y_1(w_0 \cdot x_1 + b_0) = 0$$

未能正确分类，更新  $w, b$

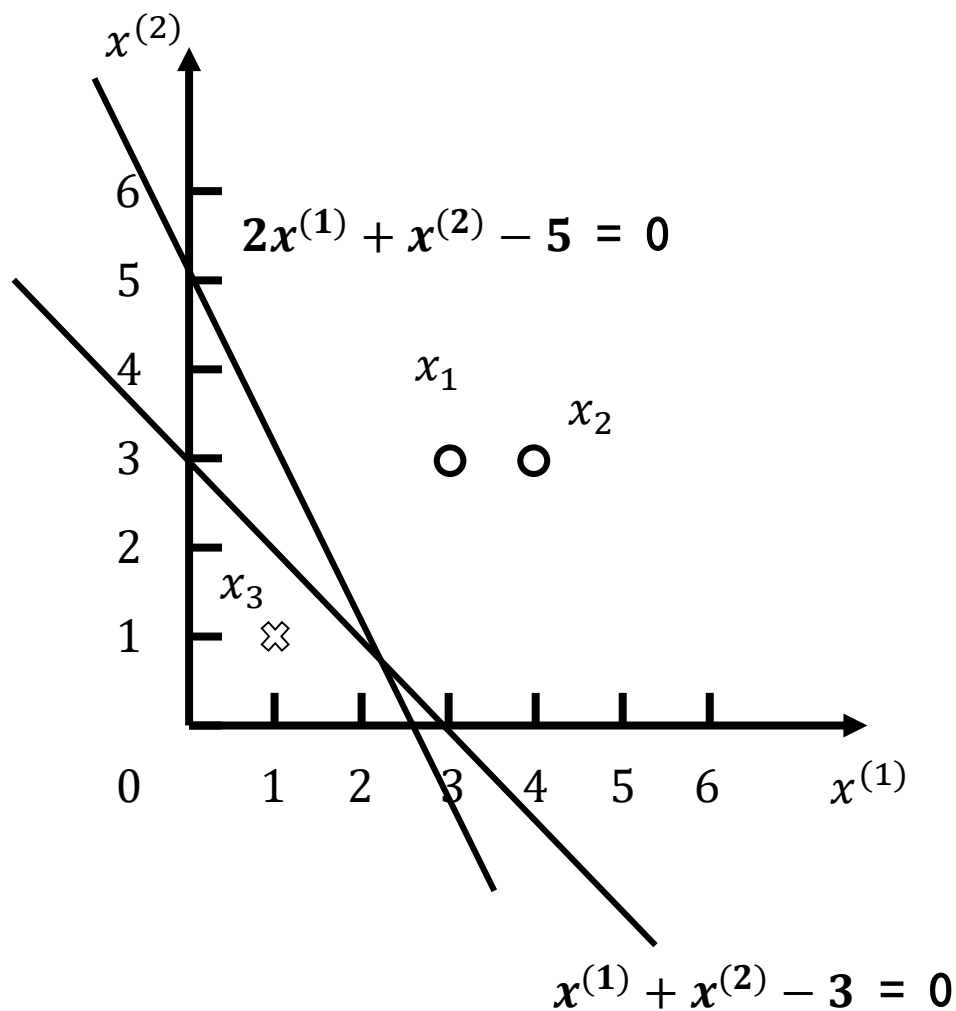
$$w_1 = w_0 + y_1 x_1 = (3, 3)^T$$

$$b_1 = b_0 + y_1 = 1$$

得到线性模型

$$w_1 \cdot x + b_1 = 3x^{(1)} + 3x^{(2)} + 1$$

## 四、感知机的学习算法：例题



(3) 对于  $x_1, x_2, y_i (w \cdot x_i + b) > 0$

被分类正确, 不修改  $w, b$

对于  $x_3 = (1, 1)^T$ :

$$y_3(w_1 \cdot x_3 + b_1) < 0$$

未能正确分类, 更新  $w, b$

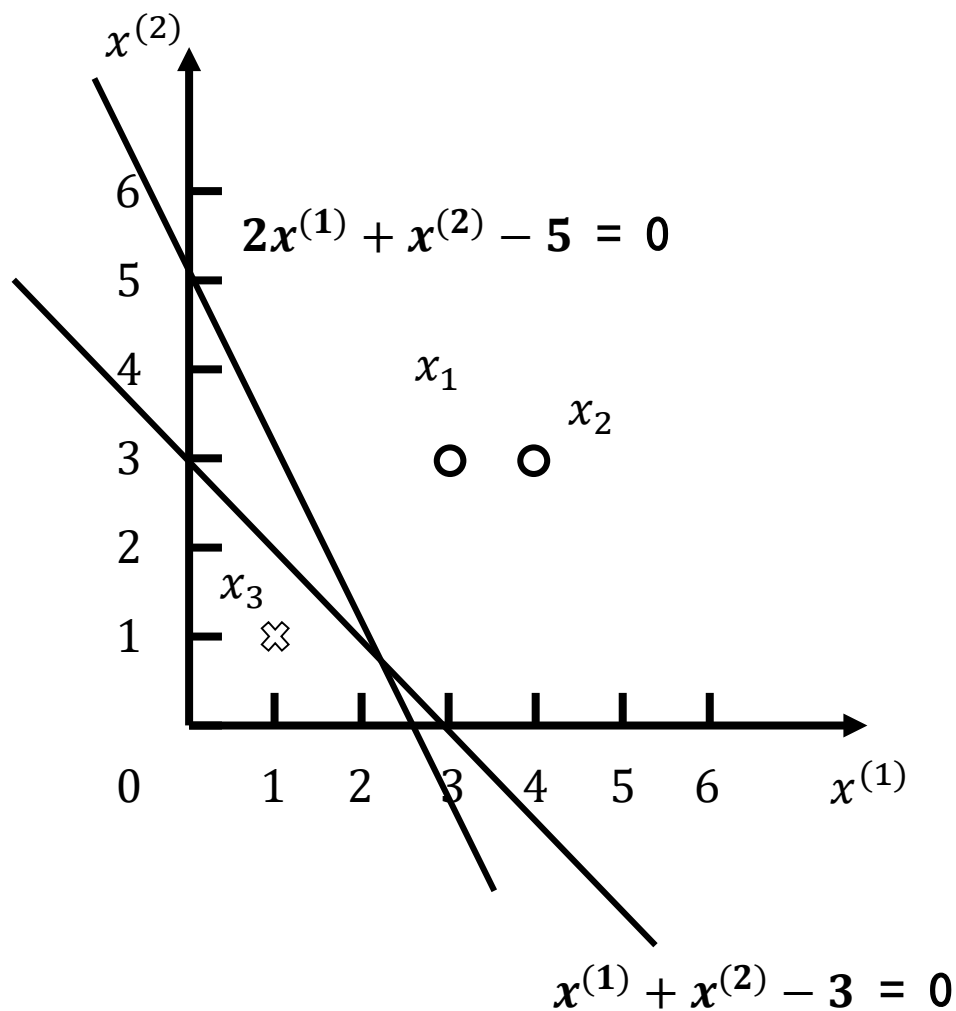
$$w_2 = w_1 + y_3 x_3 = (2, 2)^T$$

$$b_1 = b_0 + y_3 = 0$$

得到线性模型

$$w_2 \cdot x + b_2 = 2x^{(1)} + 2x^{(2)}$$

## 四、感知机的学习算法：例题



(4) 如此下去，直到

$$w_7 = (1, 1)^T$$

$$b_7 = -3$$

得到线性模型

$$w_7 \cdot x + b_7 = x^{(1)} + x^{(2)} - 3$$

对所有数据点，均有

$$y_i(w_7 \cdot x_i + b_7) > 0$$

没有误分类点，损失函数达到极小。

分离超平面： $x^{(1)} + x^{(2)} - 3 = 0$

感知机模型： $f(x) = \text{sign}(x^{(1)} + x^{(2)} - 3)$



## 四、感知机的学习算法：例题

### 例题的求解迭代过程

迭代次数	误分类点	$w$	$b$	$w \cdot x + b$
0		0	0	0
1	$x_1$	$(3,3)^T$	1	$3x^{(1)} + 3x^{(2)} + 1$
2	$x_3$	$(2,2)^T$	0	$2x^{(1)} + 2x^{(2)}$
3	$x_3$	$(1,1)^T$	-1	$x^{(1)} + x^{(2)} - 1$
4	$x_3$	$(0,0)^T$	-2	-2
5	$x_1$	$(3,3)^T$	-1	$3x^{(1)} + 3x^{(2)} - 1$
6	$x_3$	$(2,2)^T$	-2	$2x^{(1)} + 2x^{(2)} - 2$
7	$x_3$	$(1,1)^T$	-3	$x^{(1)} + x^{(2)} - 3$
8	0	$(1,1)^T$	-3	$x^{(1)} + x^{(2)} - 3$

## 四、感知机的学习算法

- 感知机算法的基本算法中：选取**误分类点**更新，

$$w \leftarrow w + \eta y_i x_i \quad b \leftarrow b + \eta y_i$$

逐步修改 $w, b$ ，设修改 $n$ 次，则 $w, b$ 关于 $(x_i, y_i)$ 的**增量**分别是 $\alpha_i y_i x_i$ 和 $\alpha_i y_i$ ，这里 $\alpha_i = n_i \eta$ ，若假设 $w, b$ 初值均为0

- 这样，从学习过程不难看出，最后学习到的 $w, b$ 可以分别表示为

$$w = \sum_{i=1}^N \alpha_i y_i x_i, \quad b = \sum_{i=1}^N \alpha_i y_i$$

- 这里， $\alpha_i \geq 0$ ， $i = 1, 2, \dots, N$ ， $n_i$ 表示第 $i$ 个实例点由于误分而进行更新的次数。**实例点更新次数越多，意味着它距离分离超平面越近，也就越难正确分类。**换句话说，这样的实例对学习结果影响最大。

## 四、感知机的学习算法：对偶形式

### 感知机学习算法的对偶形式：

- 输入：  $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$

其中，  $x_i \subseteq R^n$ ，  $y_i = \{+1, -1\}$ ，  $i = 1, 2, \dots, N$ ， 学习率  $\eta$  ( $0 < \eta \leq 1$ )

- 输出：  $\alpha, b$  感知机模型  $f(x) = \text{sign}(w \cdot x + b)$

(1) 初值  $\alpha \leftarrow 0$ ,  $b \leftarrow 0$

(2) 在训练集中选取数据  $(x_i, y_i)$

(3) 如果  $y_i \left( \sum_{j=1}^n \alpha_j y_j x_j \cdot x_i + b \right) \leq 0$

$$\alpha_i \leftarrow \alpha_i + \eta \quad b \leftarrow b + \eta y_i$$

(4) 转至 (2)，直到训练集中没有误分类点

## 四、感知机的学习算法：对偶形式

- 对偶形式中训练实例仅以**内积**的形式出现。
- 为了方便，可以预先将训练集中实例间的内积计算出来并以矩阵的形式存储，这个矩阵就是所谓的**Gram矩阵**(Gram matrix)

$$G = [x_i \cdot x_j]_{N \times N} = \begin{bmatrix} x_1 \cdot x_1 & \dots & x_1 \cdot x_N \\ \vdots & \ddots & \vdots \\ x_N \cdot x_1 & \dots & x_N \cdot x_N \end{bmatrix}_{N \times N}$$

- 根据Gram矩阵，可以快速求出所有样本的内积值

## 四、感知机的学习算法：例题

例题：

如图所示的训练数据集，其正实例点是 $x_1 = (3,3)^T$ ， $x_2 = (4,3)^T$ ，负实例点是 $x_3 = (1,1)^T$ ，试用感知机学习算法的对偶形式求感知机模型。

使用感知机学习算法的对偶形式：

(1) 取 $\alpha_i = 0, i = 1, 2, 3$

$$b_0 = 0, \quad \eta = 1$$

(2) 计算Gram矩阵

$$G = \begin{bmatrix} 18 & 21 & 6 \\ 21 & 25 & 7 \\ 6 & 7 & 2 \end{bmatrix}$$

(3) 误分条件：

$$y_i \left( \sum_{j=1}^n \alpha_j y_j x_j x_i + b \right) \leq 0$$

## 四、感知机的学习算法：例题

例题：

如图所示的训练数据集，其正实例点是 $x_1 = (3,3)^T$ ， $x_2 = (4,3)^T$ ，负实例点是 $x_3 = (1,1)^T$ ，试用感知机学习算法的对偶形式求感知机模型。

参数更新：

$$\alpha_i \leftarrow \alpha_i + \eta \quad b \leftarrow b + \eta y_i$$

(4) 迭代……

$$(5) \quad w = 2x_1 + 0x_2 - 5x_3 = (1, 1)^T$$
$$b = -3$$

分离超平面： $x^{(1)} + x^{(2)} - 3 = 0$

感知机模型： $f(x) = \text{sign}(x^{(1)} + x^{(2)} - 3)$

## 本节的主要内容—K近邻

- K近邻算法基本原理
- K近邻算法模型推导
- K近邻算法的实现：kd树原理的讲解
- K近邻算法的拓展



五、K近邻算法基本原理

近朱者赤，近墨者黑



## 五、K近邻算法基本原理：算法原理

## K-Nearest Neighbors Algorithm, KNN

### • 算法原理

- 存在一个样本数据集合，也称作**训练样本集**，并且样本集中每个数据都存在标签，即我们知道样本集中每个数据与**所属分类**的对应关系。
- **输入**没有标签的**新数据**后，将新数据的每个特征与样本集中数据对应的特征进行比较，然后算法提取样本集中特征**最相似数据（最近邻）**的分类标签。
- 一般来说，只选择样本数据集中**前K个最相似的数据**。最后，选择k个中出现次数最多的分类，作为新数据的分类。

## 五、K近邻算法基本原理：算法

### 算法（K近邻法）

- **输入**：训练数据集  $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ ；实例特征向量  $x$

其中  $x_i$  为实例的**特征向量**，  $y_i$  为实例的**类别**，  $i = 1, 2, \dots, N$

- **输出**：实例  $x$  的所属类别  $y$

(1) 根据给定的**距离度量**，在训练集  $T$  中**找出与  $x$  最邻近的  $k$  个点**，涵盖这  $k$  个点的  $x$  的**邻域**记作  $N_k(x)$

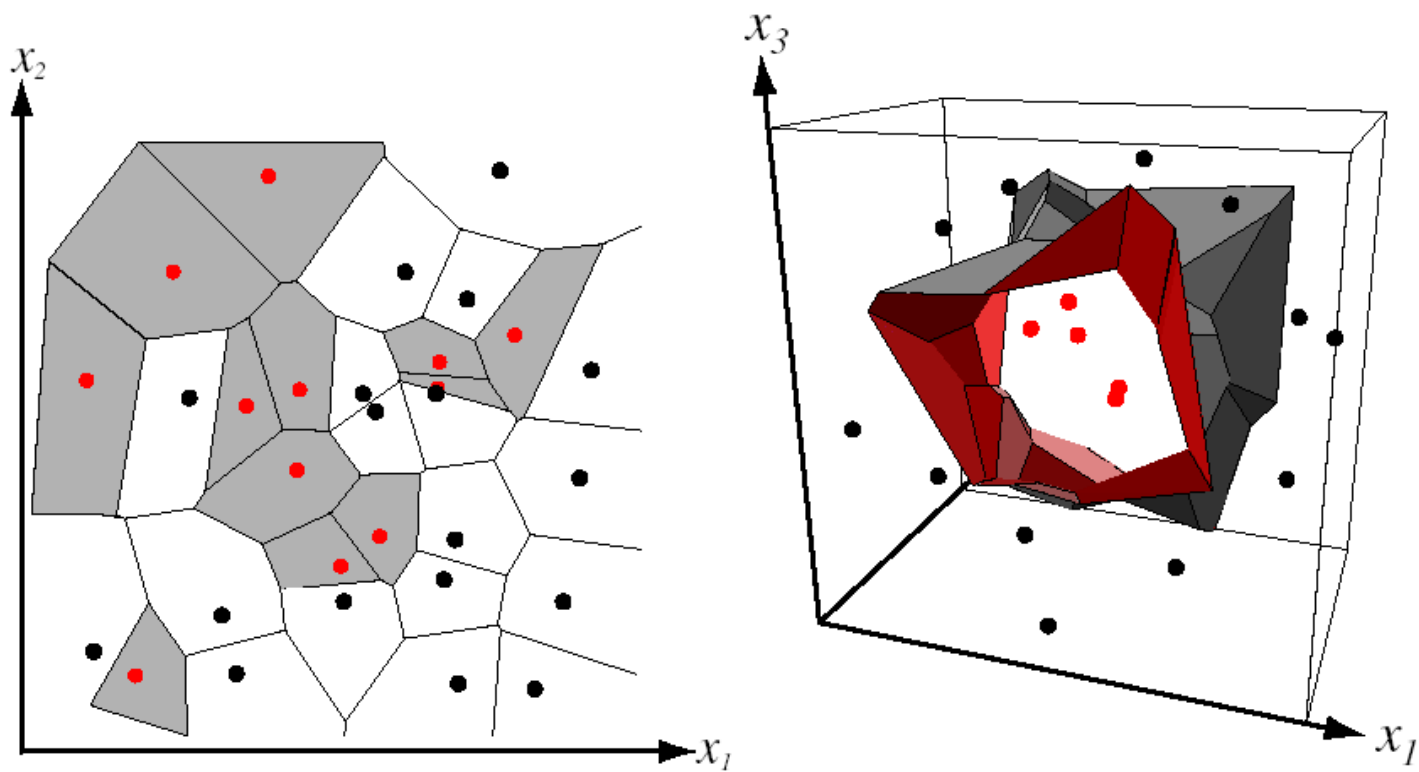
(2) 在  $N_k(x)$  中根据**分类决策规则**（如多数表决）**决定  $x$  的类别  $y$**

$$y = \arg \max_{C_j} \sum_{x_i \in N_k(x)} I(y_i = C_j), \quad i = 1, 2, \dots, N, \quad j = 1, 2, \dots, K$$

其中， $I$  为**指示函数**，即当  $y_i = C_j$  时  $I$  为 1，否则  $I$  为 0.

## 六、K近邻算法模型：模型三要素

- K近邻的模型三要素：**距离度量**，**k值选择**，**分类决策规则**

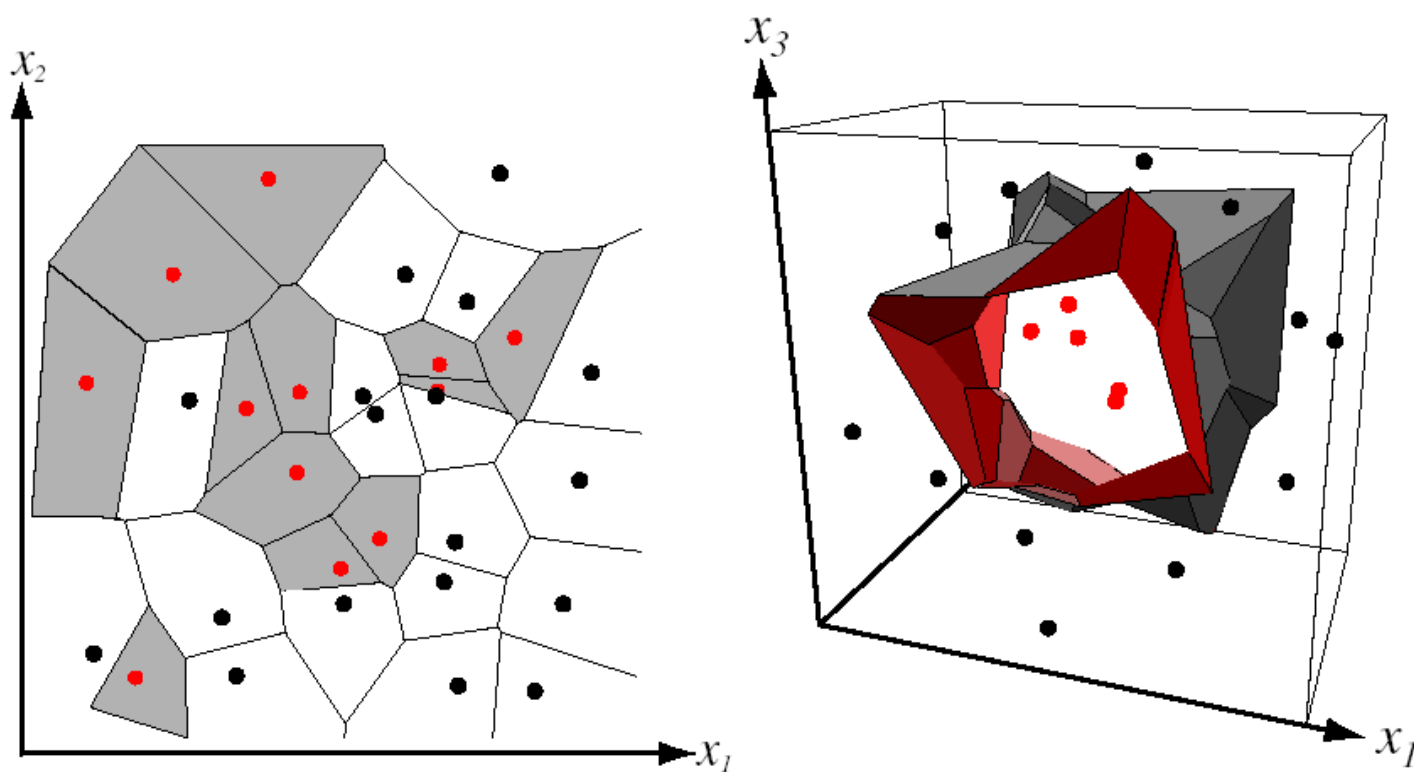


K近邻的模型对应特征空间的划分示例

k近邻法中，当**训练集**、**距离度量**（如欧氏距离）、**k值**及**分类决策规则**（如多数表决）确定后，对于任何一个新的输入实例，它所属的类别唯一地确定。这相当于根据上述要素将**特征空间**划分为一些**子空间**，确定子空间里的每个点所属的类别。

## 六、K近邻算法模型：模型三要素

- K近邻的模型三要素：距离度量，k值选择，分类决策规则

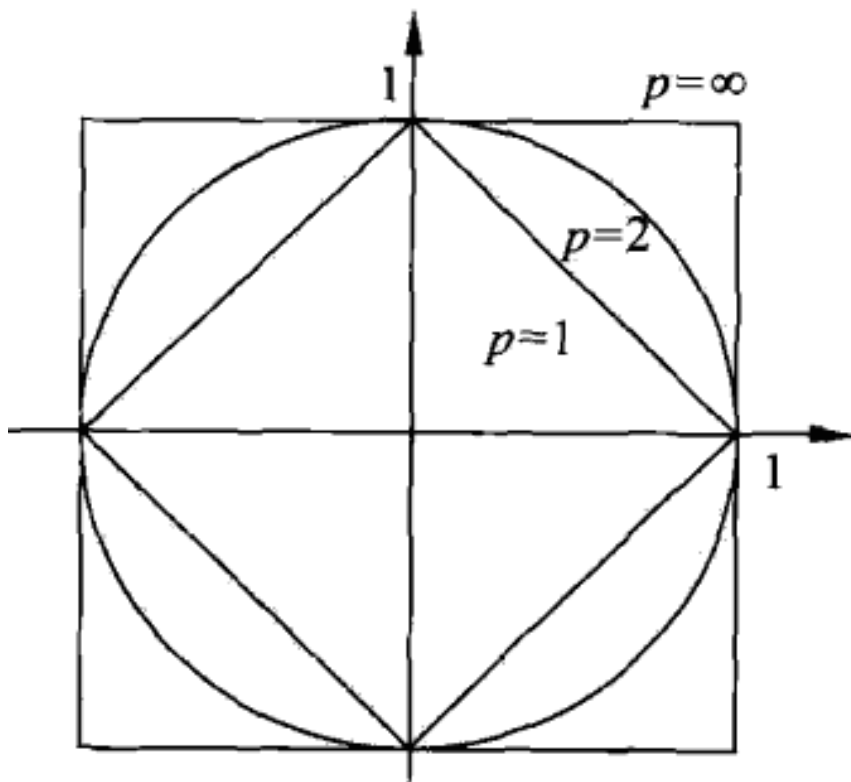


K近邻的模型对应特征空间的划分示例

特征空间中，对每个训练实例点  $x_i$ ，距离该点比其他点更近的所有点组成一个区域，叫作单元 (cell)。每个训练实例点拥有一个单元，所有训练实例点的单元构成对特征空间的一个划分。最近邻法将实例  $x_i$  的类  $y_i$  作为其单元中所有点的类标记 (class label)。这样，每个单元的实例点的类别是确定的。

## 六、K近邻算法模型：距离度量

### 距离度量



与原点的 $L_p$ 距离为1  
( $L_p=1$ ) 的点的图形

$$x_i = (x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(n)})^T$$

●  $L_p$  距离

$$L_p(x_i, x_j) = \left( \sum_{l=1}^n |x_i^{(l)} - x_j^{(l)}|^p \right)^{\frac{1}{p}}$$

● 欧式距离

$$L_2(x_i, x_j) = \left( \sum_{l=1}^n |x_i^{(l)} - x_j^{(l)}|^2 \right)^{\frac{1}{2}}$$

● 曼哈顿距离

$$L_1(x_i, x_j) = \sum_{l=1}^n |x_i^{(l)} - x_j^{(l)}|$$

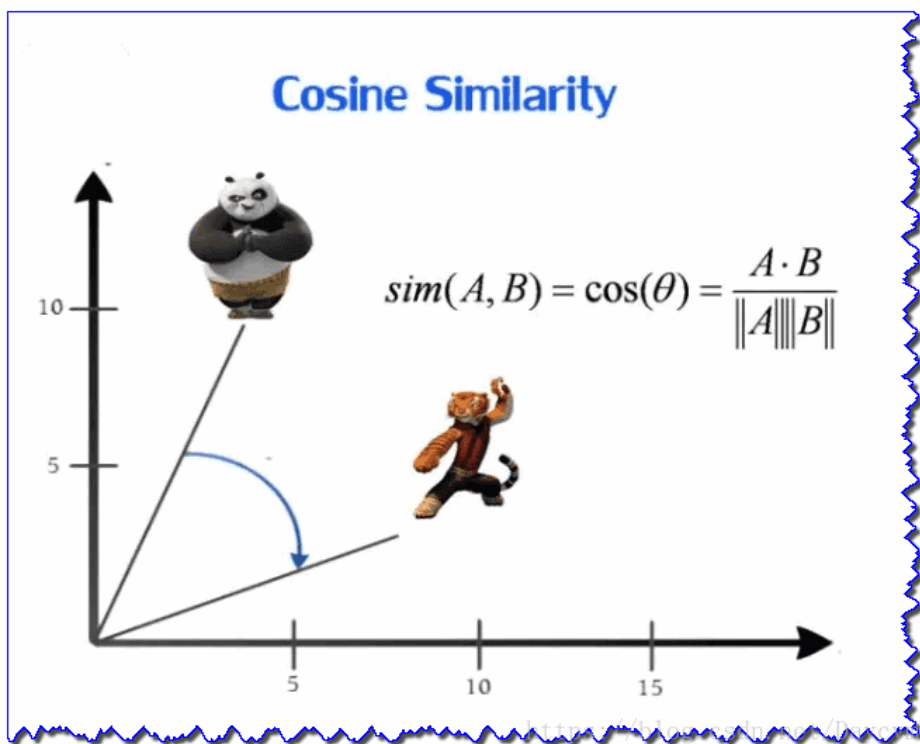
●  $L_\infty$  距离

$$L_\infty(x_i, x_j) = \max_l |x_i^{(l)} - x_j^{(l)}|$$



## 六、K近邻算法模型：距离度量

### 距离度量



- 余弦相似度用向量空间中两个向量夹角的余弦值作为衡量两个个体间差异的大小。
- 相比距离度量，余弦相似度更加注重两个向量在方向上的差异，而非距离或长度上。典型的应用场景就是人脸识别（arcface）。
- 余弦值的范围在 $[-1, 1]$ 之间，值越趋近于1，代表两个向量的方向越接近；越趋近于-1，他们的方向越相反；接近于0，表示两个向量近乎于正交。

## 六、K近邻算法模型：例题

距离度量例子：

已知二维空间的3个点  $x_1 = (1,1)^T$ ,  $x_2 = (5,1)^T$ ,  $x_3 = (4,4)^T$ ，试

求在  $P$  取不同值时， $L_p$  距离下  $x_1$  的最近邻点

$$L_1(x_1, x_2) = 4 \quad L_2(x_1, x_2) = 4 \quad L_3(x_1, x_2) = 4 \quad L_4(x_1, x_2) = 4$$

$$L_1(x_1, x_3) = 6 \quad L_2(x_1, x_3) = 4 \quad L_3(x_1, x_3) = 3.78 \quad L_4(x_1, x_3) = 3.57$$

- $P$  等于1或2时， $x_2$  是  $x_1$  的最近邻点；
- $P$  大于等于3时， $x_3$  是  $x_1$  的最近邻点。

由不同的距离度量所确定的最近邻点是不同的。



## 六、K近邻算法模型：k值选择

### • K值选择

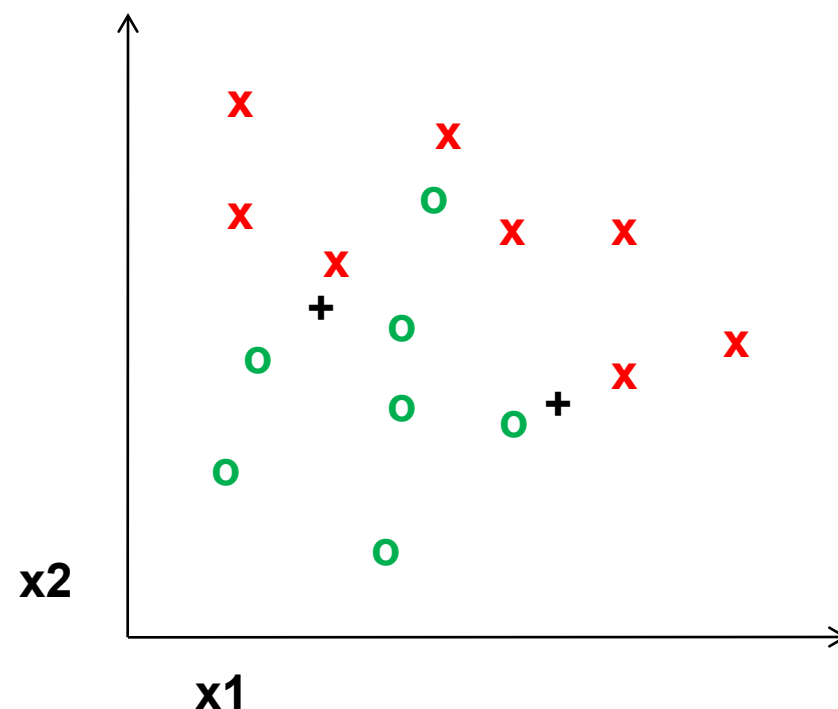
- 如果选择较小的K值
  - “学习”的近似误差 (approximation error) 会减小，但 “学习”的估计误差 (estimation error) 会增大；
  - 噪声敏感；
  - K值的减小就意味着整体的模型变得复杂，容易发生过拟合。
- 如果选择较大的K值
  - 减少学习的估计误差，但缺点是学习的近似误差会增大；
  - K值的增大就意味着整体的模型变得简单。

## 六、K近邻算法模型：近似误差与估计误差

- **近似误差**：可以理解为对现有**训练集**的训练误差。
  - 近似误差，更关注于“训练”。
  - 如果近似误差小了会出现过拟合的现象，对现有的训练集能有很好的预测，但是对未知的测试样本将会出现较大偏差的预测。模型本身不是最接近最佳模型。
- **估计误差**：可以理解为对**测试集**的测试误差。
  - 估计误差，更关注于“测试”、“泛化”。
  - 估计误差小了说明对未知数据的预测能力好。模型本身最接近最佳模型。

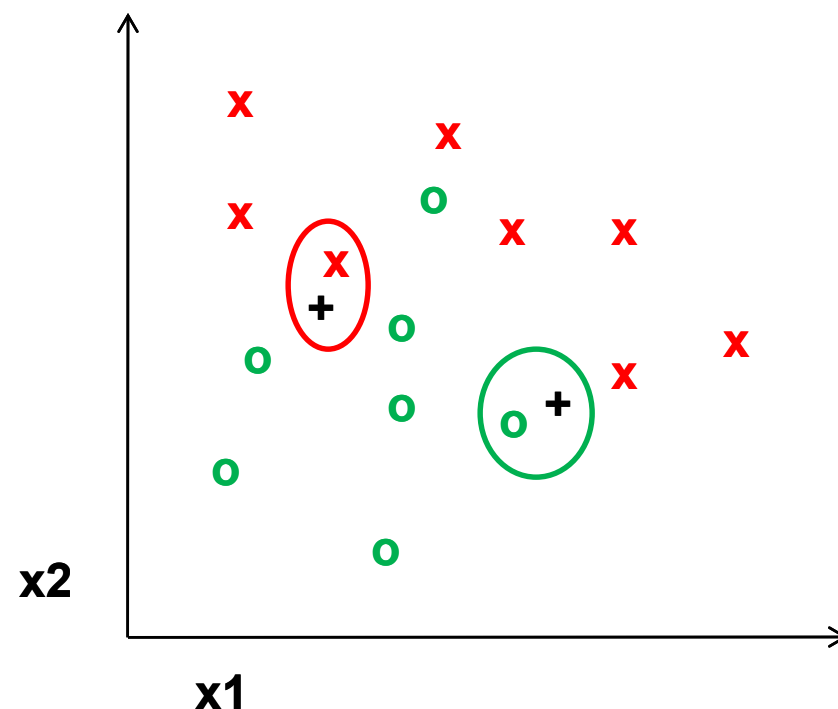
## 六、K近邻算法模型：k值选择

### K-nearest neighbor



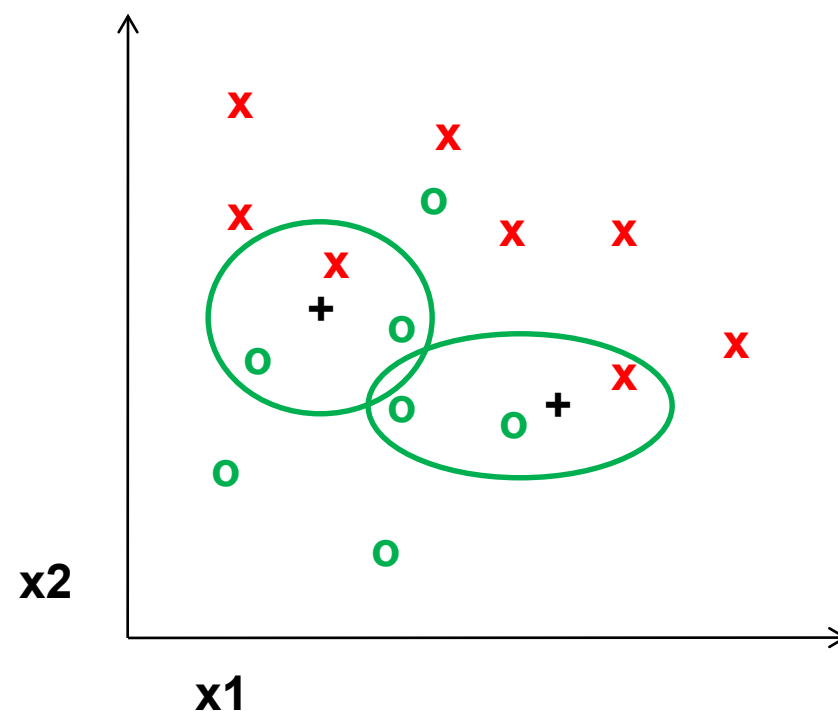
## 六、K近邻算法模型：k值选择

### 1-nearest neighbor



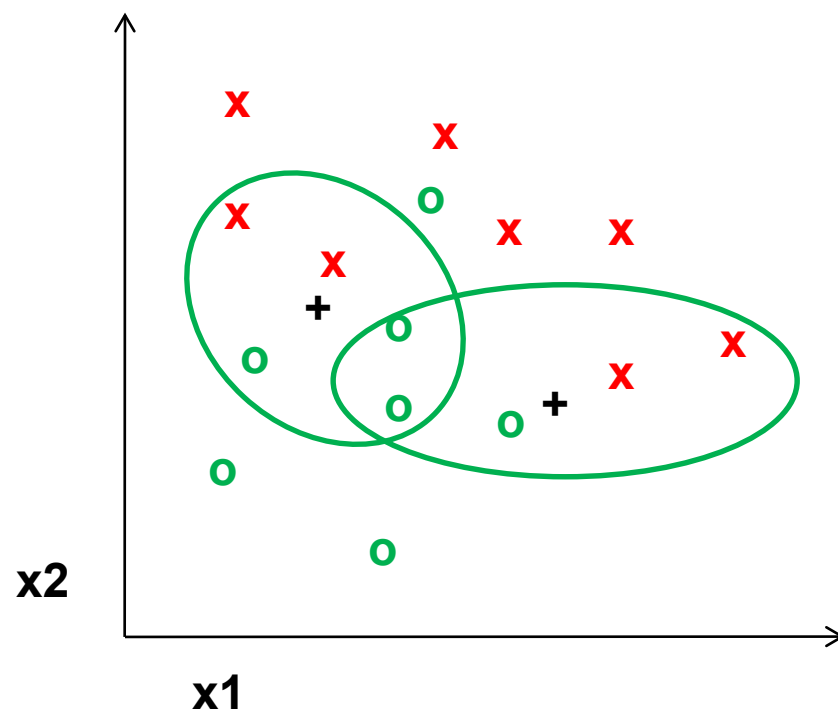
## 六、K近邻算法模型：k值选择

### 3-nearest neighbor

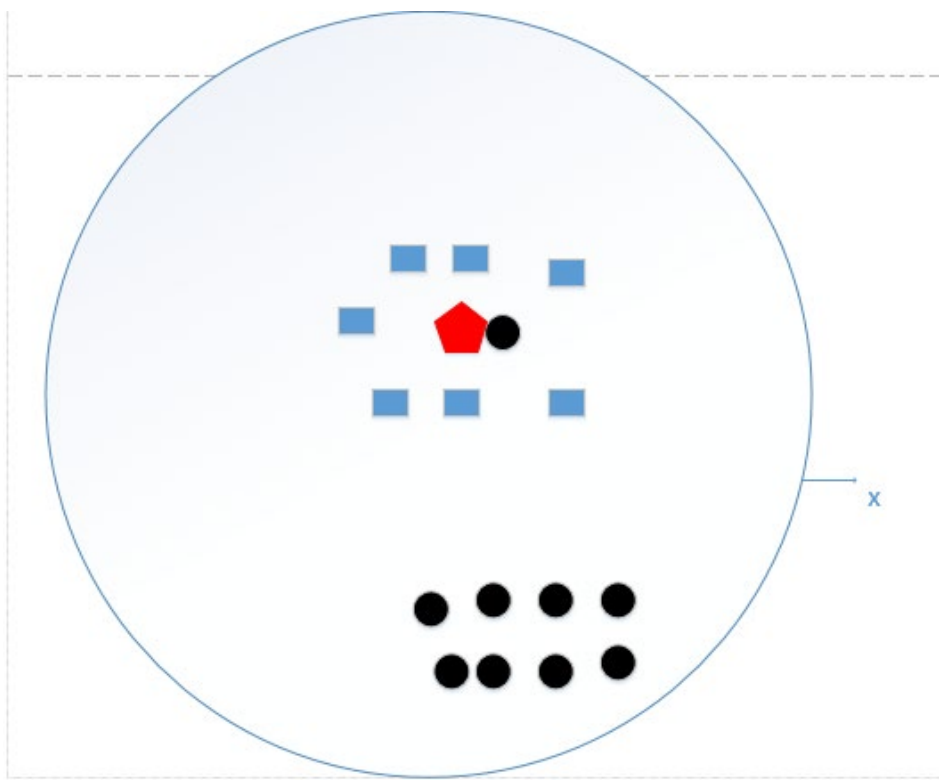


## 六、K近邻算法模型：k值选择

### 5-nearest neighbor



## 六、K近邻算法模型：k值的极端情况



$k = N$ 的情况

- 如果  $k = N$  ( $N$  为训练样本的个数), 那么无论输入实例是什么, 都将简单地预测它属于在训练实例中最多的类。这个时候, 模型过于简单, 完全忽略训练数据实例中的大量有用信息, 是不可取的。相当于直接拿训练数据统计了一下各个数据的类别, 找最大的而已!
- 如果  $k = 1$ , 那么就是最近邻算法, 受噪声影响比较大。

## 六、K近邻算法模型：选择k值

- 那么我们一般怎么选取K值呢？
- 我们一般选取一个较小的数值，通常采取交叉验证法来选取最优的k值。
- 也就是说，选取k值很重要的关键是实验调参，类似于神经网络选取多少层这种，通过调整超参数来得到一个较好的结果。





## 六、K近邻算法模型：分类决策规则

- 分类决策规则
- k近邻法中的分类决策规则往往是多数表决，即由输入实例的k个邻近的训练实例中的多数类决定输入实例的类。
- 多数表决规则( majority voting rule)有如下解释：
- 如果分类的损失函数为0-1损失函数，分类函数为

$$f: \mathbf{R}^n \rightarrow \{C_1, C_2, \dots, C_K\}$$

- 那么误分类的概率是
  - $P(Y \neq f(X)) = 1 - P(Y = f(X))$

## 六、K近邻算法模型：分类决策规则

- 对给定的实例  $x \in X$ ,
- 其最近邻的  $k$  个训练实例点构成集合  $N_k(x)$
- 如果涵盖  $N_k(x)$  的区域的类别是  $C_j$ , 那么误分类率是:

$$\frac{1}{k} \sum_{x_i \in N_k(x)} I(y_i \neq C_j) = 1 - \frac{1}{k} \sum_{x_i \in N_k(x)} I(y_i = C_j)$$

- 要使误分类率最小即经验风险最小, 就要使  $\sum_{x_i \in N_k(x)} I(y_i = C_j)$  最大,
- $k$ 近邻法中的分类决策规则往往是多数表决(投票), 即由输入实例的  $k$  个近邻的训练实例中的多数类决定输入实例的类别, 多数表决规则等价于经验风险最小化。

## 六、K近邻算法模型：优缺点

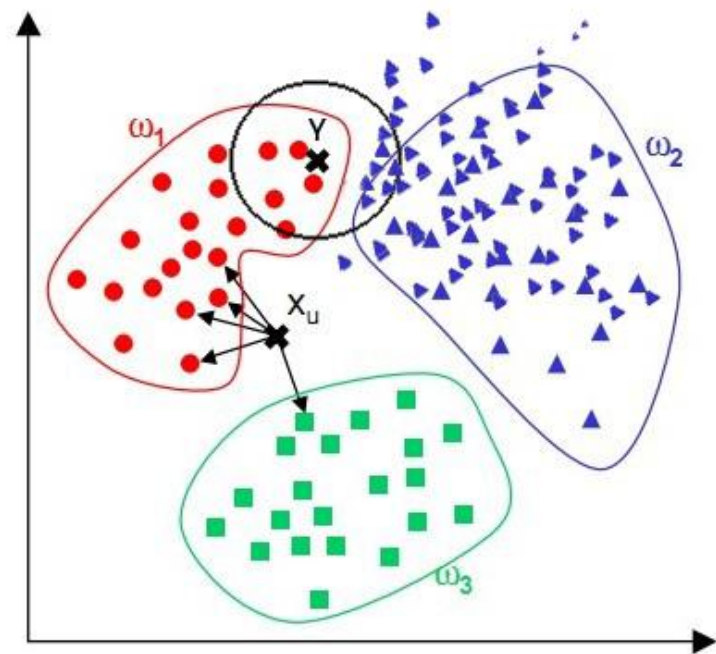
### K近邻模型的优点：

- 简单好用，容易理解，精度高
- 理论成熟，既可以用来做分类也可以用来做回归；
- 可用于数值型数据和离散型数据；
- 对异常值不敏感。

## 六、K近邻算法模型；优缺点

### K近邻模型的缺点：

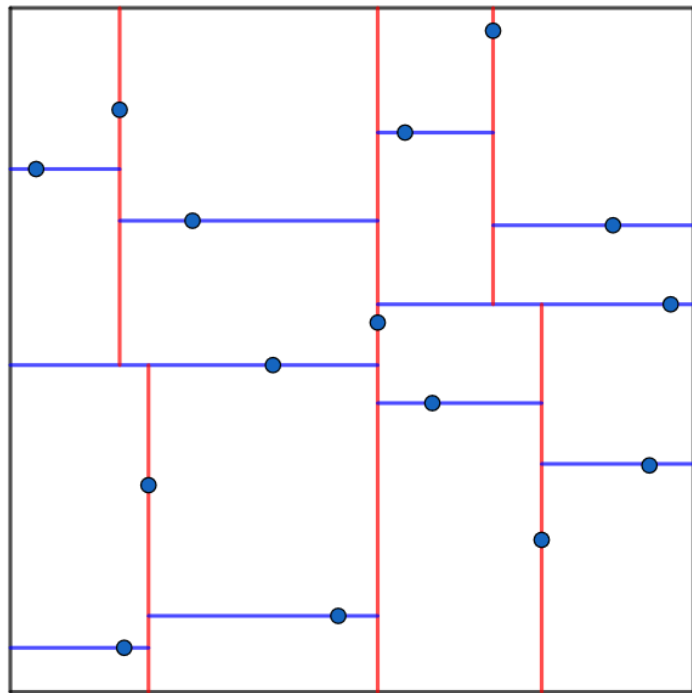
- 计算复杂性高；空间复杂性高；
- **样本不平衡问题**（即有些类别的样本数量很多，而其它样本的数量很少）；
- 一般数值很大的时候不用这个，计算量太大。但是单个样本又不能太少，否则容易发生误分。
- 最大的缺点是无法给出数据的内在含义。
- 补充一点：由于它属于懒惰学习，因此需要大量的空间来存储训练实例，在预测时它还需要与已知所有实例进行比较，增大了计算量。



样本不平衡问题示例（Y点）

## 七、k近邻算法的实现：kd树原理的讲解

- KNN计算量优化：KD树
- kd树是一种对k维空间中的实例点进行存储以便对其进行快速检索的**树形数据结构**
- kd树是二叉树，表示对**k维空间**的一个**划分**(partition)。
- 构造kd树相当于不断地用垂直于坐标轴的超平面将k维空间切分，构成一系列的k维超矩形区域。kd树的每个结点对应于一个k维超矩形区域。
- 利用kd树可以省去对大部分数据点的搜索，从而减少搜索的计算量。



## 七、k近邻算法的实现：kd树原理的讲解

- 构造树的方法如下：
- 构造**根结点**，使根结点对应于k维空间中包含所有实例点的超矩形区域；通过下面的**递归**方法，不断地对k维空间进行切分，**生成子结点**。
- 在**超矩形区域(结点)**上选择一个坐标轴和在此坐标轴上的一个**切分点**，确定一个超平面，这个超平面通过选定的切分点并**垂直于**选定的坐标轴，将当前超矩形区域切分为左右两个子区域(子结点)；这时，实例被分到两个子区域。这个过程**直到子区域内没有实例时终止**(终止时的结点为叶结点)。
- 在此过程中，将实例保存在相应的结点上。通常，依次选择坐标轴对空间切分，选择训练实例点在选定坐标轴上的**中位数**(median)为**切分点**，这样得到的kd树是平衡的。注意，**平衡的kd树搜索时的效率未必是最优的**。

## 七、k近邻算法的实现：kd树原理的讲解

- 算法：（构造平衡kd树）
- 输入：k维空间数据集  $T = \{x_1, x_2, \dots, x_N\}$ ,

其中  $x_i = \left(x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(k)}\right)^T, i = 1, 2, \dots, N$

- 输出：kd 树

## 七、k近邻算法的实现：kd树原理的讲解

(1) 开始：

- **构造根结点**，根结点对应于包含 $T$ 的 $k$ 维空间的超矩形区域。
- 选择 $x^{(1)}$ 为坐标轴，以 $T$ 中所有实例的 $x^{(1)}$ 叫坐标的**中位数**为**切分点**，将根结点对应的超矩形区域切分为两个子区域，**切分**由通过切分点并与坐标轴 $x^{(1)}$ 垂直的超平面实现。
- 由根结点生成深度为1的**左、右子结点**：左子结点对应坐标 $x^{(1)}$ 小于切分点的子区域，右子结点对应于坐标 $x^{(1)}$ 大于切分点的子区域。
- 将落在切分超平面上的实例点保存在根结点。



## 七、k近邻算法的实现：kd树原理的讲解

### (2) 重复：

- 对深度为 $j$ 的结点，选择 $x^{(l)}$ 为切分的坐标轴， $l = j(\bmod k) + 1$ ，以该结点的区域中所有实例的 $x^{(l)}$ 坐标的中位数为切分点，将该结点对应的超矩形区域切分为两个子区域。切分由通过切分点并与坐标轴 $x^{(l)}$ 垂直的超平面实现。
- 由该结点生成深度为 $j+1$ 的左、右子结点：左子结点对应坐标 $x^{(l)}$ 小于切分点的子区域，右子结点对应坐标 $x^{(l)}$ 大于切分点的子区域。
- 将落在切分超平面上的实例点保存在该结点。

(3) 直到两个子区域没有实例存在时停止。从而形成kd树的区域划分。

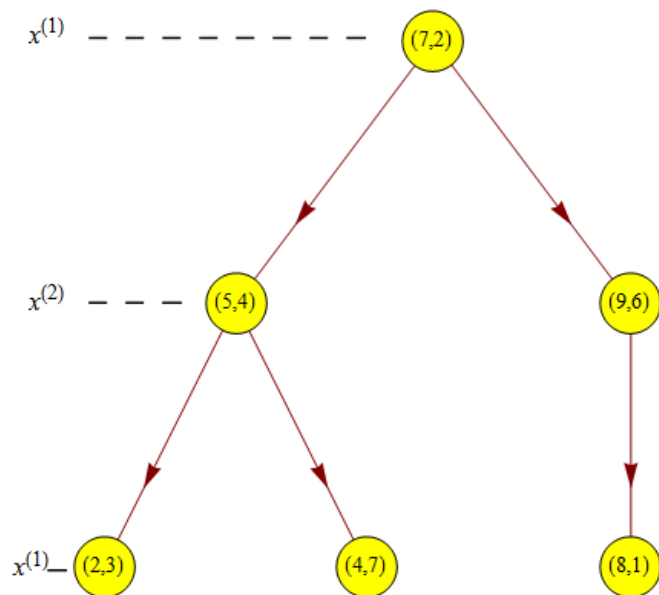
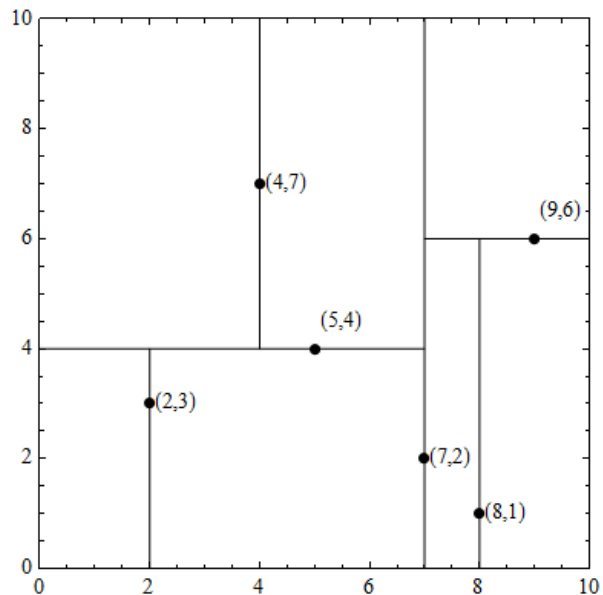
## 七、k近邻算法的实现：kd树原理的讲解

例子：给定一个二维空间数据集：

$$T = \{(2,3), (5,4), (9,6), (4,7), (8,1), (7,2)\},$$

构造一个平衡kd树？

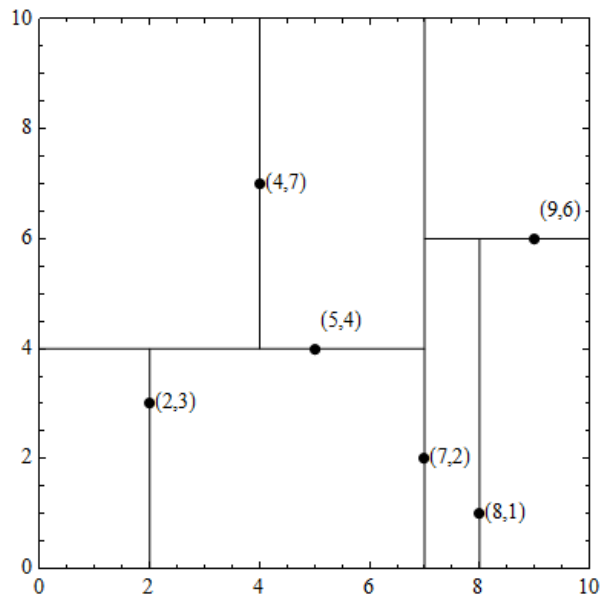
## 七、k近邻算法的实现：kd树原理的讲解



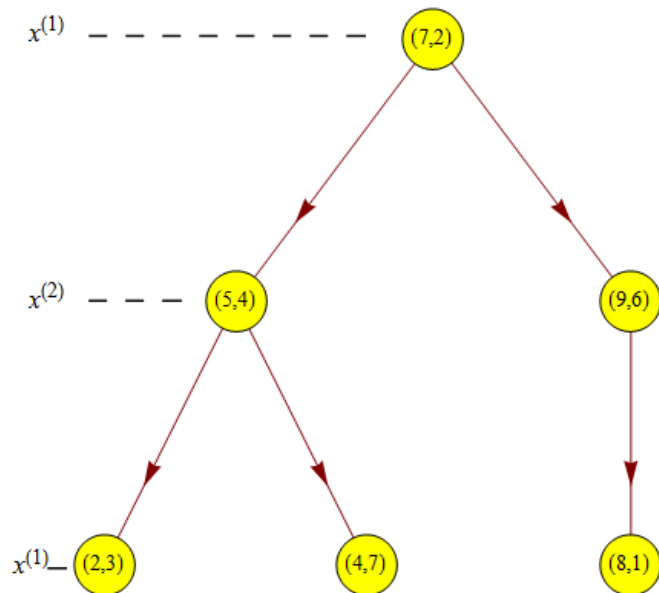
**第一步：** 6个数据点的 $x^{(1)}$ 坐标中位数是6，这里选最接近的(7, 2)点，将空间分为左、右两个子矩形（子结点）

$$T = \{(2,3), (5,4), (9,6), (4,7), (8,1), (7,2)\}$$

## 七、k近邻算法的实现：kd树原理的讲解



$$T = \{(2,3), (5,4), (9,6), (4,7), (8,1), (7,2)\}$$



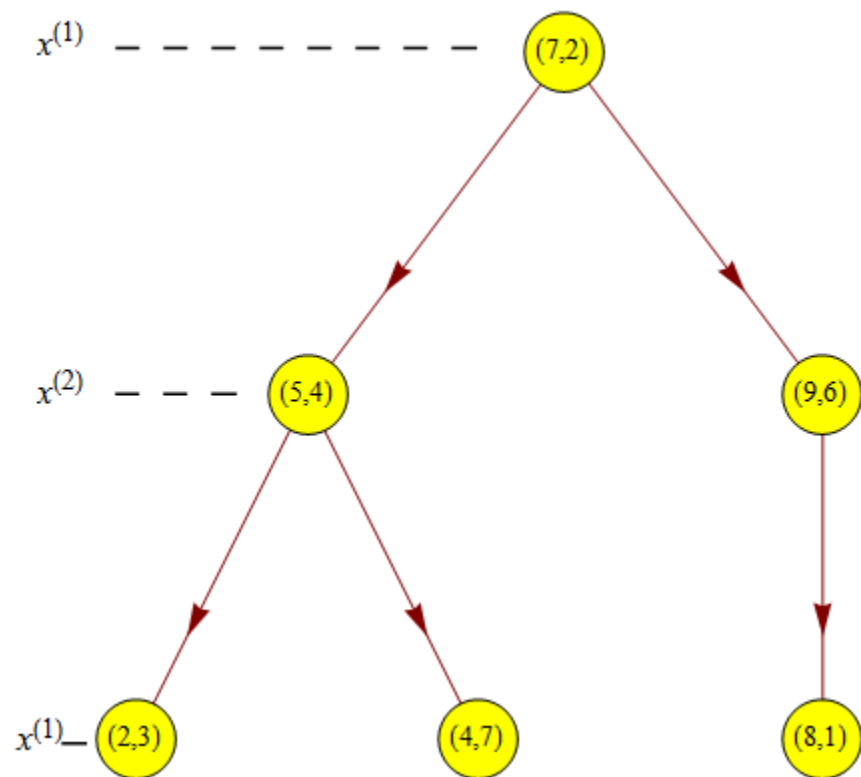
第二步：

左矩形以 $x^{(2)} = 4$ 分为两个子矩形  
(左矩形中 $\{(2,3), (5,4), (4,7)\}$   
点的 $x^{(2)}$ 坐标中位数正好为4，右矩  
形以 $x^{(2)} = 6$ ，6离 $(6+1)/2$ 近一  
点，分为两个子矩形。

如此递归，最后得到如图所示的特  
征空间划分和kd树。

## 七、k近邻算法的实现：kd树原理的讲解

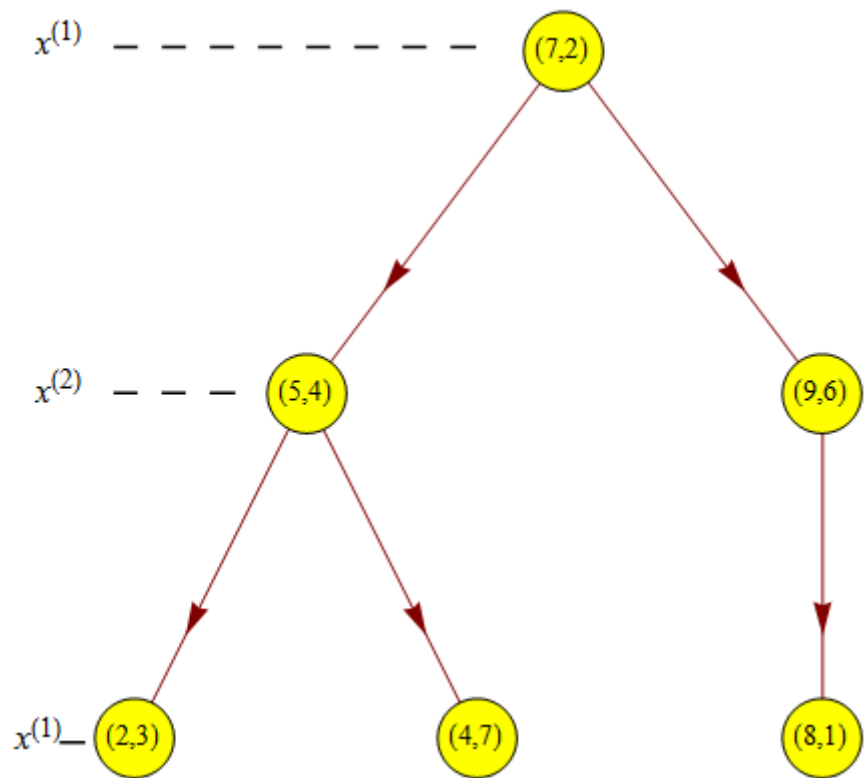
### • 搜索kd树



- 我们可以看到，利用kd树可以省去对大部分数据点的搜索，从而减少搜索的计算量。
- 这里以最近邻为例加以叙述，同样的方法可以应用到k近邻。
- 给定一个目标点，搜索其最近邻。首先找到包含目标点的叶结点；然后从该叶结点出发，依次回退到父结点；不断查找与目标点最邻近的结点，当确定不可能存在更近的结点时终止。
- 这样搜索就被限制在空间的局部区域上，效率大为提高。

## 七、k近邻算法的实现：kd树原理的讲解

### • 搜索kd树



- 包含目标点的叶结点对应包含目标点的最小超矩形区域  
以此叶结点的实例点作为当前最近点。
- 目标点的最近邻一定在以目标点为中心并通过当前最近点的超球体的内部。
- 然后返回当前结点的父结点，如果父结点的另一子结点的超矩形区域与超球体相交，那么在相交的区域内寻找与目标点更近的实例点。如果存在这样的点，将此点作为新的当前最近点。算法转到更上一级的父结点，继续上述过程。
- 如果父结点的另一子结点的超矩形区域与超球体不相交，或不存在比当前最近点更近的点，则停止搜索。

## 七、k近邻算法的实现：kd树原理的讲解

### 算法（用kd树的最近邻搜索）

输入：已构造的kd树；目标点 $x$

输出： $x$  的最近邻。

(1) 在kd树中找出包含目标点 $x$ 的叶结点：

从根结点出发，递归地向下访问kd树。若目标点 $x$ 当前维的坐标小于切分点的坐标，则移动到左子结点，否则移动到右子结点。直到子结点为叶结点为止。

(2) 以此叶结点为“当前最近点”

(3) 递归地向上回退，在每个结点进行以下操作：

## 七、k近邻算法的实现：kd树原理的讲解

### 算法（用kd树的最近邻搜索）

**输入：**已构造的kd树；目标点 $x$

**输出：** $x$  的最近邻。

- (a) 如果该结点保存的实例点比当前最近点距离目标点更近，则以该实例点为“当前最近点”。
- (b) 当前最近点一定存在于该结点一个子结点对应的区域。检查该子结点的父结点的另一子结点对应的区域是否有更近的点。
  - 具体地，检查另一子结点对应的区域是否与以目标点为球心、以目标点与“当前最近点”间的距离为半径的超球体相交。
  - 如果相交，可能在另一个子结点对应的区域内存在距目标点更近的点，移动到另一个子结点。接着，递归地进行最近邻搜索；如果不相交，向上回退。



## 七、k近邻算法的实现：kd树原理的讲解

### 算法（用kd树的最近邻搜索）

输入：已构造的kd树；目标点 $x$

输出： $x$  的最近邻。

(4) 当回退到根结点时，搜索结束。最后的“当前最近点”即为 $x$ 的最近邻点。

如果实例点是随机分布的，kd树搜索的平均计算复杂度是 $O(\log N)$ ，这里 $N$ 是训练实例数。

- kd树更适用于训练实例数远大于空间维数时的k近邻搜索。
- 当空间维数接近训练实例数时，它的效率会迅速下降，几乎接近线性扫描。

## 七、k近邻算法的实现：kd树原理的讲解

例题：给定一个如图所示的kd树,根结点为A,其子结点为B,C等。树上共存储7个实例点；另有一个输入目标实例点S，求S的最近邻。

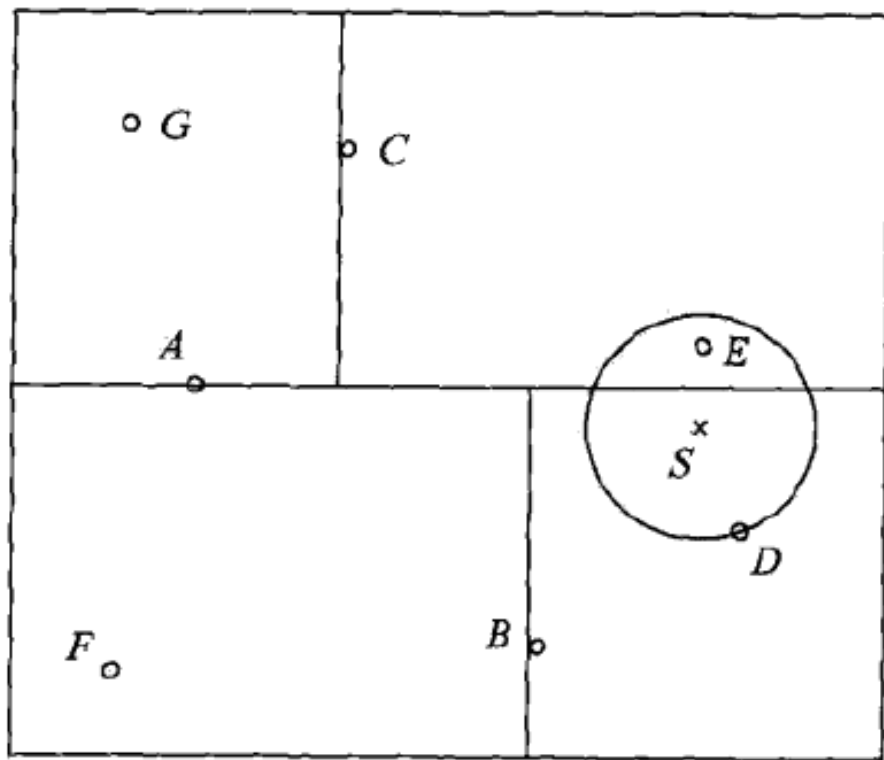


图 3.5 通过  $kd$  树搜索最近邻

## 七、k近邻算法的实现：kd树原理的讲解

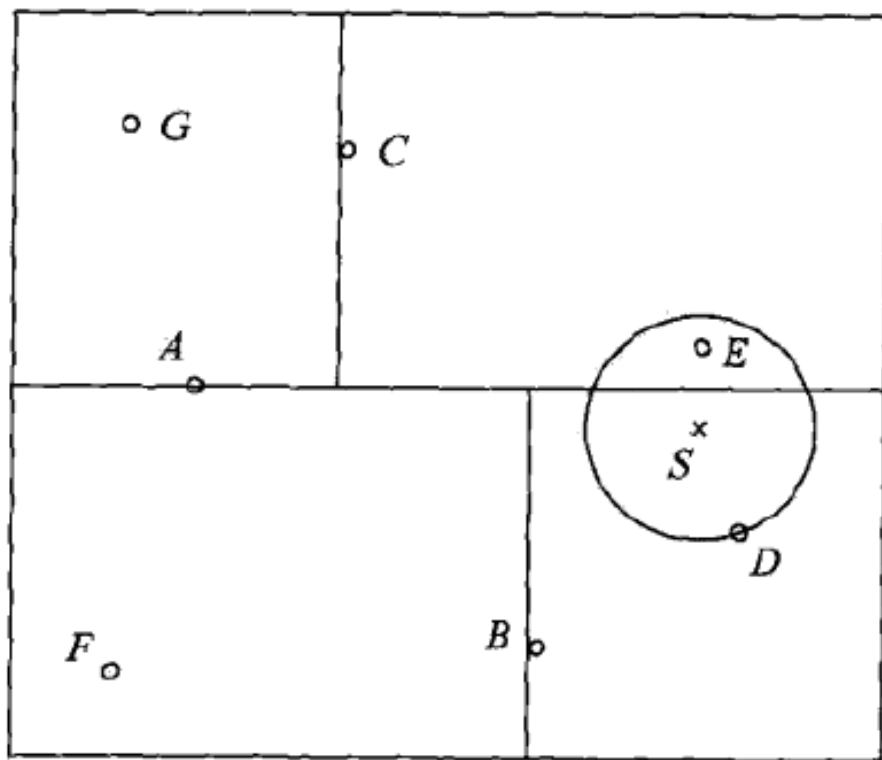


图 3.5 通过  $kd$  树搜索最近邻

- 首先在kd树中找到包含点S的叶结点D(图中的右下区域)，以点D作为近似最近邻。
- 真正最近邻一定在以点S为中心通过点D的圆的内部。然后返回结点D的父结点B，在结点B的另一子结点F的区域内搜索最近邻。结点F的区域与圆不相交，不可能有最近邻点。
- 继续返回上一级父结点A，在结点A的另一子结点C的区域内搜索最近邻。结点C的区域与圆相交；该区域在圆内的实例点有点E，点E比点D更近，成为新的最近邻近似。
- 最后得到点E是点S的最近邻。

## 八、k近邻算法的拓展

- **K近邻算法与原型网络：**

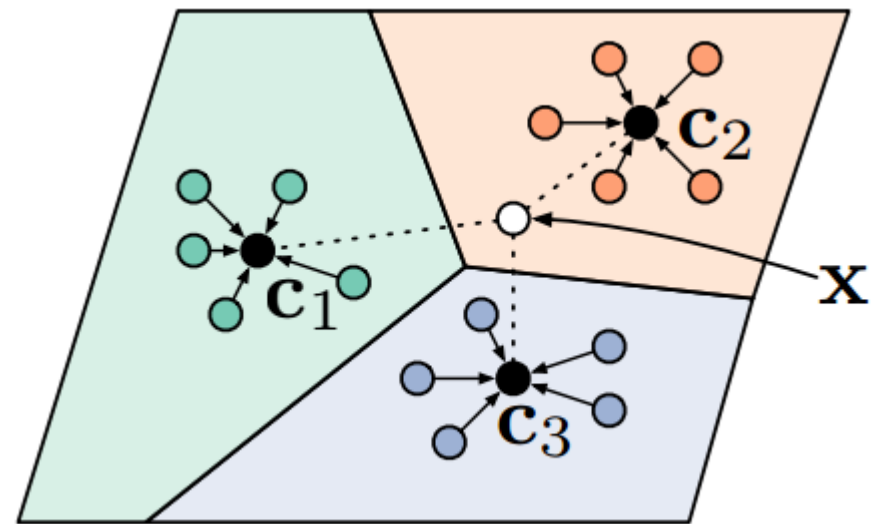
- 还有一种扩展，**最近质心算法**。

(1) 它首先把样本按K近邻算法的输出进行归类。

(2) 对于第 $i$ 类的 $c_i$ 个样本，它会对这 $c_i$ 个样本的 $n$ 维特征中每一维特征求平均值，最终该类别所有维度的 $n$ 个平均值形成所谓的**质心点**。对于样本中的所有出现的类别，每个类别会最终得到一个质心点。

(3) 当我们做**预测**时，仅仅需要比较预测样本和这些质心的距离，**最小的距离对应的质心类别即为预测的类别**。

- 这种方法在小样本学习的应用中叫做**原型网络**。



Prototypical Networks for Few-shot Learning

## 八、k近邻算法的拓展

### 作业

- (必做) 在线性空间中, 证明一个点 $\hat{x}$ 到平面 $f(x; w, b) = w^T x + b = 0$ 的距离为 $|f(\hat{x}; w, b)| / \|w\|$ ;

- (选做) 证明Novikoff定理:

设训练数据集 $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ 是线性可分的, 其中 $x_i \in R^n$ ,  $y_i = \{+1, -1\}$ ,  $i = 1, 2, \dots, N$ , 则

(1) 存在满足条件 $\|\hat{w}_{opt}\| = 1$ 的超平面 $\hat{w}_{opt} \cdot \hat{x} = w_{opt} \cdot x + b_{opt} = 0$ 将训练数据集完全正确分开; 且存在 $\gamma > 0$ , 对所有 $i = 1, 2, \dots, N$

$$y_i(\hat{w}_{opt} \cdot \hat{x}) = y_i(w_{opt} \cdot x + b_{opt}) \geq \gamma$$

(2) 令 $R = \max_{1 \leq i \leq N} \|\hat{x}_i\|$ , 则感知机算法 (P20) 在训练数据集上的误分类次数 $k$ 满足不等式

$$k \leq (R/\gamma)^2$$

# 谢谢！

李爽

E-mail: [shuangli@bit.edu.cn](mailto:shuangli@bit.edu.cn)

Homepage: [shuangli.xyz](http://shuangli.xyz)