机器学习（本科生公选课）GEC6531

第6节 感知机 Perceptron

计算机科学与技术学院

张瑞 教授

邮箱：ruizhang6@hust.edu.cn

签到 & 思考

■ **微助教签到（学校要求）**

1. 加入课堂：微信扫码或者通过微助教公众号

2. 微信扫码签到

回忆线性回归和逻辑回归公式

今天的目录

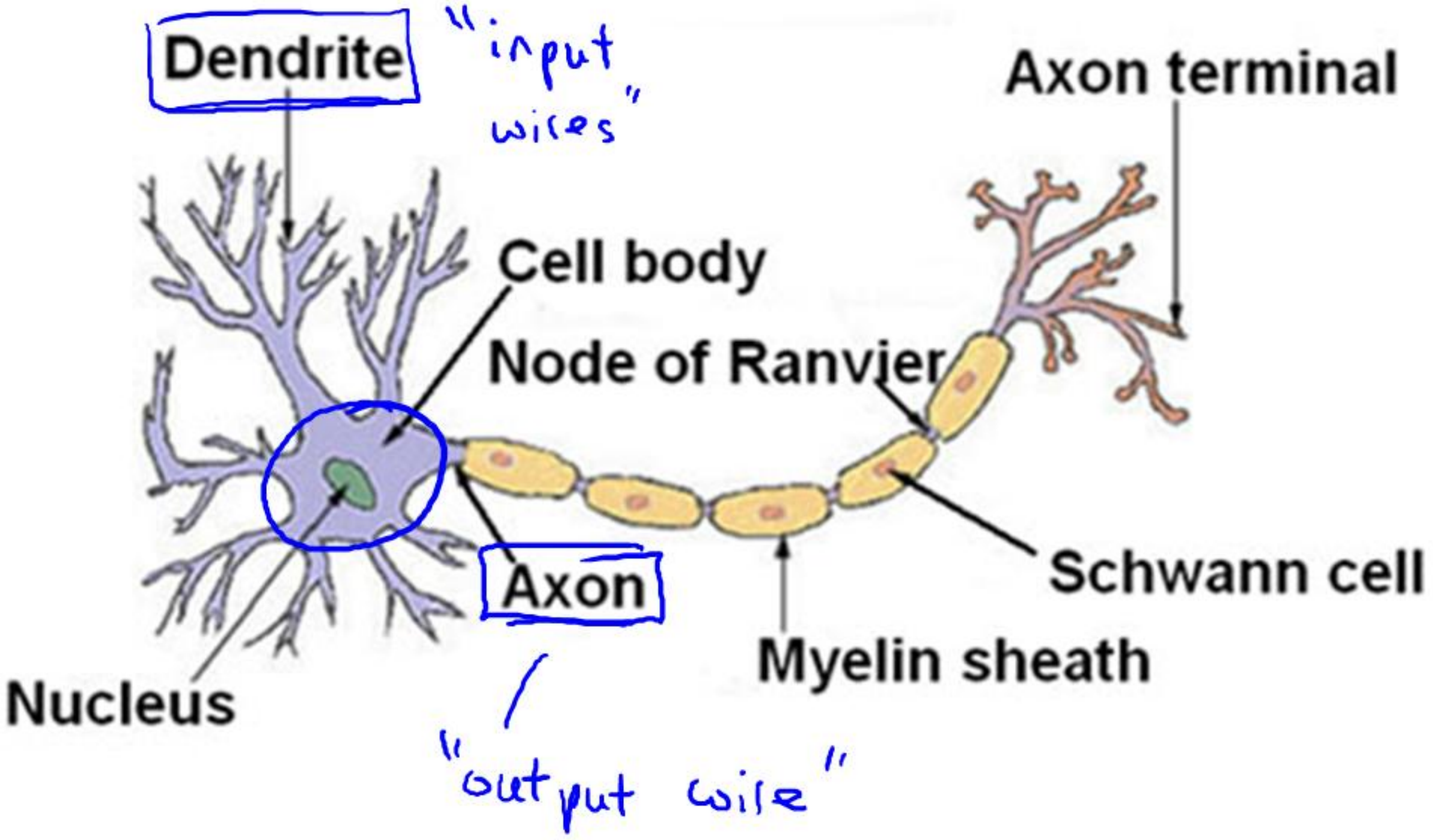
- **感知机**
  - 与脑神经类比
  - 形式化定义
  - 限制和假设
- **参数选择**
  - 直观理解
  - 感知机算法
  - 收敛性
- **感知机的历史**
  - 首个感知机
  - 异或问题
  - 从感知机到支持向量机
  - 从感知机到神经网络

今天的目录

- **感知机**
  - 与脑神经类比
  - 形式化定义
  - 限制和假设
- **参数选择**
  - 直观理解
  - 感知机算法
  - 收敛性
- **感知机的历史**
  - 首个感知机
  - 异或问题
  - 从感知机到支持向量机
  - 从感知机到神经网络

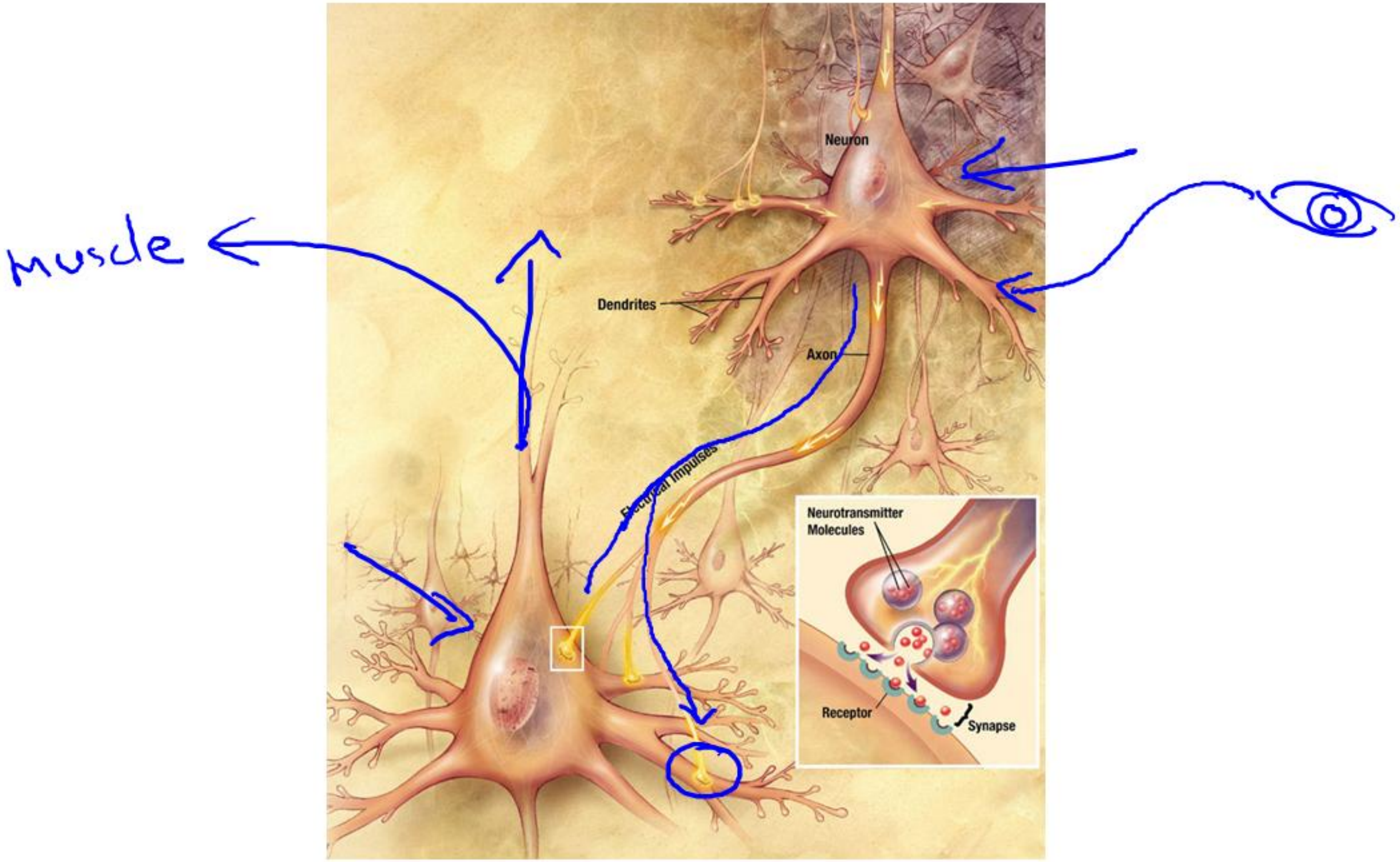
与脑神经类比

Neuron in the brain



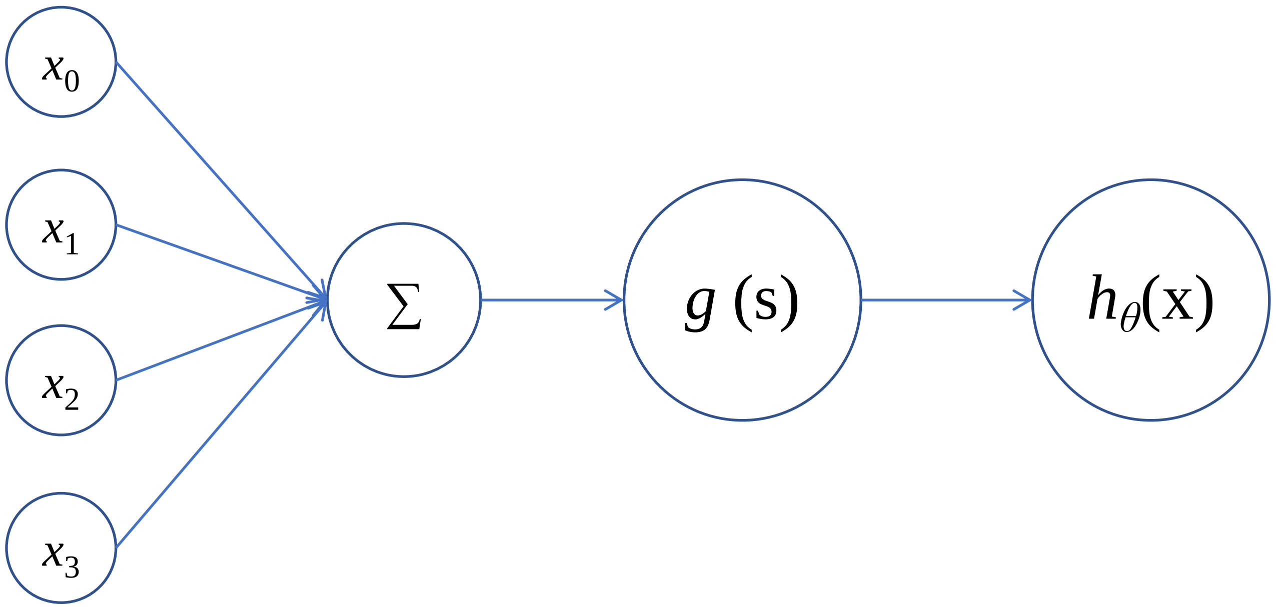
与脑神经类比

Neurons in the brain



[Credit: US National Institutes of Health, National Institute on Aging]

感知机 Perceptron



$$s = \sum_{i=0}^{m} x_i w_i = \mathbf{w}^{\mathrm{T}}\mathbf{x}$$

- x_1 , x_2 , x_3 --- 输入
- w_1, w_2 , w_3 --- 权重
- w_0 --- 偏置
- g(s) --- 激活函数

用最大似然估计（MLE）求参

$$= \underset{\mathbf{w}}{\operatorname{argmax}} -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(\mathbf{x}_i^\top \mathbf{w} - y_i)^2$$

$$= \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i^\top \mathbf{w} - y_i)^2$$

这里的 w 就是前面的 θ

- 我们最小化损失函数，$\ell(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i^\top \mathbf{w} - y_i)^2$。这个特殊的损失函数也被称为均方损失或最小二乘损失 (Ordinary Least Squares)。最小二乘损失可以用梯度下降法、牛顿法或闭式解进行优化。

闭式解形式 $\mathbf{w} = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{y}$，其中 $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ 且 $\mathbf{y} = [y_1, \ldots, y_n]^\top$.

- 在 m 元线性回归中，$y = w_0 + \sum_{i=1}^{m} x_i w_i$ ，为了表示方便，令 x_0=1, 就可以用向量表示为 $y = \sum_{i=0}^{m} x_i w_i = \mathbf{x}^\mathrm{T}\mathbf{w} = \mathbf{w}^\mathrm{T}\mathbf{x}$
- 通过上面向量表示, $d = m + 1$, 在一元线性回归中 $m = 1$, $d = m + 1 = 2$
- 注意：黑体 **x**, **y**, **w** 都是向量（d 维），非黑体是标量，**X** 是矩阵
- $\mathbf{x}_i$ 是 d 维的列向量 (i=1, 2, ... n), n 是训练样本个数, **X** 是一个 $d \times n$ 的矩阵
- $\mathbf{X}\mathbf{X}^\mathrm{T}$ 是 $d \times d$ 矩阵, $(\mathbf{X}\mathbf{X}^\mathrm{T})^{-1}$是$d \times d$ 矩阵, $(\mathbf{X}\mathbf{X}^\mathrm{T})^{-1}\mathbf{X}$是$d \times n$ 矩阵, $(\mathbf{X}\mathbf{X}^\mathrm{T})^{-1}\mathbf{X}\mathbf{y}$是$d \times 1$ 矩阵, 即 d 维列向量

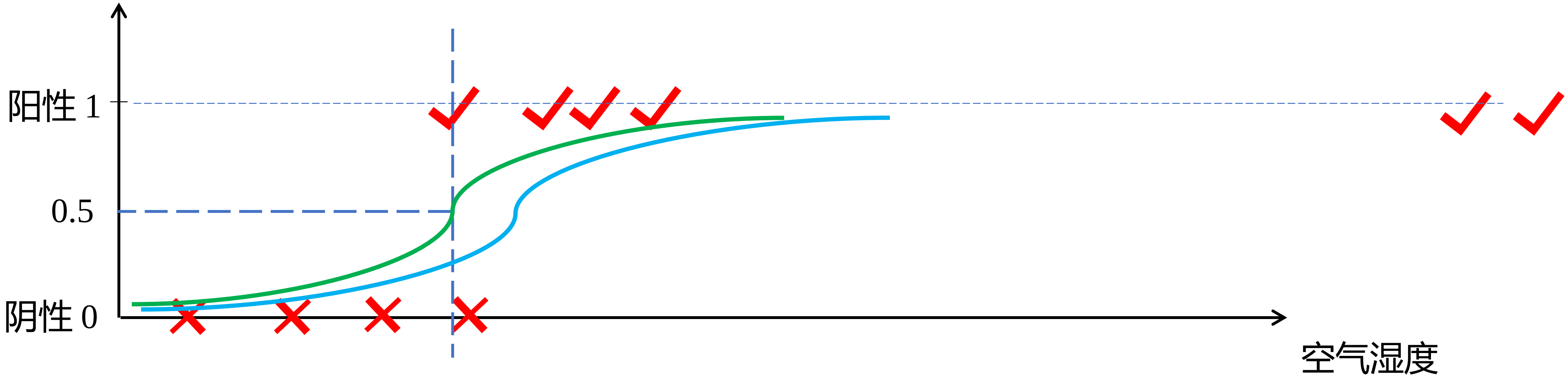
逻辑回归的表示

■ **把逻辑函数里的自变量替换成 $\mathbf{w}^{\mathrm{T}}\mathbf{x}$**

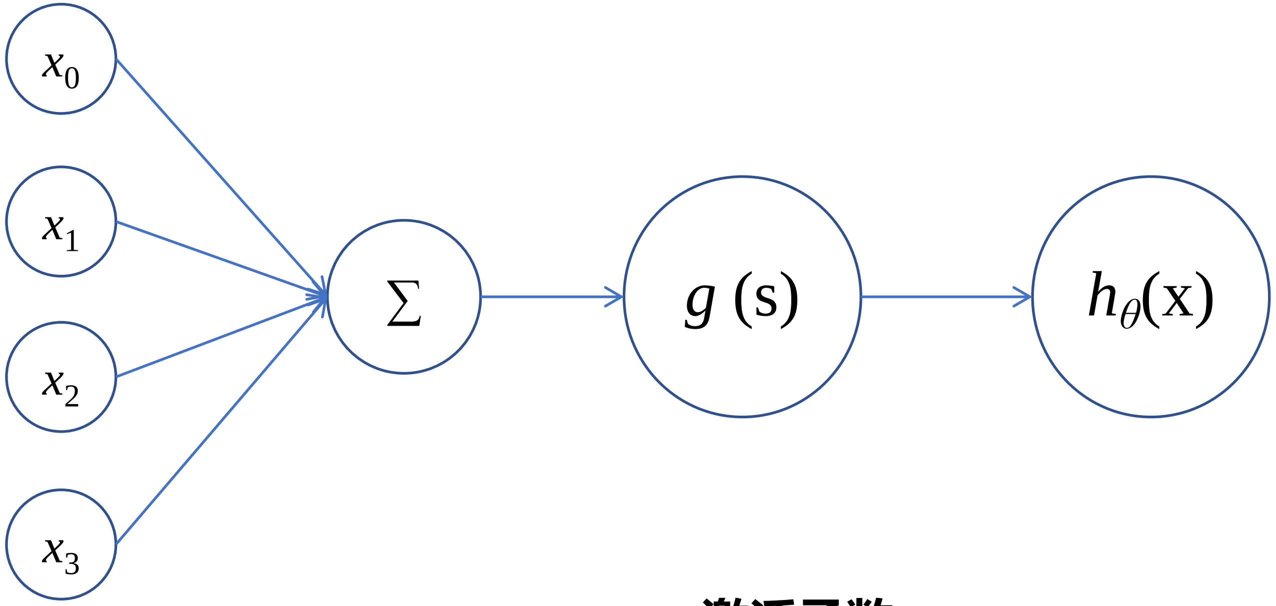
阳性概率

$$g(s) = \frac{1}{1+\exp(-s)} \quad \Rightarrow \quad p(y=1|x,w) = \frac{1}{1+e^{-w^{T}x}}$$

- $h(\mathbf{x}) = g(\mathbf{w}^{\mathrm{T}}\mathbf{x})$



感知机 Perceptron



如果 $\mathrm{s} \geq 0$, $h_\theta(\mathrm{x})$ 是正类

如果 $\mathrm{s} < 0$, $h_\theta(\mathrm{x})$ 是负类

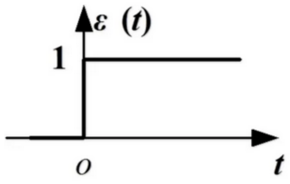
激活函数

$$s = \sum_{i=0}^{m} x_i w_i = \mathbf{w}^{\mathrm{T}}\mathbf{x}$$

Step function

阶跃函数

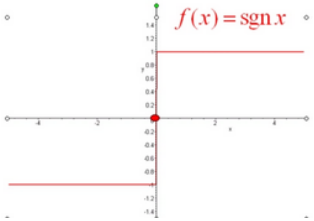
$$g(s) = \begin{cases} 1, & if\ s \geq 0 \\ 0, & if\ s < 0 \end{cases}$$



Sign function

符号函数

$$g(s) = \begin{cases} 1, & if\ s \geq 0 \\ -1, & if\ s < 0 \end{cases}$$



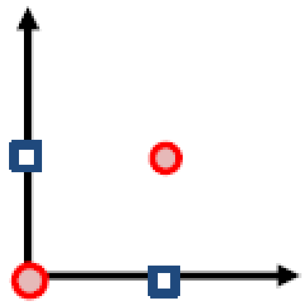
简单的例子

x_1	x_2	y
0	0	Class B
0	1	Class B
1	0	Class B
1	1	Class A

感知机的限制：线性可分

Limitations of perceptron learning

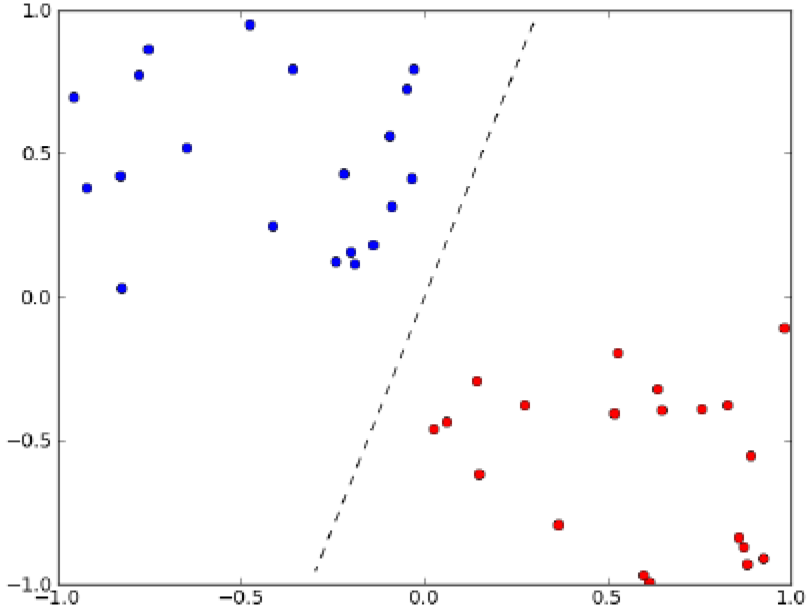
- If the data is linearly separable, the perceptron training algorithm will converge to a correct solution
  - It will converge to some solution (separating boundary), one of infinitely many possible ← bad!
- However, if the data is not linearly separable, the training will fail completely rather than give some approximate solution
  - Ugly ☹



假设

假设

- 二分类 (即 $y_i \in \{-1, +1\}$)
- 数据线性可分



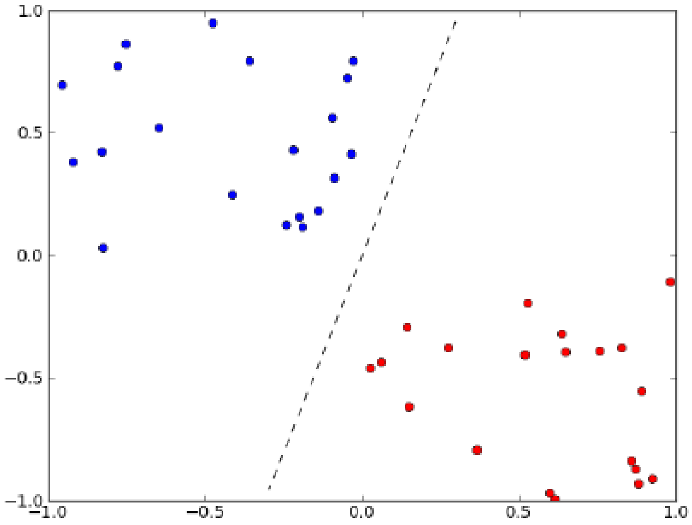
基本思想

基本思想:

- 在机器学习中，感知器是一种用于监督学习的二分类器。
- 二分类器是一个函数，决定由数字向量表示的输入是否属于某个特定的类。
- 它是一种线性分类器，即一种基于一组权重与特征向量相结合的线性预测函数进行预测的分类算法。

$$\text{假设空间：} \mathcal{H} = \{h(x) = \mathrm{w}^\top \mathrm{x} + b = 0\}$$

对应于特征空间中的一个超平面，其中：w 是超平面的法向量，b 是超平面的截距。

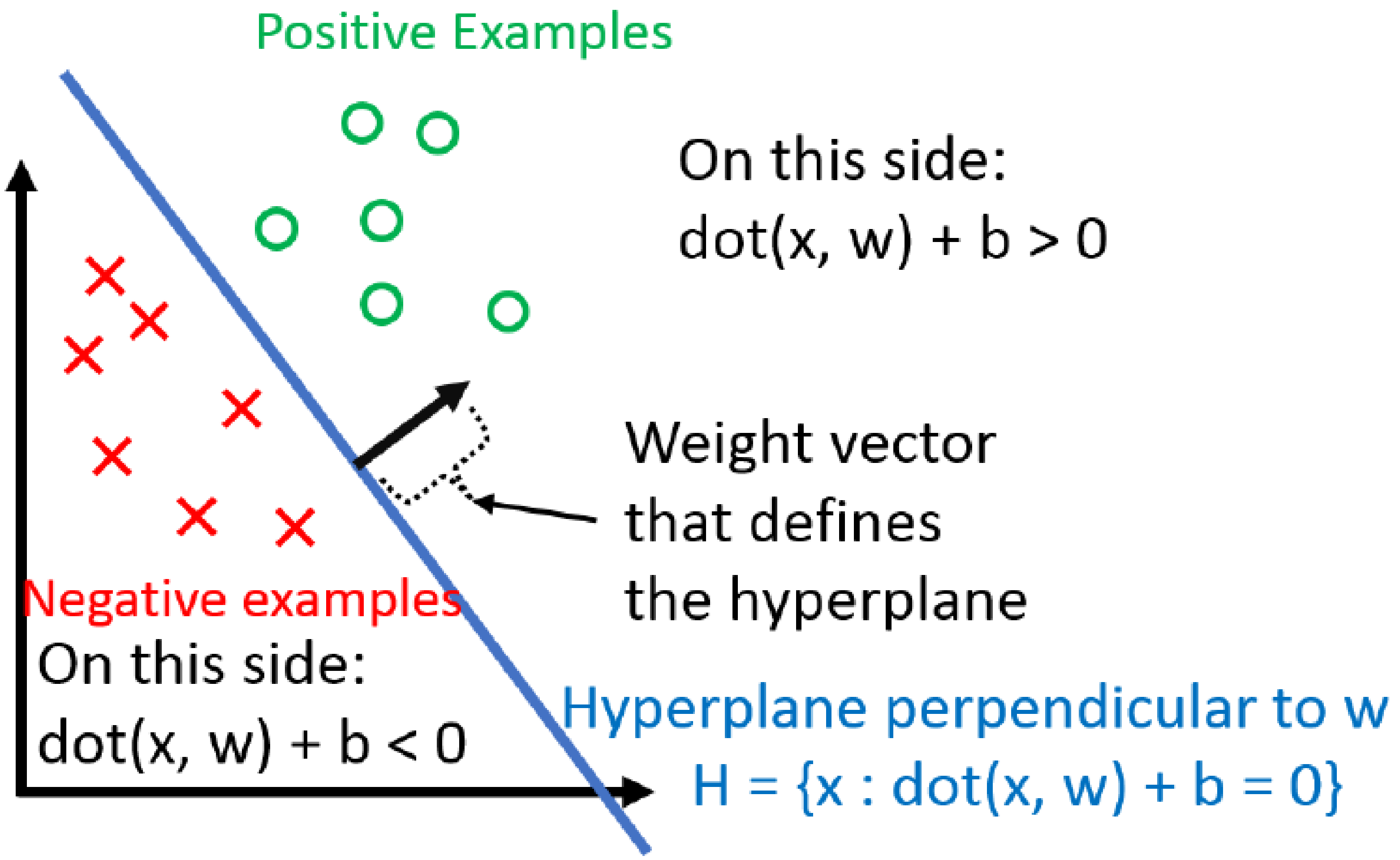


今天的目录

- **感知机**
  - 与脑神经类比
  - 形式化定义
  - 限制和假设
- **参数选择**
  - 直观理解
  - 感知机算法
  - 收敛性
- **感知机的历史**
  - 首个感知机
  - 异或问题
  - 从感知机到支持向量机
  - 从感知机到神经网络

参数选择

$$h(x_i) = \mathrm{sign}(\mathbf{w}^\top \mathbf{x}_i + b)$$



参数选择

b 是偏置项 (如果没有偏置项，$\mathbf{w}$ 定义的超平面将始终经过原点)。

处理 b 可能很麻烦，所以通过添加一个额外的常量维度将它 “吸收” 到特征向量 $\mathbf{w}$ 中。
在该约定下：

$$\mathbf{x}_i \quad 变为 \quad \begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix}$$

$$\mathbf{w} \quad 变为 \quad \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$

可以验证：$\begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix}^\top \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} = \mathbf{w}^\top \mathbf{x}_i + b$

从而得到：

$$\mathcal{H} = \{h(x) = \mathbf{w}^\top \mathbf{x} = 0\}$$

超平面

观察

请注意,

$$y_i(\mathbf{w}^\top \mathbf{x}_i) > 0 \iff \mathbf{x}_i \quad \text{分类正确}$$

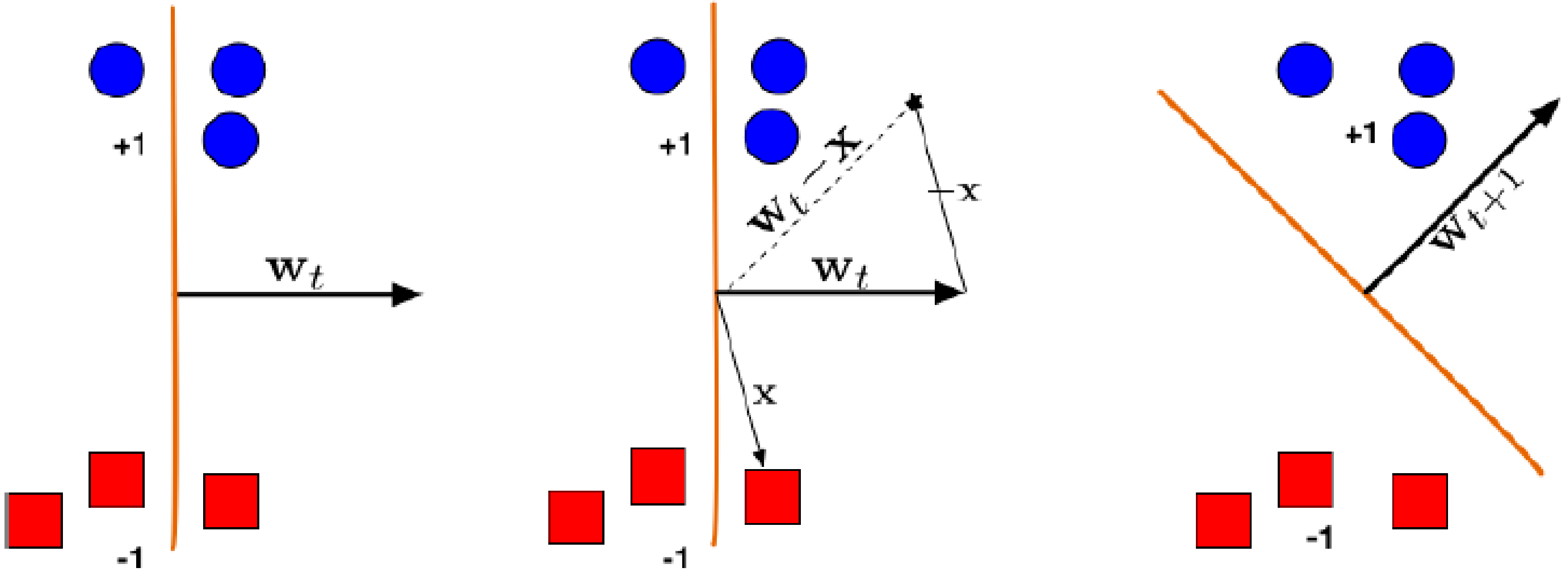
其中“分类正确”意味着 x_i 在由 $\mathbf{w}$ 定义的超平面的正确一侧。
另外，左边依赖于 $y_i \in \{-1, +1\}$（若 $y_i \in \{0, +1\}$，就不起作用了)。

感知机算法

我们知道了 w 应该做什么(定义一个分离数据的超平面)，接下来看看如何获得这样的 w

```
Initialize w⃗ = 0⃗                      // Initialize w⃗. w⃗ = 0⃗ misclassifies everything.
while TRUE do                          // Keep looping
    m = 0                              // Count the number of misclassifications, m
    for (x_i, y_i) ∈ D do              // Loop over each (data, label) pair in the dataset, D
        if y_i(w⃗^T · x⃗_i) ≤ 0 then     // If the pair (x⃗_i, y_i) is misclassified
            w⃗ ← w⃗ + y x⃗                // Update the weight vector w⃗
            m ← m + 1                  // Counter the number of misclassification
        end if
    end for
    if m = 0 then                      // If the most recent w⃗ gave 0 misclassifications
        break                          // Break out of the while-loop
    end if
end while                              // Otherwise, keep looping!
```

几何直觉



感知器更新的示例:

(左:) 由 $\mathbf{w}_t$ 定义的超平面错误地分类了一个红点 (-1) 和一个蓝点 $(+1)$。

(中:) 红点 $\mathbf{x}$ 被选中并用于更新。因为它的标签是 -1，我们需要从 $\mathbf{w}_t$ 中减去 $\mathbf{x}$。

(右:) 已更新的超平面 $\mathbf{w}_{t+1} = \mathbf{w}_t - \mathbf{x}$ 正确分离了两个类，感知机算法已经收敛。

几何直觉

前提：
所在数据点线性可分，即可以找到一个超平面将数据点正确分开。

算法的直观解释：
对所有数据点进行枚举：
当发现一个数据点被当前超平面分类错误，则进行调整：$\mathbf{w} = \mathbf{w} + y_i \mathbf{x}_i$
使分类超平面向该误分类点的一侧移动，以减小该误分类点与超平面间的距离，
直至所有数据点正确分类。

注意：
对分类点的枚举顺序不同，对应的误分类点的顺序不同，可能会得到不同的分类超平面。

感知机的收敛性

感知机是一个具有强收敛性保证的算法。即：如果一个数据集是线性可分的，感知机将在有限次更新中找到一个分离的超平面。
(如果数据不是线性可分的，它将永远循环)

分析如下:

假设 $\exists \mathbf{w}^*$，使得 $\forall (\mathbf{x}_i, y_i) \in D,\ y_i(\mathbf{x}^\top \mathbf{w}^*) > 0$

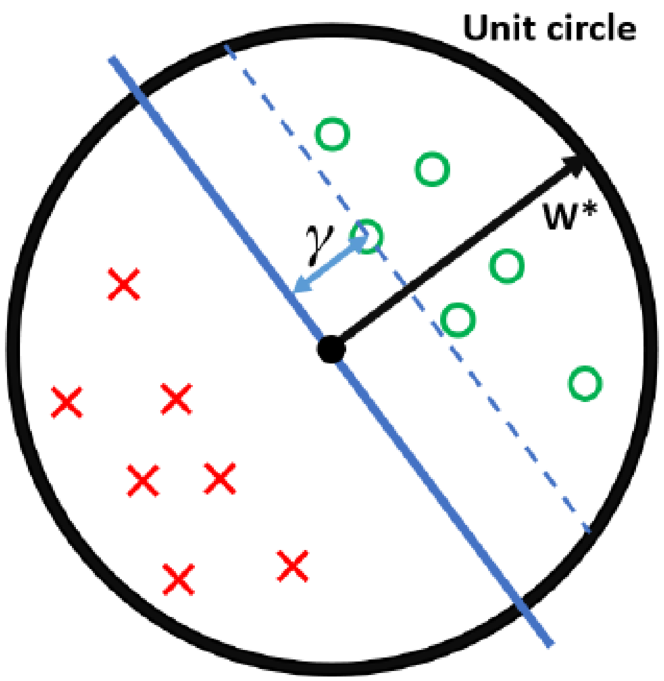
为方便分析，我们重新缩放每个数据点和 $\mathbf{w}^*$，使得

$$||\mathbf{w}^*|| = 1 \qquad \text{且} \qquad \forall \mathbf{x}_i \in D,\ \ ||\mathbf{x}_i|| \leq 1$$

注：通过对每个数据点 $\mathbf{x}_i$ 进行缩放来完成，即均除以：$\alpha = \max_j ||\mathbf{x}_j||$

感知机的收敛性

我们定义超平面 w^* 的Margin γ 为：$\gamma = \min_{(\mathrm{x}_i, y_i)\in D} |\mathrm{x}_i^\top \mathrm{w}^*|$.



总结一下我们的设置:

- 所有输入 x_i 位于单位球内
- 存在一个由 w^* 定义的分离超平面，$\|\mathrm{w}\|^* = 1$ (即 w^* 恰位于单位球上)。
- γ 是这个超平面 (蓝色) 到最近数据点的距离

定理与证明

定理： 若以上假设都成立，则感知机算法最多会出现 $\frac{1}{\gamma^2}$ 次错误。

证明： 基于上述定义，考虑更新 (w 更新为 $\mathrm{w} + y\mathrm{x}$) 对 $\mathrm{w}^\top \mathrm{w}^*$ 和 $\mathrm{w}^\top \mathrm{w}$ 两项的影响。

我们将利用以下两个事实：

- $y(\mathrm{x}^\top \mathrm{w}) \leq 0$：这是因为 x 被 w 错误分类了—否则我们不会进行更新。
- $y(\mathrm{x}^\top \mathrm{w}^*) > 0$：这是因为 w^* 是一个分离超平面，其正确分类了所有的点。

定理与证明

1. 考虑 $\mathbf{w}^\top \mathbf{w}^* \Rightarrow (\mathbf{w}+y\mathbf{x})^\top \mathbf{w}^*$ 的影响:

$$(\mathbf{w}+y\mathbf{x})^\top \mathbf{w}^* = \mathbf{w}^\top \mathbf{w}^* + y(\mathbf{x}^\top \mathbf{w}^*) \geq \mathbf{w}^\top \mathbf{w}^* + \gamma$$

这是因为: 对于 $\mathbf{w}^*$, $\mathbf{w}^*$ 定义的超平面到 $\mathbf{x}$ 的距离必须至少为 γ(即 $y(\mathbf{x}^\top \mathbf{w}^*) = |\mathbf{x}^\top \mathbf{w}^*| \geq \gamma$)。
<u>这意味着对于每一次更新</u>, $\mathbf{w}^\top \mathbf{w}^*$ <u>**至少增加** γ.</u>

2. 考虑 $\mathbf{w}^\top \mathbf{w} \Rightarrow (\mathbf{w}+y\mathbf{x})^\top (\mathbf{w}+y\mathbf{x})$ 的影响:

$$(\mathbf{w}+y\mathbf{x})^\top (\mathbf{w}+y\mathbf{x}) = \mathbf{w}^\top \mathbf{w} + \underbrace{2y(\mathbf{w}^\top \mathbf{x})}_{<0} + \underbrace{y^2(\mathbf{x}^\top \mathbf{x})}_{0\leq \quad \leq 1} \leq \mathbf{w}^\top \mathbf{w} + 1$$

该不等式来自如下分析:

- $2y(\mathbf{w}^\top \mathbf{x}) < 0$: 当我们进行了一次更新之后, 意味着 $\mathbf{x}$ 被错误分类了
- $0 \leq y^2(\mathbf{x}^\top \mathbf{x}) \leq 1$, 因为 $y^2 = 1$ 且都有 $\mathbf{x}^\top \mathbf{x} \leq 1$ (because $\|\mathbf{x}\| \leq 1$).

<u>这意味着对于每一次更新</u>, $\mathbf{w}^\top \mathbf{w}$ <u>的增长幅度**至多为 1**.</u>

定理与证明

3. 现在我们可以把上面的推导放在一起。假设我们做了 M 次更新：

$$
\begin{aligned}
M\gamma &\le \mathbf{w}^\top \mathbf{w}^* && \text{By first point} && (1)\\
&= |\mathbf{w}^\top \mathbf{w}^*| && \text{Simply because } M\gamma \ge 0 && (2)\\
&\le ||\mathbf{w}||\ ||\mathbf{w}^*|| && \text{By Cauchy-Schwartz inequality}^* && (3)\\
&= ||\mathbf{w}|| && \text{As } ||\mathbf{w}^*|| = 1 && (4)\\
&= \sqrt{\mathbf{w}^\top \mathbf{w}} && \text{by definition of } ||\mathbf{w}|| && (5)\\
&\le \sqrt{M} && \text{By second point} && (6)\\
& && && (7)\\
&\Rightarrow M\gamma \le \sqrt{M} && && (8)\\
&\Rightarrow M^2\gamma^2 \le M && && (9)\\
&\Rightarrow M \le \frac{1}{\gamma^2} && && (10)
\end{aligned}
$$

因此，更新的总次数 M 限界于一个常数。

* 替代解释:$|\mathbf{w}^\top \mathbf{w}^*| = ||\mathbf{w}||||\mathbf{w}^*||| \cos(\alpha)|$, but $|\cos(\alpha)| \le 1$

定理与证明

提问

基于上述定理，

1） 关于分类器的边界距离，边界距离大还是小更理想?

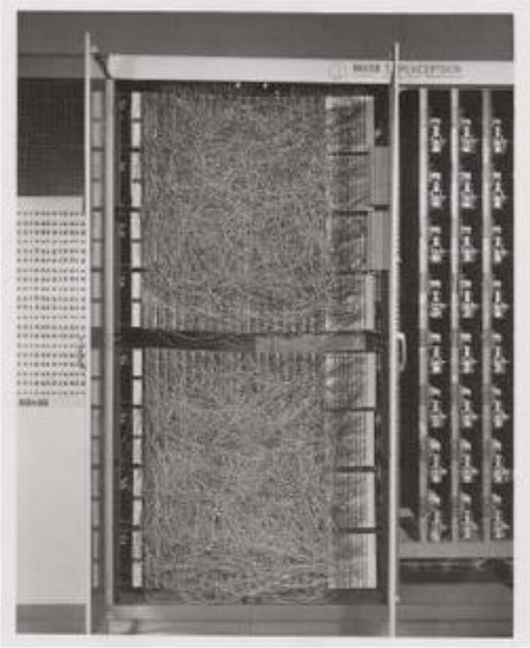
2） 感知器算法快速收敛的数据集具有什么特征? 请试举一例。

今天的目录

- **感知机**
  - 与脑神经类比
  - 形式化定义
  - 限制和假设
- **参数选择**
  - 直观理解
  - 感知机算法
  - 收敛性
- **感知机的历史**
  - 首个感知机
  - 异或问题
  - 从感知机到支持向量机
  - 从感知机到神经网络

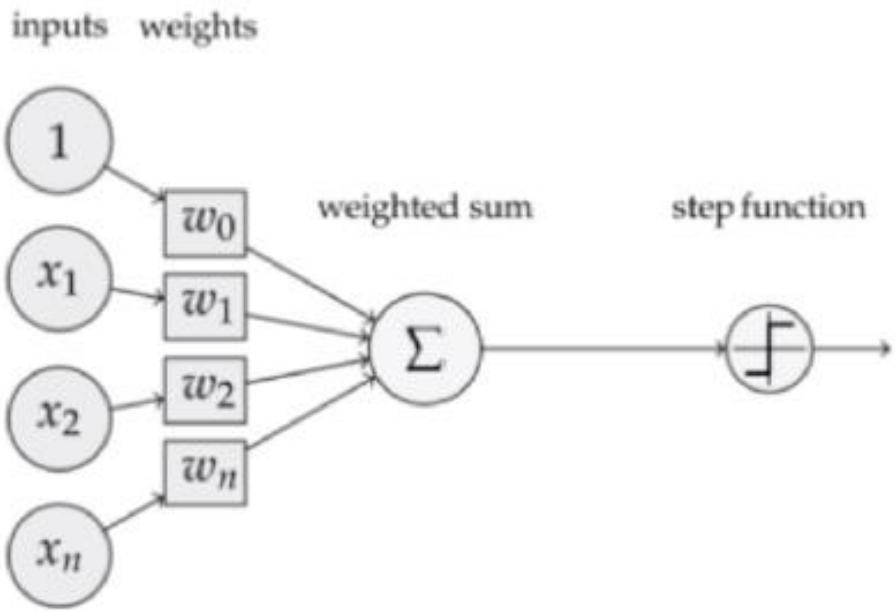
感知机的历史

- 感知机是 1957 年由康奈尔大学航空实验室的 Frank Rosenblatt 发明的。
- Mark I 感知机，是首个感知机算法的实现。
  它连接到一个带有 20×20 硫化镉光电池的相机，可以拍摄 400 像素的图像。主要可见的特征是一个配线架，用于设置输入特征的不同组合。右边是实现自适应权重的电位器阵列。



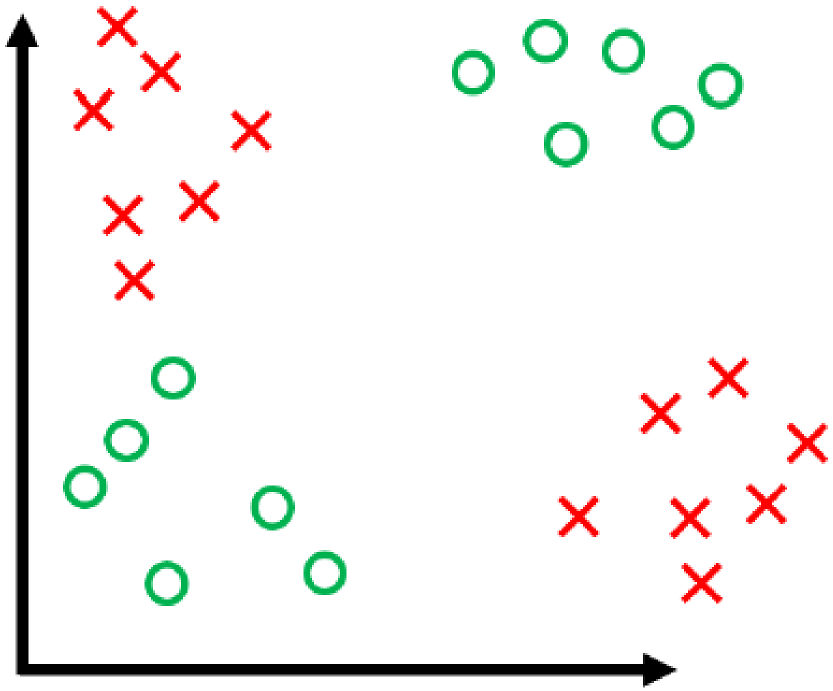
以神经元的方式理解感知机

“Mark 1 感知机” 是为图像识别设计的机器：它有 400 个光电管阵列，随机连接到 “神经元”。权重被编码在电位器中，学习过程中的权重更新由电动机执行。



感知机的历史

- 起初，引起了巨大的轰动（“数字大脑”）(见 1958 年 12 月的《纽约客》)
- 然后，终结于简单非线性可分离数据集的著名例子，异或问题 (Minsky 1969)。导致了人工智能的冬天。



AND, OR, NOT, XOR

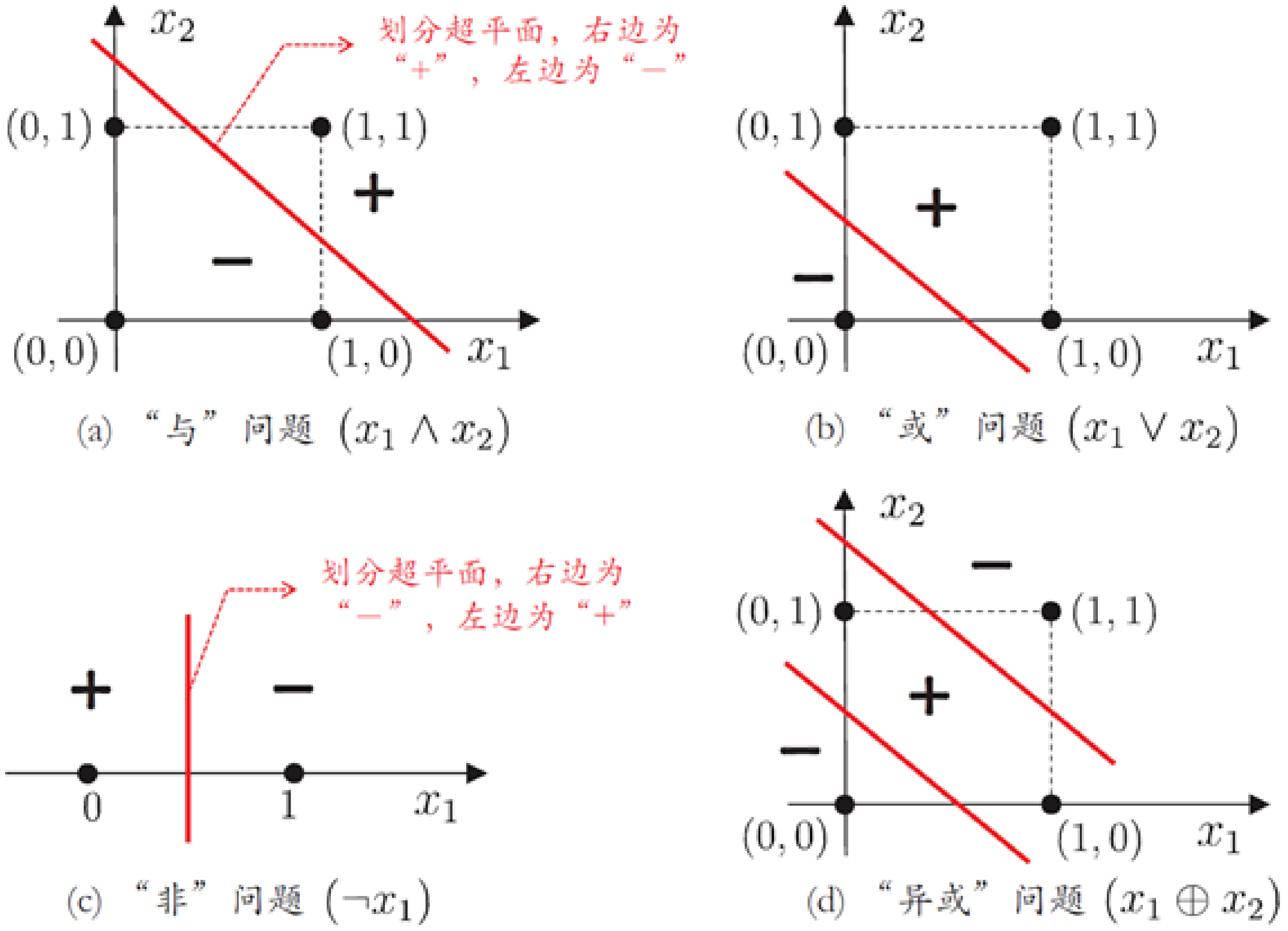


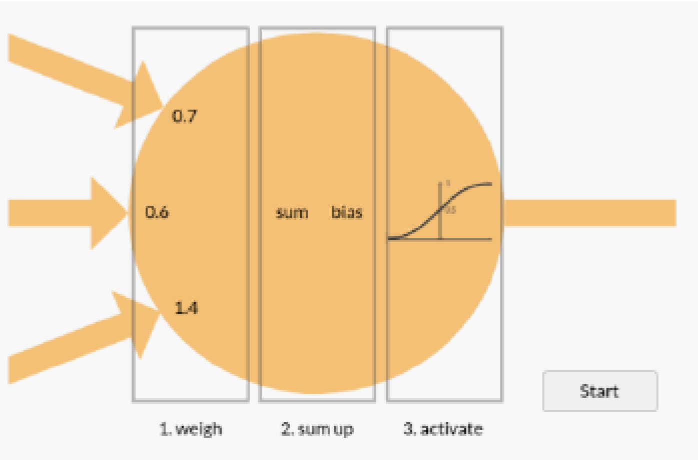
图 5.4 线性可分的“与”“或”“非”问题与非线性可分的“异或”问题

从感知机到支持向量机

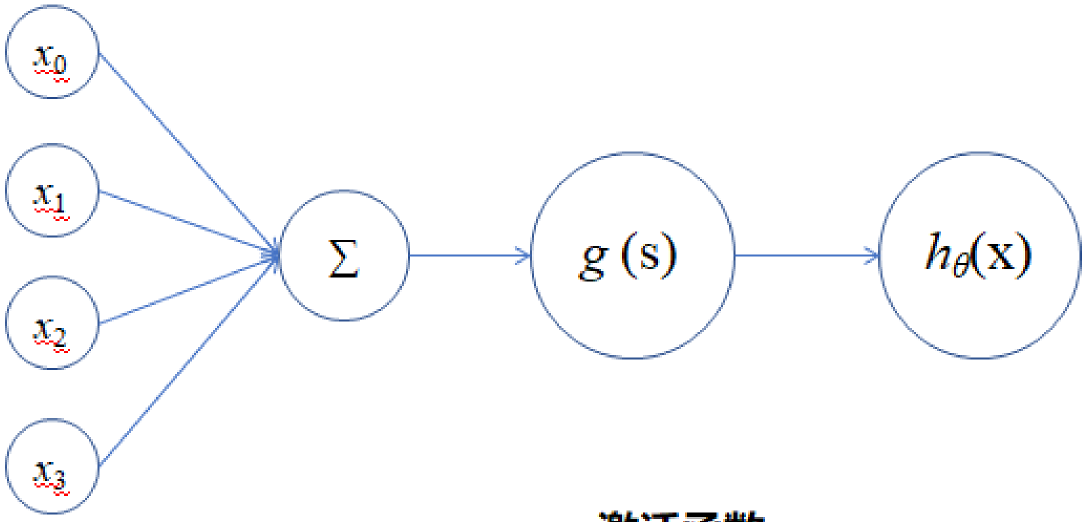
- 合法的分类超平面可能会有多个，甚至无穷个
- 感知机的解依赖于初值的选择、迭代过程中误分类点的顺序
- 如何得到一个最好的（同时可能也是唯一的）超平面？=> 线性支持向量机
- 感知机存在对偶形式，支持向量机也存在对偶形式

从感知机到神经网络

- **Input**: 所有的特征都成为感知器的输入，$x = [x_1, x_2\ ..., x_n]$。
- **Weights**: 权重是在模型训练过程中计算的值。初始化时，我们从某初始权重出发，基于每次训练误差进行权重更新。$w = [w_1, w_2, ...w_n]$。
- **BIAS**: 偏置神经元使得分类器可以将决策边界向左或向右移动。用代数的术语，偏置神经元允许分类器平移其决策边界。BIAS 有助于更快训练模型，获得更好的性能。
- **加权求和**: 加权求和是将每个特征值与对应权重相乘后得到的值之和。
- **激活函数**: 激活函数的作用是使神经网络具有非线性。
- **输出**: 加权求和被传递给激活函数，计算后得到的值即我们的预测输出。



总结



如果 $s \geq 0$, $h_\theta(x)$ 是正类

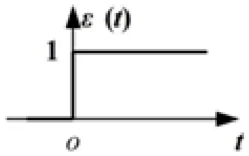
如果 $s < 0$, $h_\theta(x)$ 是负类

$$s = \sum_{i=0}^{m} x_i w_i = \mathbf{w}^{\mathrm{T}}\mathbf{x}$$

激活函数

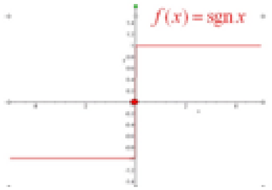
Step function
阶跃函数

$$g(s) = \begin{cases} 1, & if\ s \geq 0 \\ 0, & if\ s < 0 \end{cases}$$



Sign function
符号函数

$$g(s) = \begin{cases} 1, & if\ s \geq 0 \\ -1, & if\ s < 0 \end{cases}$$



总结

- **参数选择算法**
  - 修改“错误”
  - 收敛性

感知机：知错就改

《左传·宣公二年》：“过而能改，善莫大焉”

- **感知机的限制**
  - 无法解决线性不可分问题