第二讲：神经元

南京大学人工智能学院

申富饶

目录

CONTENTS

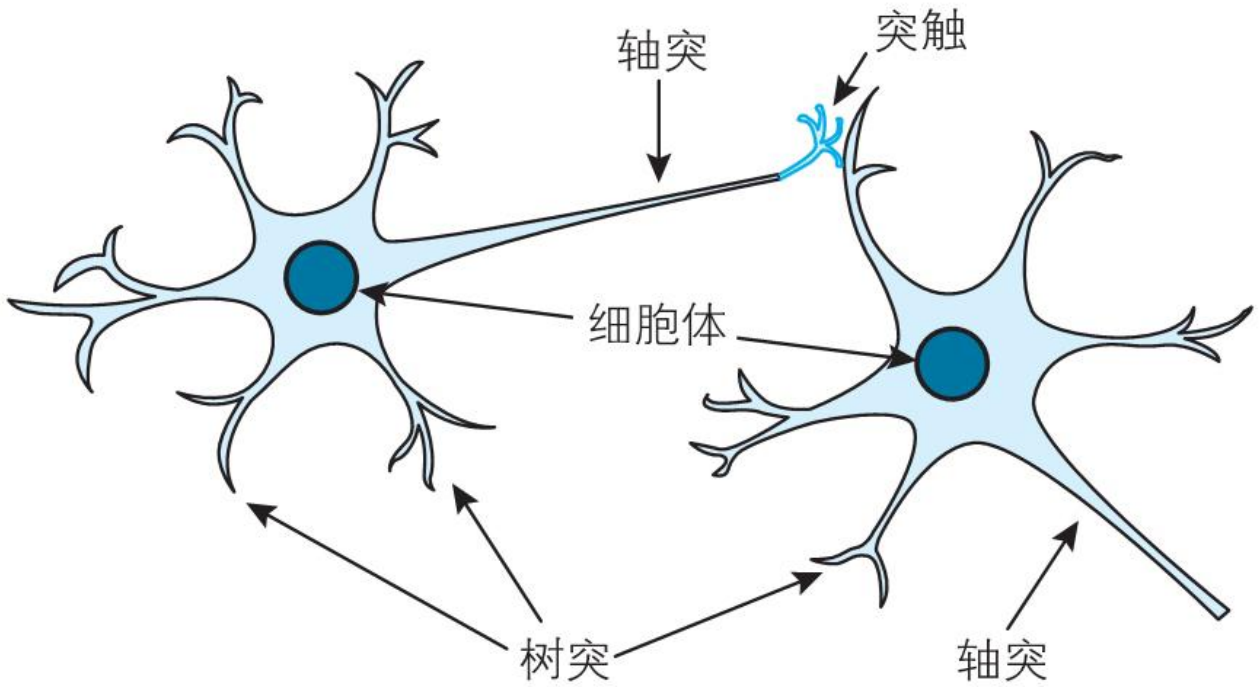
01 神经元模型

生物神经元

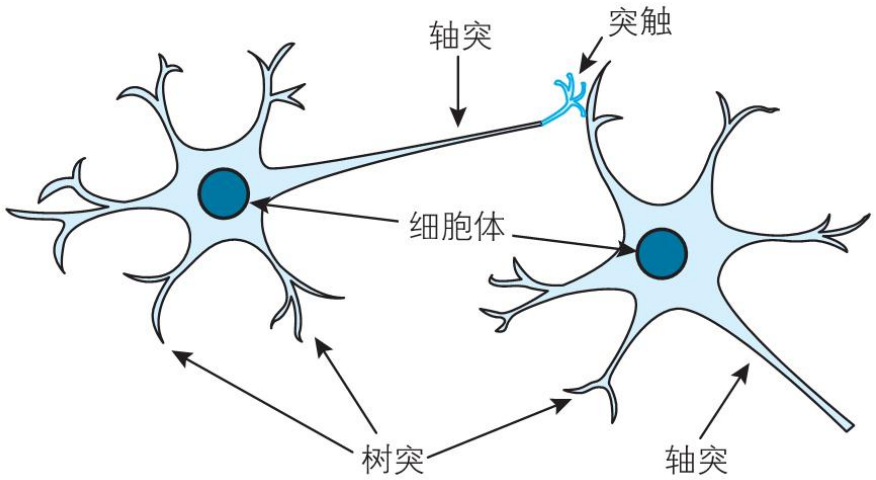
MP神经元

生物神经元

生物神经元主要由细胞体、树突、轴突和突触等构成。



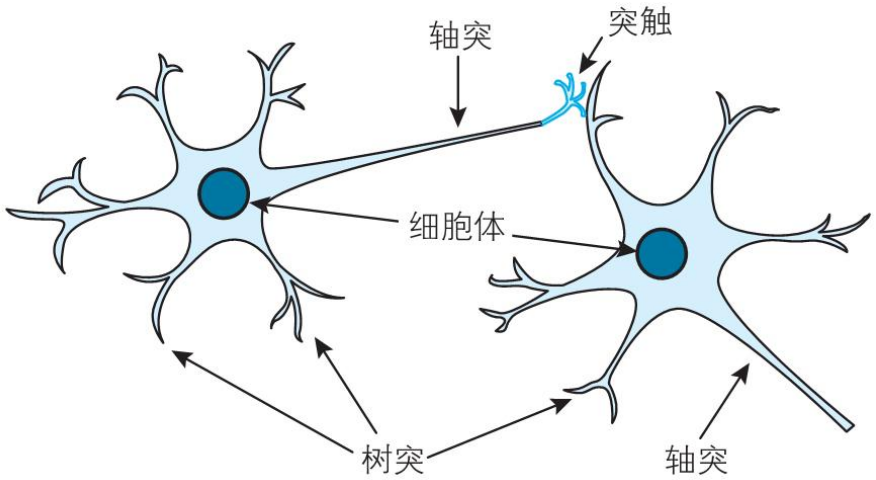
生物神经元



- **细胞体**(soma)
  - 神经元的控制中心，细胞核所在的位置。细胞体的边界是细胞膜，细胞膜将膜内外细胞液分开，膜内外存在离子浓度差，所以会出现电位差，这种电位差称为膜电位。

- **树突**(dendrite)
  - 突触的其中一种。从细胞体向外延伸出很多树突，负责接受来自其他神经元的信号，相当于神经元的输入端(input)。

生物神经元

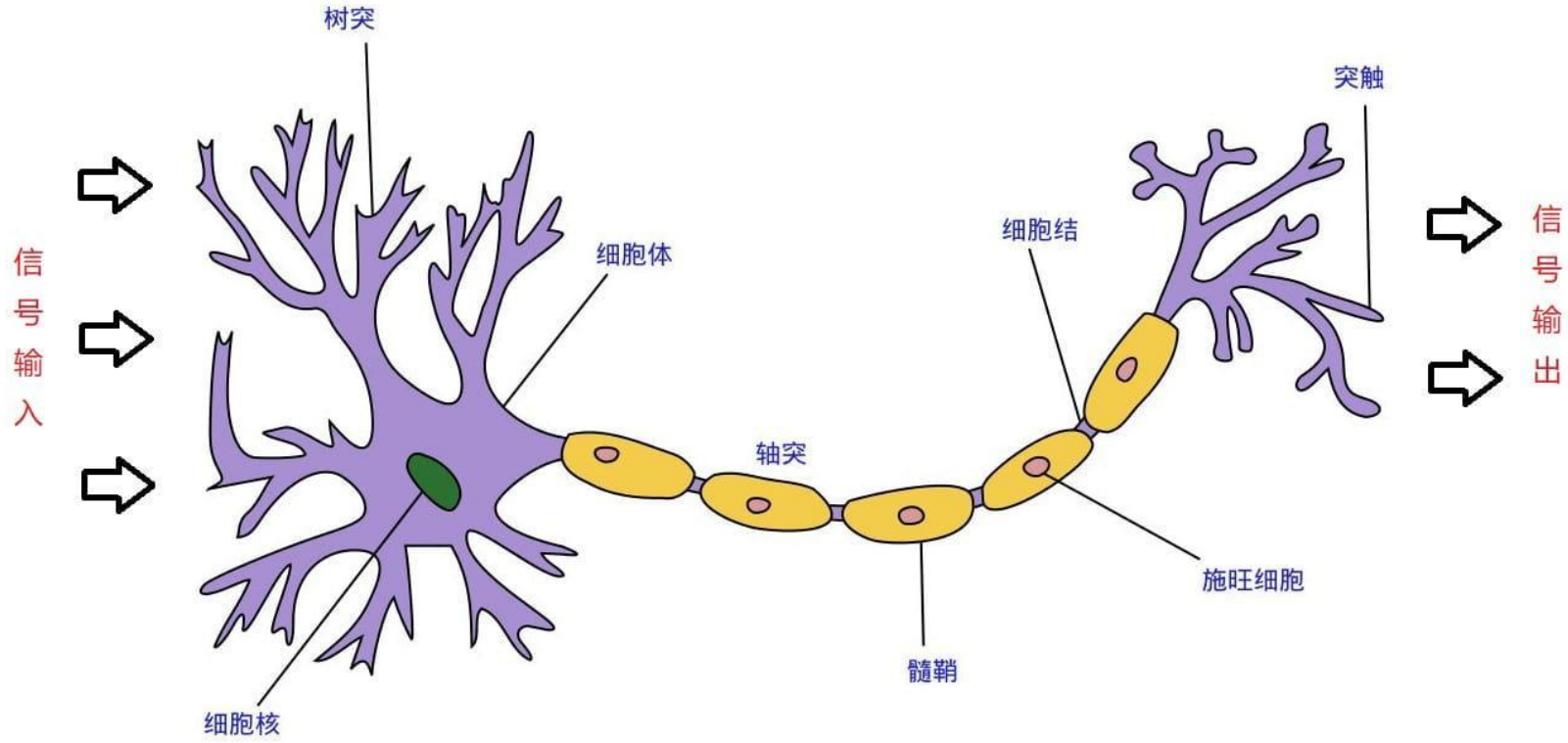


- **轴突**(axon)
  - 突触的其中一种。从细胞体向外延伸出的最长的一条突起。轴突比树突长而细，轴突也叫神经纤维，末端处有很多细的分支称为神经末梢，每一条神经末梢可以向四面八方传出信号，相当于神经元的输出端(output)。

- **突触**(synapse)
  - 一个神经元通过其轴突的神经末梢和另一个神经元的细胞体或树突进行通信连接，这种连接相当于神经元之间的输入/输出接口(I/O)。

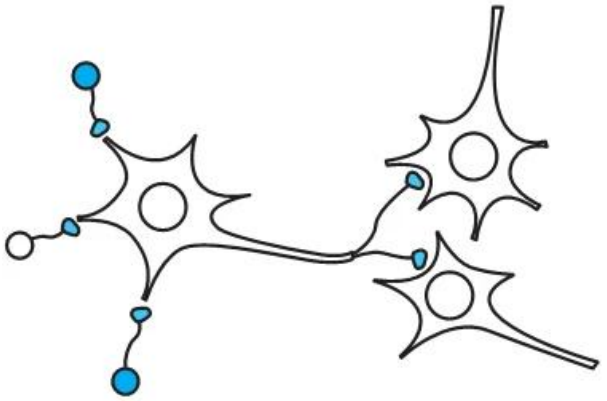
生物神经元

生物神经元信号传递过程：其他神经元的信号（输入信号）通过树突传递到细胞体中，细胞体把从其他多个神经元传递进来的输入信号进行合并加工，然后再通过轴突前端的突触将输出信号传递给别的神经元。

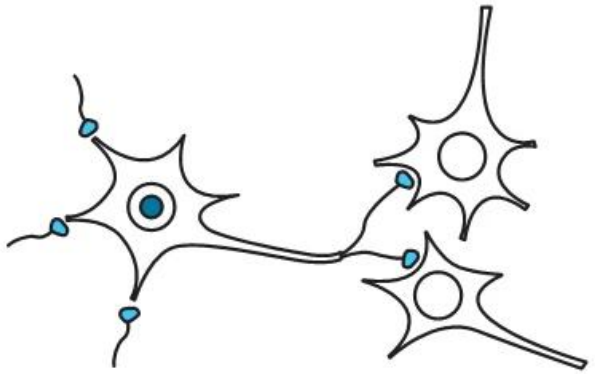


生物神经元

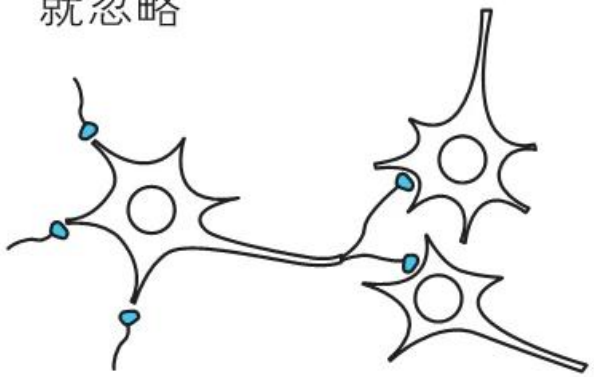
信号被输入到神经元



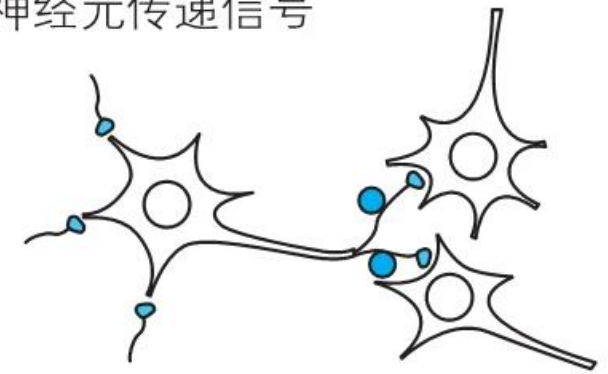
细胞体判断信号之和



当信号之和小于阈值时
就忽略



当信号之和大于阈值时，
进行点火，并向相邻的
神经元传递信号



MP神经元

MP设计思路

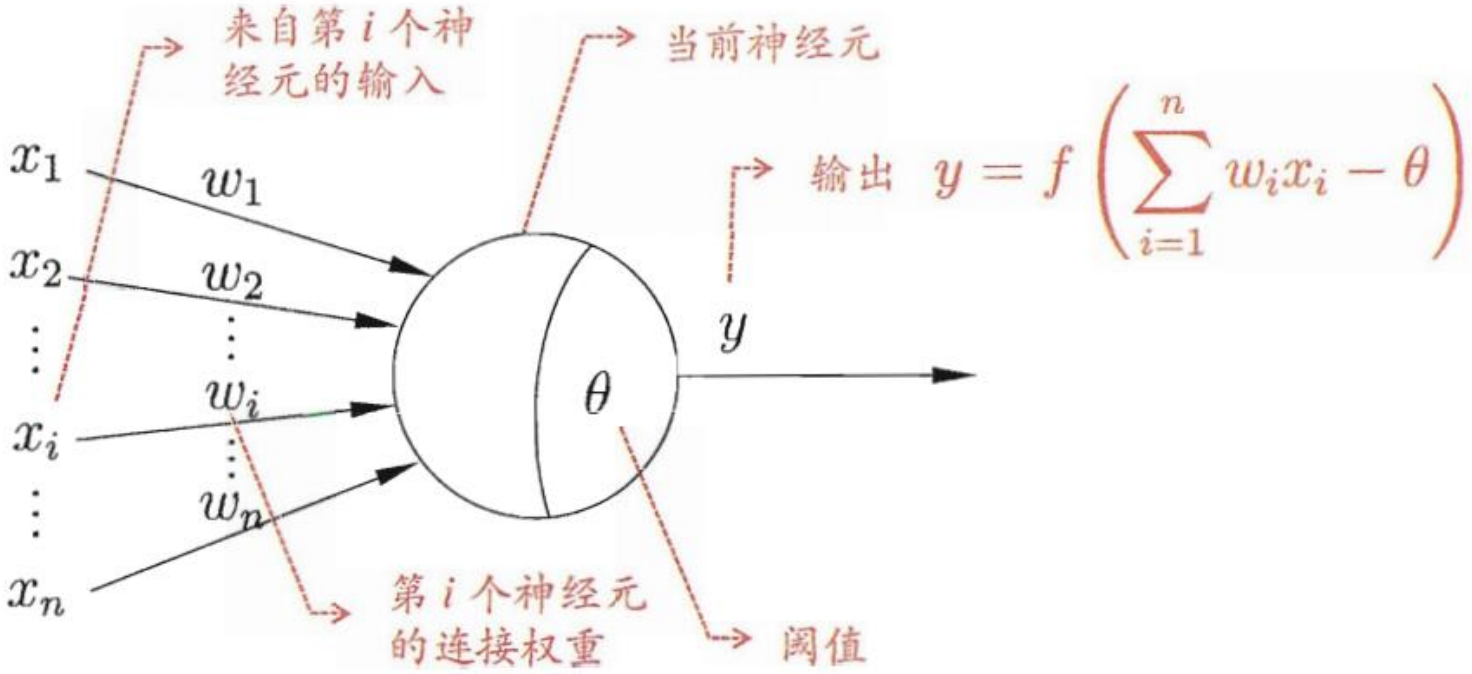
1943年，根据**生物神经元**的结构和工作原理，麦卡洛克（McCulloch）和皮茨（Pitts）提出几点假设：

- 每个神经元都是一个**多输入单输出**的信息处理单元；
- 神经元输入分**兴奋性输入**和**抑制性输入**两种类型；
- 神经元具有**空间整合特性**和**阈值特性**；
- 神经元输入与输出间有固定的时滞，主要取决于**突触间传播的延迟**；
- 忽略时间整合作用和不应期等；

基于这些假设，抽象出一个类神经元的运算模型，称为**M-P模型**。

MP神经元

MP模型结构



- 输入：$u = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$

- 输出：$y = \begin{cases} 0, & u \le \theta \\ 1, & u > \theta \end{cases}$

MP神经元

MP模型特征

- 二值网络（binary：fire-1, or not fire-0）：自变量及其函数的值、向量分量的值只取0和1的函数、向量。
- 由有方向的、带权重的路径联系
- 权为正：刺激；权为负：抑制；
- all-or-none：每个神经元有一个固定的阈值，如果输入大于阈值，就fires
- 绝对抑制：阈值被设为使得抑制为绝对的，即非 0 的抑制输入将阻止该神经元兴奋
- 花费一个时间单位使得信号通过一个连接

MP神经元

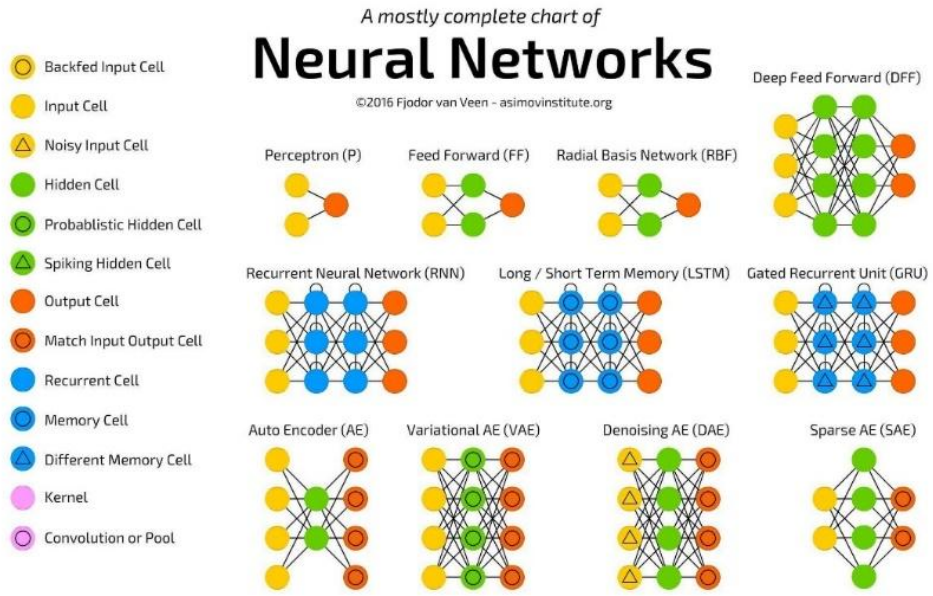
MP模型特征

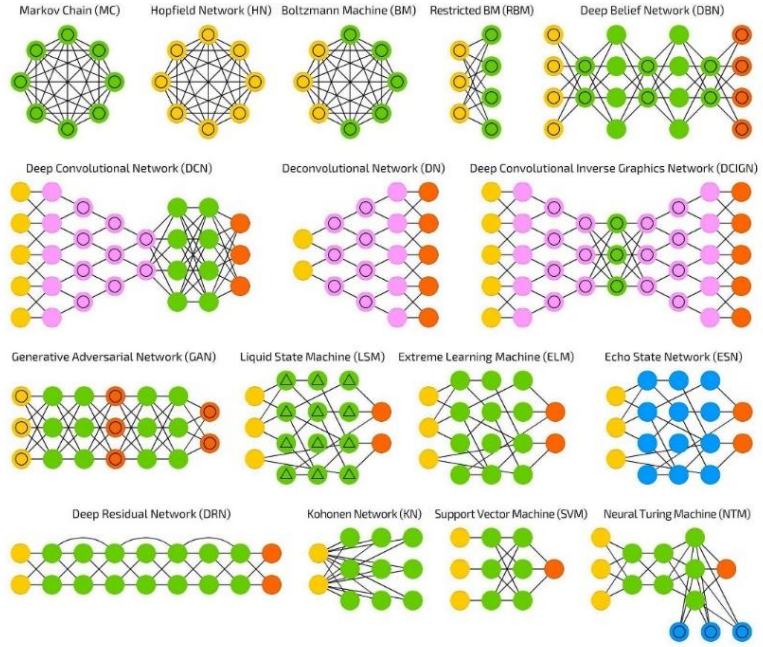
- 固定权重和阈值（用户指定），每个神经元可提供一个简单的逻辑函数
- 输出变为兴奋（fire）状态：接收到k个或者更多的刺激输入，没有抑制输入
- 采用离散时间，能够使用MP模型来模拟有时间延迟下的物理现象
- 任意命题逻辑函数都可由一两层的MP模型计算
- 所有的命题逻辑函数都可以用MP AND逻辑门、MP OR逻辑门、MP NOT逻辑门予以表达和实现
- MP模型具有神经计算模型一般和普遍的特性，可表达一般人工神经网络的赋权联结和相对抑制

MP神经元

MP模型特征

- 之后大部分的神经元结构都采用MP神经元的多输入单输出模式
- 多输入单输出的神经元通过不同的连接方式构成不同类型的神经网络
- 神经元内部对数据进行简单的线性或非线性变换实现对数据的拟合；神经网络通过整合多个神经元实现对复杂任务的处理



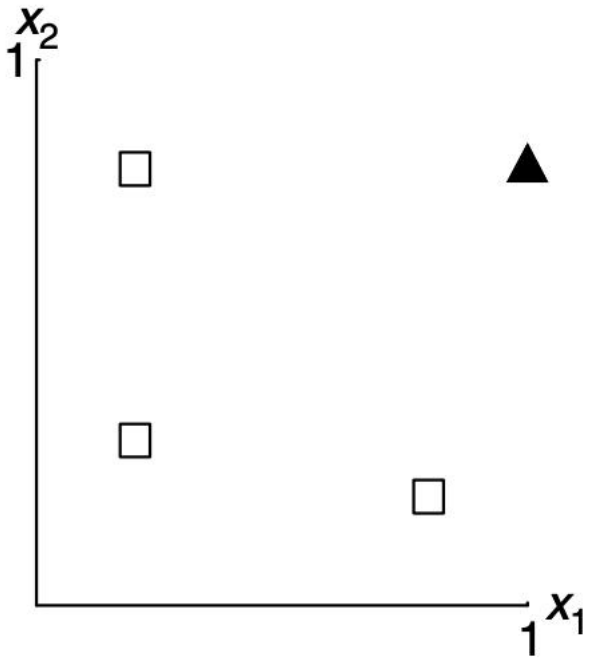


MP神经元示例：简单分类器

- 使用一个MP神经元实现简单分类器的功能

- 分类数据：

$X1$	$X2$	y
0.2	0.3	0
0.2	0.8	0
0.8	0.2	0
1.0	0.8	1



MP神经元示例：简单分类器

- **输入：** x_1和x_2（2维输入）
- **输出：** 正确的类别
- **神经元实现：**

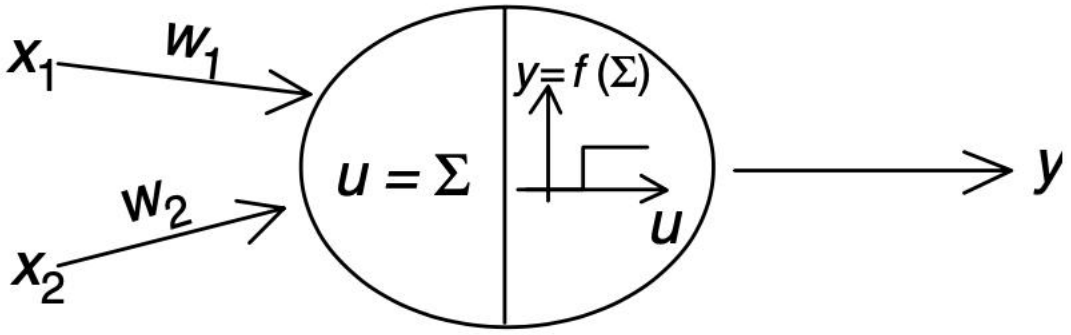
（1）通过w_1, w_2接收两个输入，在这里设定两个权值都为**1.0**

（2）计算过程：

a.计算网络输入u, $u = w_1x_1 + w_2x_2 = x_1+x_2$

b.使用阈值函数确定输出y，设定阈值为1.3

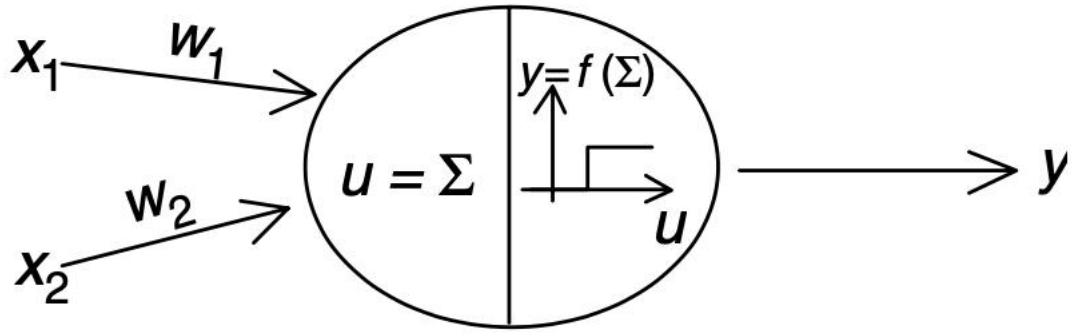
$$f(\Sigma) = y = \begin{cases} 0 & u < 1.3 \\ 1 & u \geq 1.3. \end{cases}$$



MP神经元示例：简单分类器

- 使用这个简单分类器，现在就能够检查它的性能。
- 对四个输入，神经元的输出结果如下表所示：

$(x1, x2)$	u	y
(0.2,0.3)	0.5	0
(0.2,0.8)	1.0	0
(0.8,0.2)	1.0	0
(1.0,0.8)	1.8	1

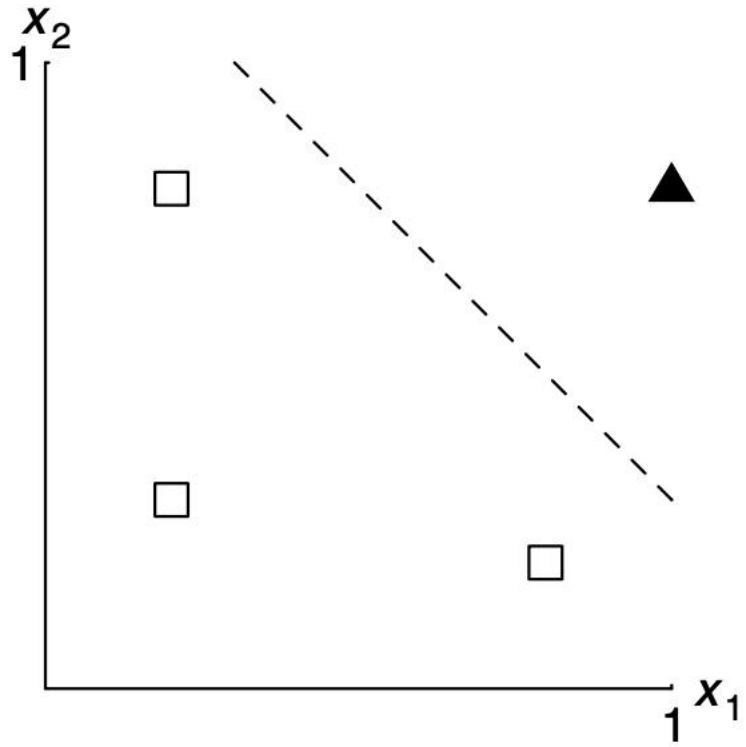


MP神经元示例：简单分类器

- 神经元根据预先设置的 $u = 1.3$ 的阈值函数，可以对数据进行正确分类。
- 阈值函数的位置定义了两类，$u = 1.3$ 决定一个分类界限可以表达为

$$= x_1 + x_2 = 1.3$$

$$x_2 = 1.3 - x_1$$



- 分类界限左下标记为0
- 分类界限右上标记为1

MP神经元的发展

- MP神经元是当下应用最广泛的神经元，他的设计遵循**Occam's razor**理念，简单的结构让其在很多任务中都减弱了过拟合的影响。

- MP神经元后来权重变为连续形式，作用在更多的连接结构中（例如卷积结构，循环结构等），从而产生了当下十分丰富的神经网络模型。

- MP神经元是**延展性最好**的一种神经元，研究者根据生物学理论和非线性理论提出过很多种神经元模型，例如脉冲神经元和乘法神经元，但往往存在神经元结构过于复杂或者计算量过大的问题，目前MP神经元也是在深度学习中应用最广的神经元。

02

神经元的组成成分

信息整合-激发

设计新型神经元

神经元的组成成分分析

1. 来自其他神经元的**输入信号**（$x_1, x_2 \ldots, x_n$）
2. 每一个输入信号都有一个与之对应的**突触权重**（$w_1, w_2, \ldots, w_n$），权重的高低反映了输入信号对神经元的重要性
3. **信号整合**（$\sum$）将经过加权的输入信号相加，生成一个"激活电压"
4. **激活阈值**（θ）给神经元的输出设置一个阈值
5. **激活电位**（u）是信号整合结果和激活阈值之差，如果$u \geq 0$，神经元产生的就是兴奋信号，如果$u < 0$，神经元产生的是抑制信号
6. **激活函数**（g）将神经元的输出限制在一个合理的范围内
7. 神经元产生的**输出信号**（y），可以传递给与之相连的其他神经元

神经元的组成成分分析

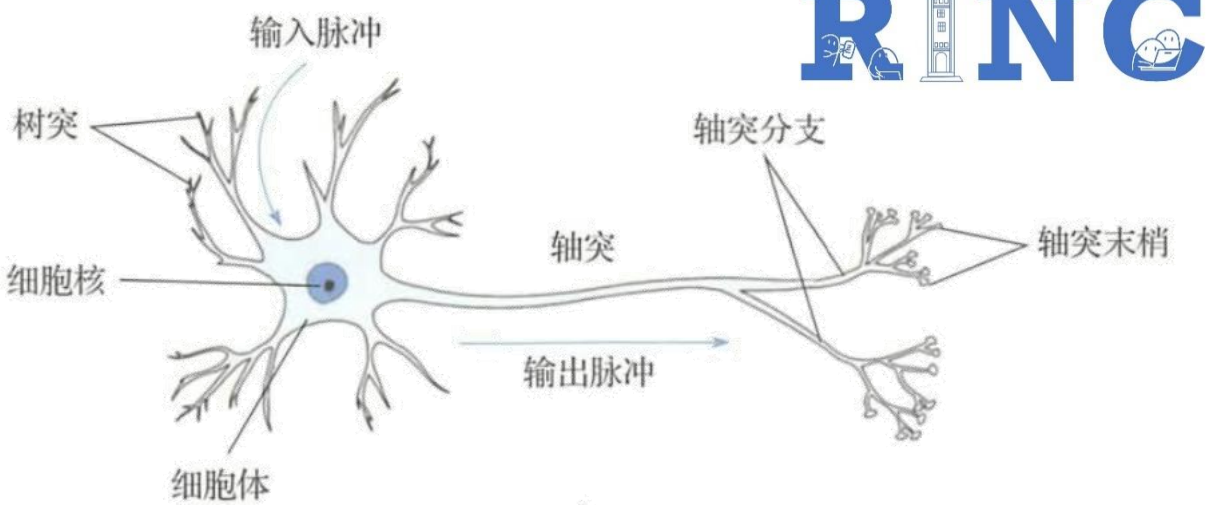
模拟生物神经元的整合-激发过程

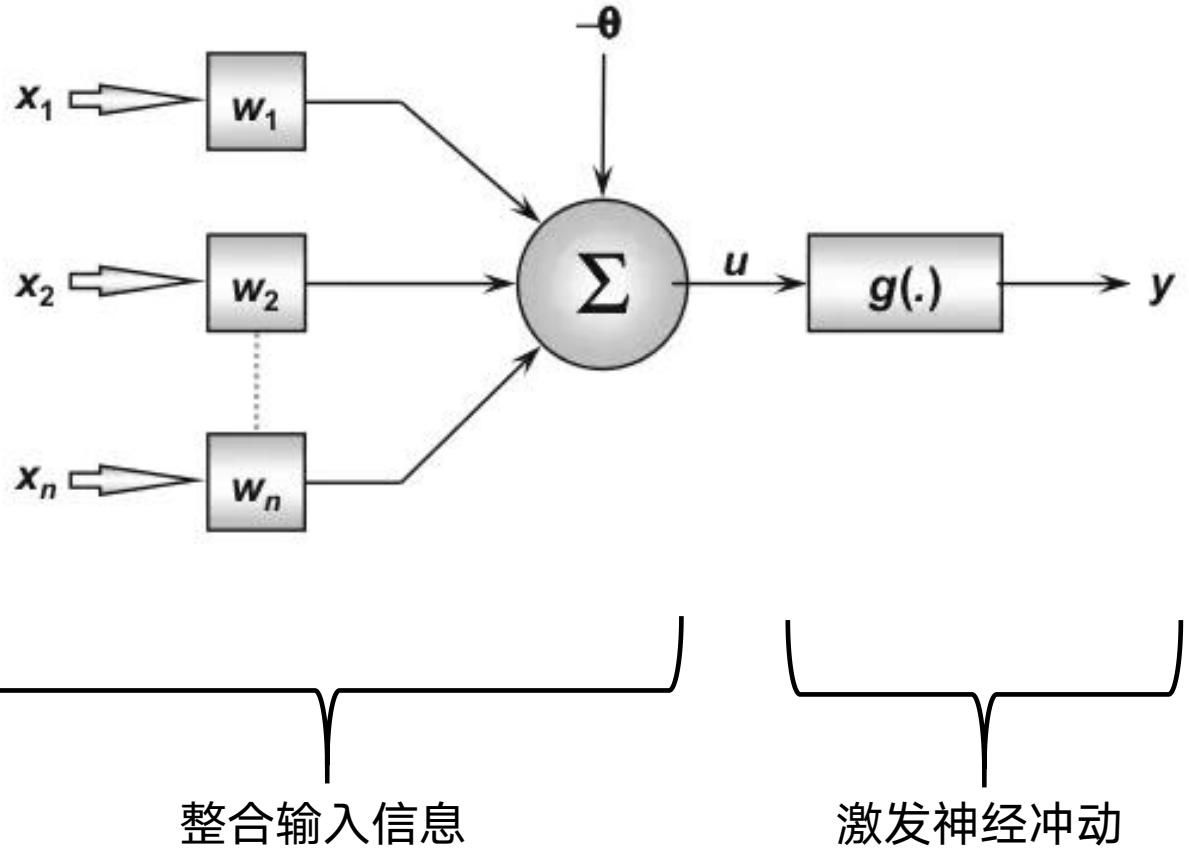
- 整合过程

$$u = \sum_{i=1}^{n} w_i \cdot x_i - \theta$$

- 激发过程

$$y = g(u)$$





设计新型神经元的思路

设计新的整合-激发方式

- 设计新的整合方式：
  - 乘法神经元

    Single Multiplicative Neuron
  - IC神经元

    Inter-Collision Neuron

- 设计新的激活函数：
  - Sigmoid函数
  - Tanh函数
  - ReLU函数

阈值函数

- 单极性阈值函数：

$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

- 多极性阈值函数：

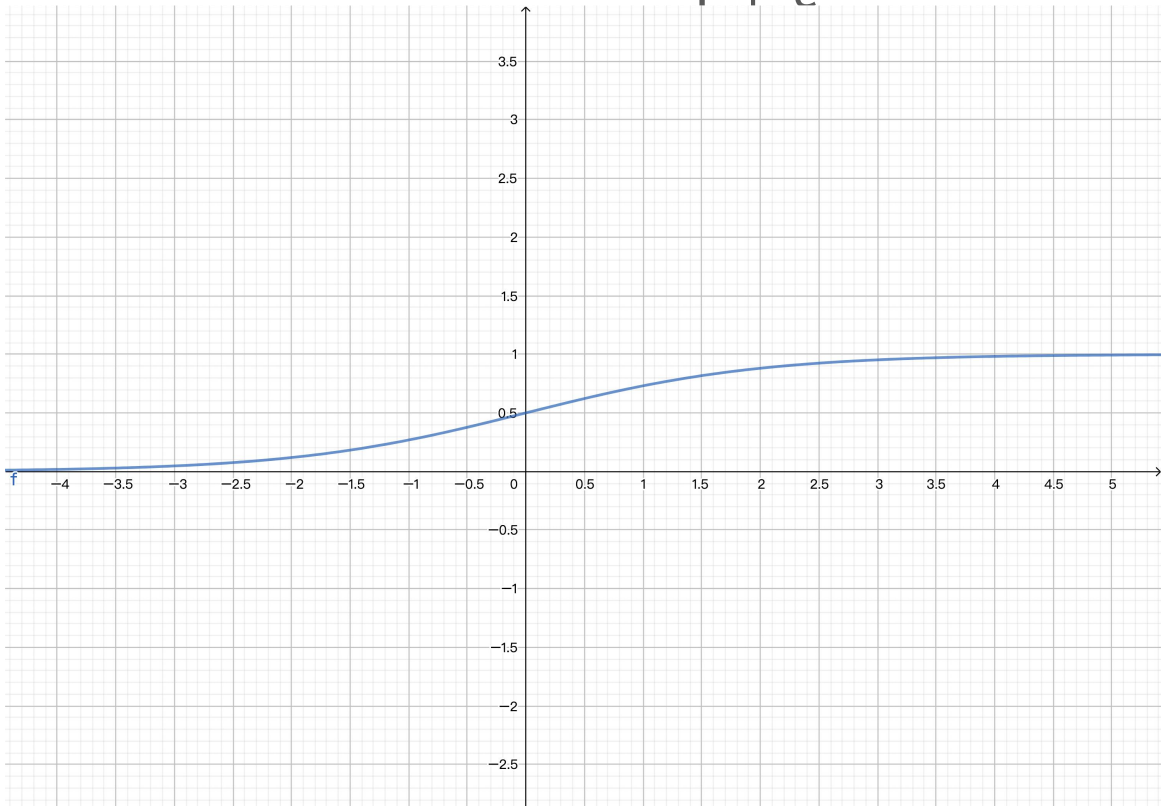
$$sgn(x) = \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases}$$

- 当结果大于某个阈值的时候，神经元为兴奋状态，输出值为1，小于阈值的时候，神经元为抑制状态，就输出0或-1。
- 但阈值函数不连续，我们可以设计新的激活函数把**输出值变为连续值。**

Sigmoid函数

Sigmoid 函数将输入从 $(-inf, inf)$ 映射到 (0,1) 区间。

$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

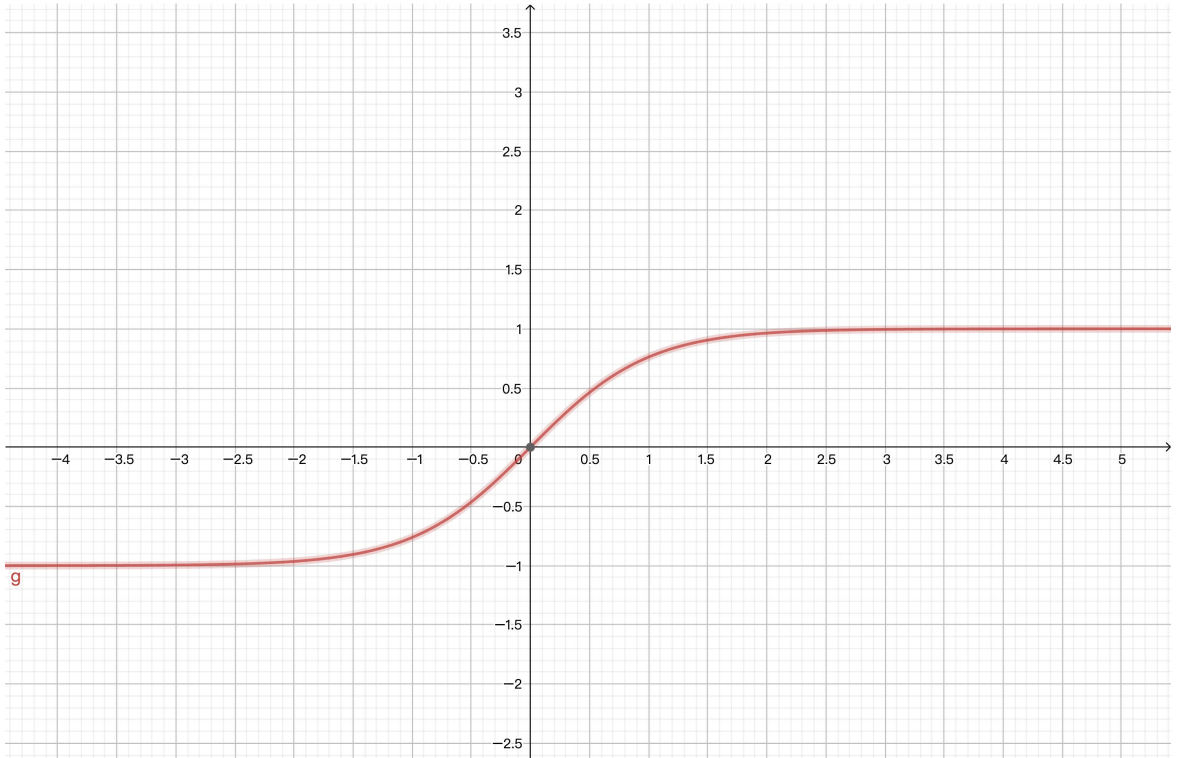


Tanh函数

作为后续网络的输入，Sigmoid 函数输出值恒为正值，随层数叠加，会产生累计偏差。

Tanh函数的形状和Sigmoid函数的形状很像，但Tanh函数在坐标系的原点上对称，是一种均值为0的激活函数。

$$tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

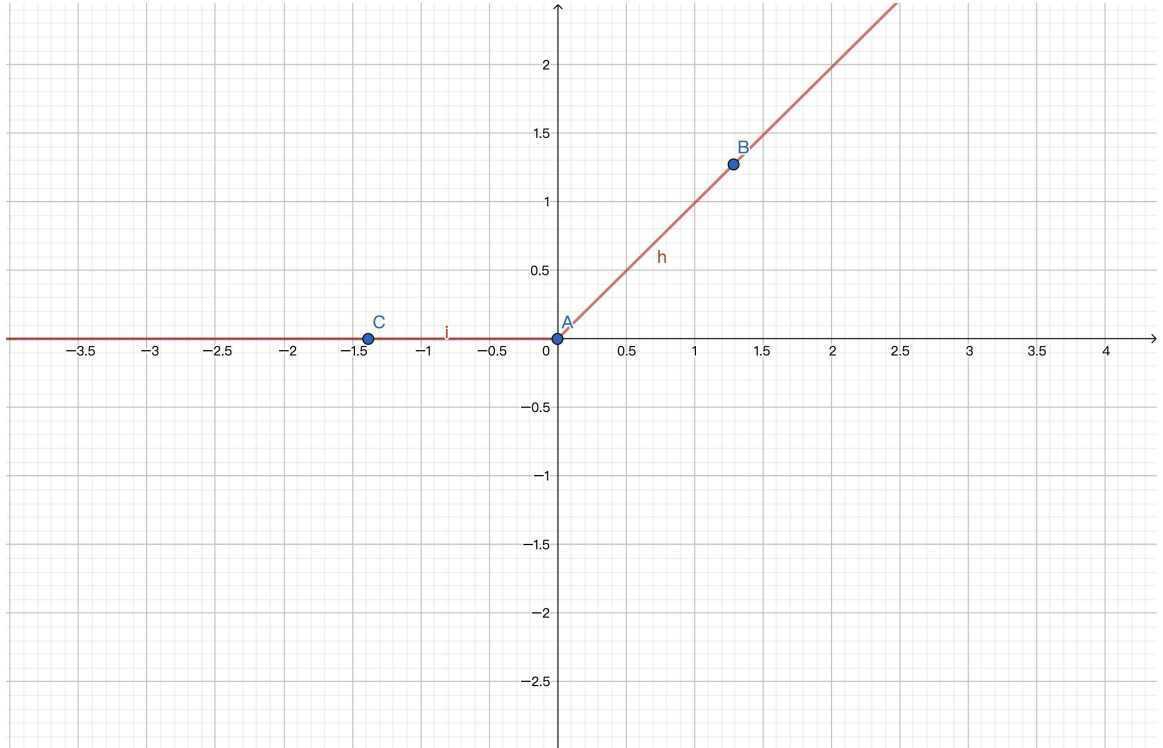


ReLU函数

当网络层数过多，Sigmoid函数和Tanh函数作为激活函数容易造成**梯度消失问题**。

ReLU函数只保留正数元素，并将负数元素清零，可以缓解梯度消失问题。

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



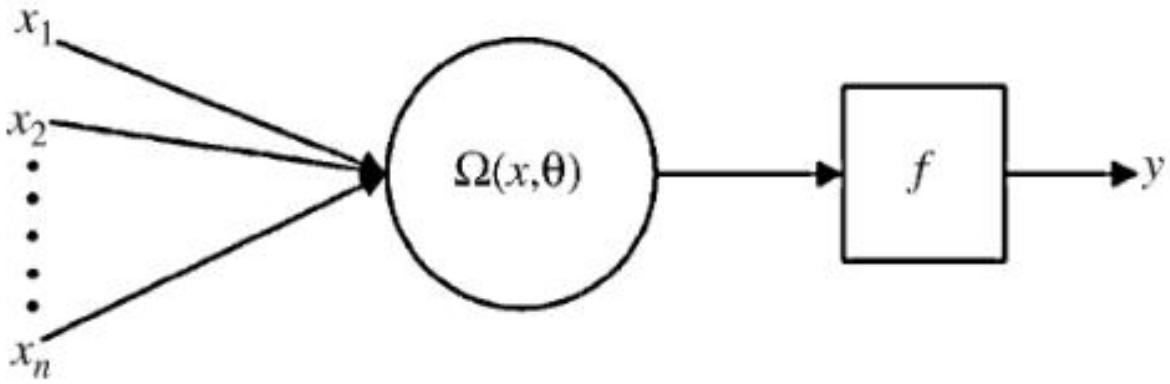
对比

激活函数	公式	比较
Sigmoid	$sigmoid(x) = \frac{1}{1+e^{-x}}$	连续且平滑便于求导； 恒为正值会产生累积偏差，存在梯度消失问题；
Tanh	$tanh(x) = \frac{2}{1+e^{-2x}} - 1$	可以减少累积偏差； 存在梯度消失问题；
ReLU	$relu(x) = \max(0, x)$	缓解梯度消失问题，收敛速度快； 存在Dead ReLU Problem(神经元坏死现象)；

乘法神经元

设计思路

从数学结构上进行优化，提升神经元表达能力



模型结构

用**累乘**来代替MP神经元中的累加

$$u = \prod_{i=1}^{n} (w_i x_i + b_i)$$

$$y = g(u) = \frac{1}{1 + e^{-u}}$$

性能

这种模型可以很好地拟合多项式分布的输入，在时间序列预测上效果较好

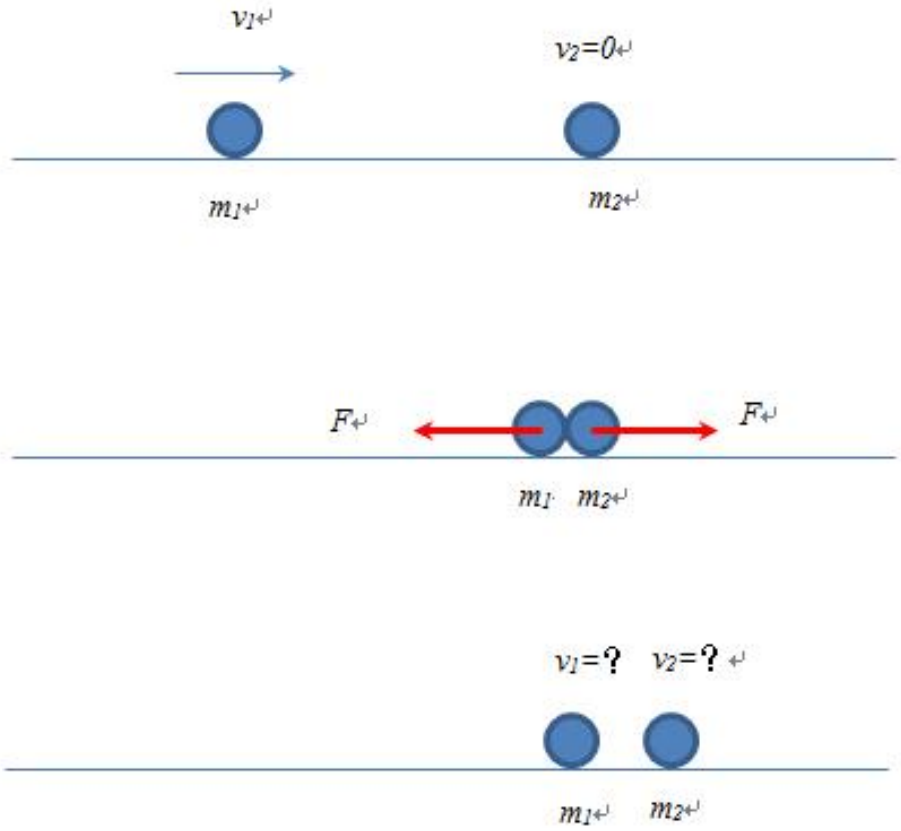
IC神经元

设计思路

- m_2 在碰撞后获得一个向右运动的速度(信息)。对应方程第一项。
- 碰撞后根据两小球的质量关系决定m_1 是否继续向右。对应方程 relu 项。

（换言之，两小球质量决定了 m_1 携带的信息是否重要到继续向后面的系统传递）

$$y_j = \sum_{N}^{i=1} w_{ij}x_i + relu(\sum_{N}^{i=1} w_{ij}x_i - w'_j x_{sum})$$

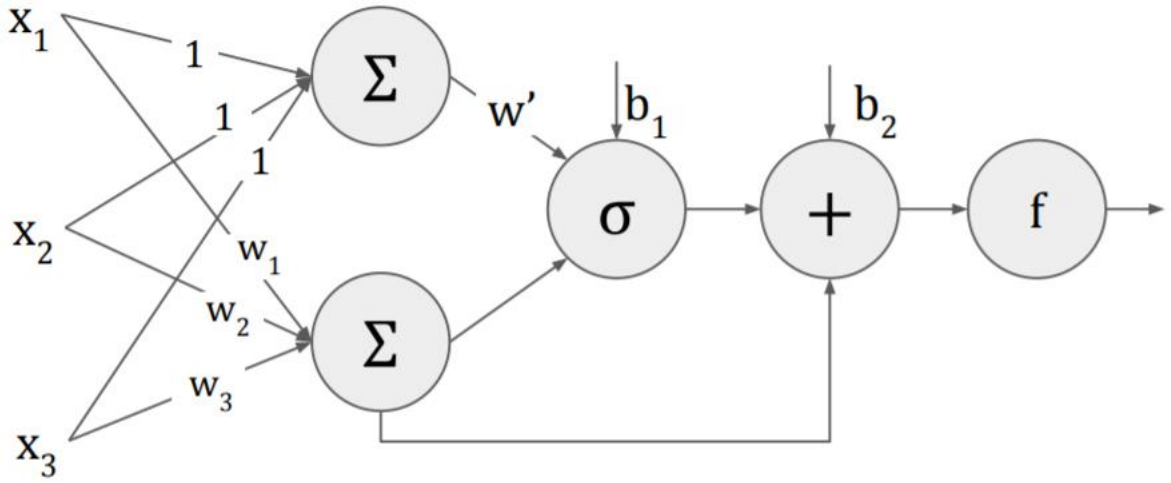


IC神经元

模型结构

➢ **非线性优化：**

用一个ReLU函数加强了MP神经元中线性变换的表达，神经元可从内部分成两部分。



$$y = \mathrm{f}\left(\sum_{i=1}^{n} w_i x_i + \sigma\left(\sum_{i=1}^{n} w_i x_i - w' x_{sum} + b_1\right) + b_2\right)$$

➢ **与MP的参数量和计算量对比：**

参数量：其中 W' 和 b_1 量是和神经数量有关的参数。

计算量：引入计算量主要在求和操作上，对于同一组输入，所有神经元可以共用一个结果。

$$y = \begin{cases} \mathrm{f}\left(2\sum_{i=1}^{n} w_i x_i - \sum_{i=1}^{n} x_i\right) & \text{if } H \geq 0 \\ \mathrm{f}\left(\sum_{i=1}^{n} w_i x_i\right) & \text{if } H < 0 \end{cases}$$

$$H = \sum_{i=1}^{n} (w_i - w') x_i$$

IC神经元

性能

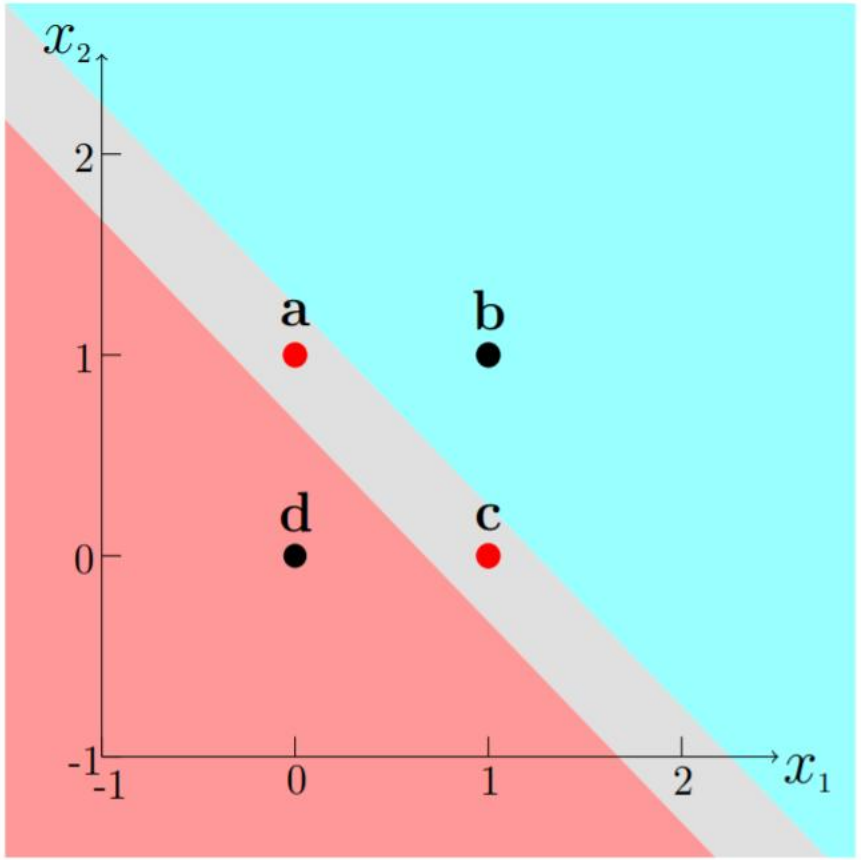
➢ **解决线性不可分问题：**

对应IC神经元公式 $w_1 = w_2 = 0.2805$

偏置 $b_1 = -0.3506$, $b_2 = 0.6463$

这里我们固定 $w' = 1$，如果通过训练寻找 w'，

我们还能找到更优的参数配置



➢ **效果：**

在能够引用MP神经元的多种网络模型和具体任务中，通过替换IC神经元，神经网络的性能能够得到一定提升且引入的参数量和计算量可以忽略。

对比

神经元	设计思路	形式化	优点	缺点
MP神经元	生物学启发	$y = f(\sum_{i=1}^{n} w_i x_i - \theta)$	模型简单且广泛适用，容易集成	无学习能力
乘法神经元	数学结构优化	$y = g(\prod_{i=1}^{n} w_i x_i + b_i)$	可以很好地拟合多项式分布的输入，适合时间序列预测	延展性差，难以集成
IC神经元	物理学启发：弹性碰撞模型	$y = f(\sum_{i=1}^{n} w_i x_i + \sigma(\sum_{i=1}^{n} w_i x_i - w' x_{sum} + b_1) + b_2)$	提升单个神经元非线性表达能力，提升网络性能	存在过拟合问题

03 感知机神经元

感知机神经元的概念

感知机神经元的学习

MP神经元的局限

MP神经元：它是首个通过模仿神经元而成的模型。

但是MP的权值只能事先给定，不能自动确定权值。**没有学习！**

↓

感知机神经元：模型结构与MP神经元基本相同。

但在MP神经元的基础上提出了**学习**的概念，设定训练样本和期望输出，通过学习不断调整实际权值。

什么是学习

人类具有学习能力，人的知识和智慧就是不断的学习与实践中逐渐形成和发展起来的。

生物神经元的角度

- 学习：是基于外界刺激而不断形成和改变神经元间突触联系的过程。

人工神经元的角度

- 学习：是基于样本数据而不断改变连接权值和拓扑结构的过程。

为什么要学习

人可以通过学习获得进步，神经网络也是如此。

- 遇到复杂问题无法通过手工设计参数和阈值；
- 如果神经元个数很多，不再适合手工设计各个神经元的参数。

↓

- 学习可以让神经元自动化调整参数，无需手工设置；
- 当大量神经元集体进行权值调整的时候，网络就呈现出“智能”的特性，其中有意义的信息就以分布的形式存储在调节后的权重矩阵中。

感知机神经元

感知机神经元模型

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。由输入空间到输出空间的如下函数：

$$f(x) = sgn(w \cdot x + b)$$

称为感知机。其中，w和b为感知机模型参数，$w \subseteq R^n$叫作权值(weight)或权值向量 (weight vector)，$b \in R$叫作偏置(bias)，$w \cdot x$表示w和x的内积。sgn是符号函数，即

$$sgn(x) = \begin{cases} +1, & x \geq 0 \\ -1, & x < 0 \end{cases}$$

感知机神经元

几何解释

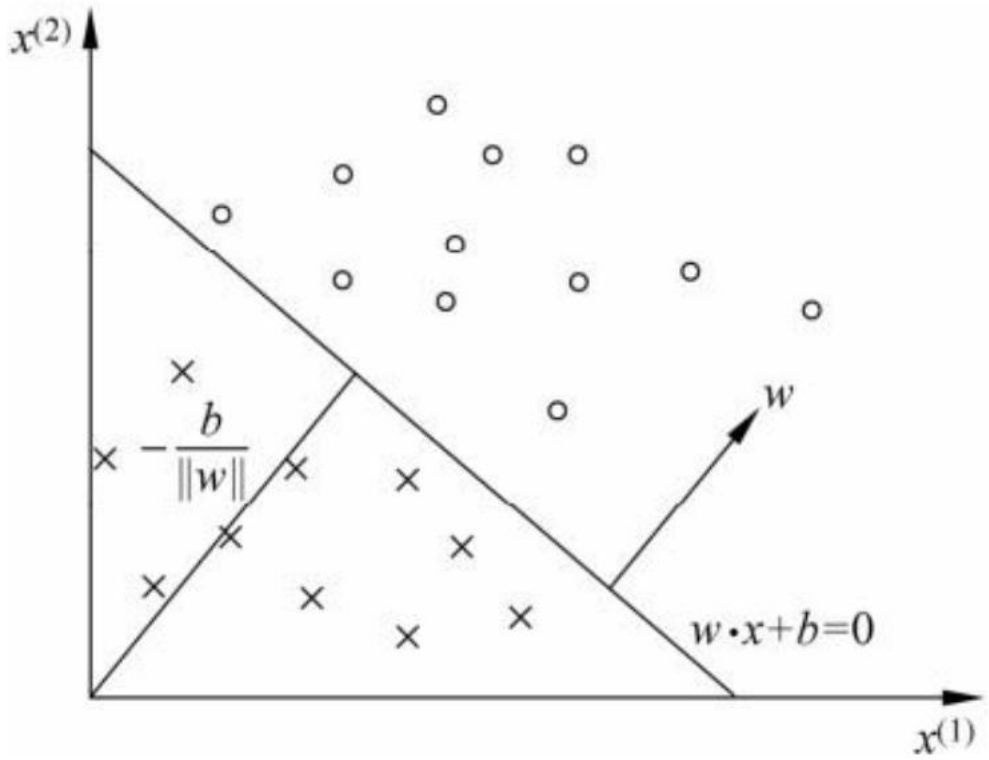
线性方程

$$w \cdot x + b = 0$$

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

这个超平面将特征空间划分为两个部分。位于两部分的点(特征向量)分别被分为正、负两类。

因此，超平面 S 称为分离超平面(separating hyperplane)。



感知机神经元的学习

感知机学习

感知机的学习是找到这个超平面的过程，由训练数据集(实例的特征向量及类别)

$$T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n))$$

其中，$x_i \in X = R^n$，$y_i \in Y=\{+1,-1\}$，$i=1,2,...,n$，求得感知机模型，即**求得模型参数w，b**。

通过学习得到的感知机模型，对于新的输入实例可以给出其对应的输出类别。

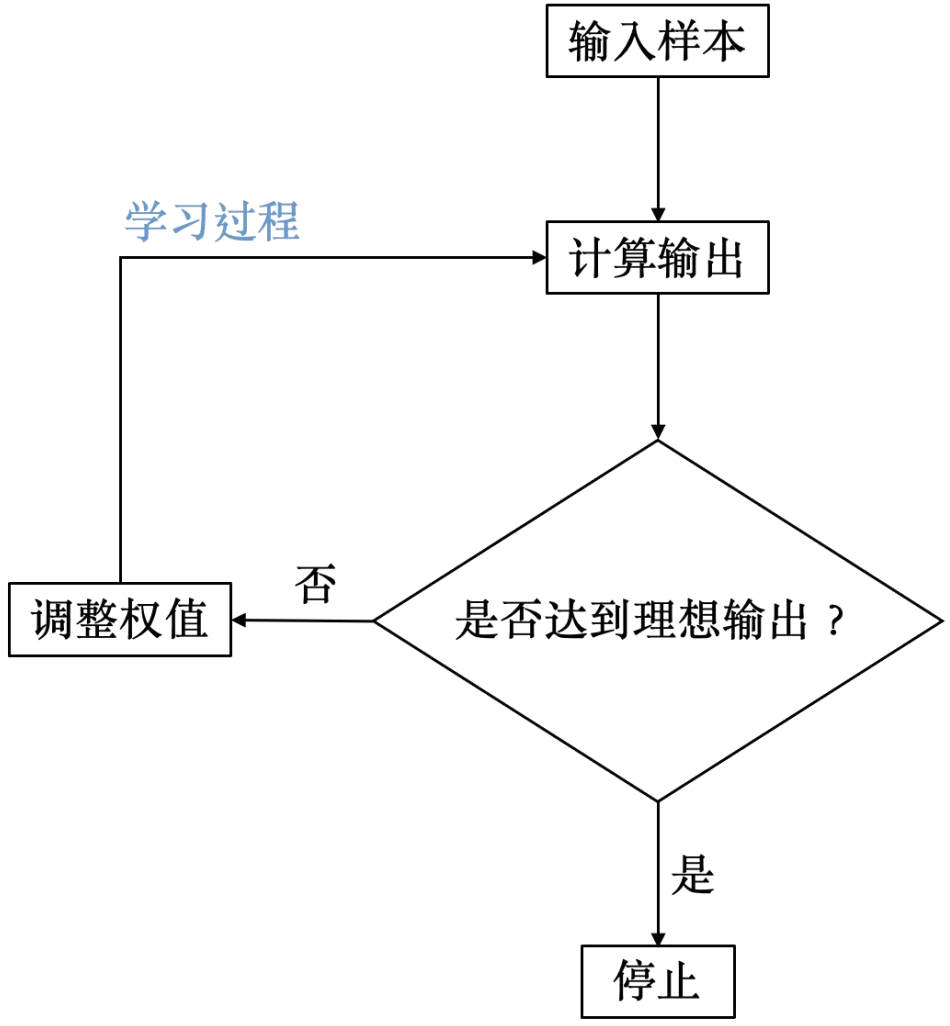
感知机学习算法的基本思路

输入：训练集$T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$

其中 $x_i \in X = R^n$， $y_i \in Y=\{+1,-1\}$， $i=1,2,...,n$

输出： w， b；

1. 对权值w， b进行初始化。
2. 训练集中选取一个样本数据 (x_i, y_i) 计算实际输出。
3. 将实际输出o_i与理想输出y_i做比较，根据**感知机学习规则**调整权值。
4. 转至（2）进行多轮迭代，直到所有样本预测正确或误差不超过指定范围。



感知机学习算法

学习规则

- 随机学习
- Hebb学习
- 基于误差的学习

感知机学习算法

学习规则1：随机学习

思路：随机更新权矩阵 W 和偏置矩阵 B。

那么：

- 在什么范围内给权矩阵 W 和偏置矩阵 B 赋值？
- 怎么判断随机给出的 W 和 B 是合适的？
- 当判断 W 和 B 不合适的时候，如何进行调整？

感知机学习算法

随机学习

解决思路：先给出随机赋值，然后慢慢对赋值进行修正。

- 在什么范围内给权矩阵 $\boldsymbol{W}$ 和偏置矩阵 $\boldsymbol{B}$ 赋值？

答：在一个小范围 $(-\delta, +\delta)$ 内给 $\boldsymbol{W}$ 和 $\boldsymbol{B}$ 赋予初始值

- 怎么判断随机给出的 $\boldsymbol{W}$ 和 $\boldsymbol{B}$ 是合适的？

答：利用 $\boldsymbol{W}$ 和 $\boldsymbol{B}$ 算出输出 $\boldsymbol{O}$，考虑它与真实值 $\boldsymbol{Y}$ 的误差 $|\boldsymbol{O}-\boldsymbol{Y}|<\epsilon$ 是否成立

- 当判断 $\boldsymbol{W}$ 和 $\boldsymbol{B}$ 不合适的时候，如何进行调整？

答：在 $\boldsymbol{W}$ 和 $\boldsymbol{B}$ 上加上一个很小的 $(-\xi, +\xi)$ 范围内的随机数，再重新判断

感知机学习算法

随机学习

总结：

尽管随机学习效率很低，但其思想简单，实现容易，且具有能找到全局最优解等特点，在对其搜索方法进行改进后，也能够具有一定的实际应用意义

感知机学习算法

学习规则2：Hebb学习

1949年，心理学家D.O.Hebb提出了关于神经网络学习机理的“突触修正”假设。

该假设指出，当神经元的突触前膜电位与后膜电位同时为正时，突触传导增强，当前膜电位与后膜电位正负相反时，突触传导减弱。

也就是说，当神经元i与神经元j同时处于兴奋状态时，两者之间的连接强度应增强。

根据该假设定义的权值调整方法，称为**Hebb学习规则**。

感知机学习算法

Hebb学习

- 权值调整量 Δw 与输入 x 和输出 o 的乘积成比例：

$$o = f(w \cdot x)$$

$$\Delta w = \eta o x$$

- 权值的新值是

$$w_{new} = w_{old} + \Delta w = w_{old} + \eta o x$$

- 比例系数 η 叫做“学习率”，决定学习发生的速度。η 越大，权值改变得越快。

感知机学习算法

Hebb学习示例

- 输入向量：

$$x_1 = (1, -2, 1.5, 0)\ ,\quad x_2 = (1, -0.5, -2, -1.5)\ ,\quad x_3 = (0, 1, -1, 1.5)$$

- 初始化权值：$w^0 = (1, -1, 0, 0.5)$
- 向量点积：$net = w \cdot x$
- 设置阈值函数：$o = f(net) = sgn(net)$
- 权值更新：$w_{new} = w_{old} + \eta o x$　（η是学习率）

感知机学习算法

Hebb学习示例

- 假定学习率 $\eta = 1$

➢ 输入第一个样本 x_1

$$net^1 = w^0 \cdot x_1 = (1)(1) + (-1)(-2) + (0)(1.5) + (0.5)(0) = 3$$

$$o^1 = f(net^1) = sgn(3) = 1$$

$$w^1 = w^0 + \eta o^1 x_1 = (1, -1, 0, 0.5) + (1, -2, 1.5, 0) = (2, -3, 1.5, 0.5)$$

感知机学习算法

Hebb学习示例

- 输入第二个样本 x_2

$$net^2 = w^1 \cdot x_2 = (2)(1) + (-3)(-0.5) + (1.5)(-2) + (0.5)(-1.5) = -0.25$$

$$o^2 = f(net^2) = sgn(-0.25) = -1$$

$$w^2 = w^1 + \eta o^2 x_2 = (2, -3, 1.5, 0.5) - (1, -0.5, -2, -1.5) = (1, -2.5, 3.5, 2)$$

感知机学习算法

Hebb学习示例

- 输入第三个样本 x_3

$$net^3 = w^2 \cdot x_3 = (1)(0) + (-2.5)(1) + (3.5)(-1) + (2)(1.5) = -3$$

$$o^3 = f(net^3) = sgn(-3) = -1$$

$$w^3 = w^2 + \eta o^3 x_3 = (1, -2.5, 3.5, 2) - (0, 1, -1, 1.5) = (1, -3.5, 4.5, 0.5)$$

感知机学习算法

基于误差的学习

设 y_i 是期望输出，$o_i = f(w \cdot x_j - \theta)$是实际输出，则误差：

$$r = y_i - o_i$$

权重调整公式为：

$$\Delta w = \eta r x$$

$$w_{new} = w_{old} + \Delta w = w_{old} + \eta r x$$

感知机学习算法

基于误差的学习示例

- 训练样本：

$x_1 = (-1,1,-2,0)$，　$y_1 = -1$；　　$x_2 = (-1,0,1.5,-0.5)$，　$y_2 = -1$；

$x_3 = (-1,-1,1,0.5)$，　$y_3 = 1$；

- 初始化权值：　$w^0 = (0.5,1,-1,0)$

- 向量点积：$net = w \cdot x$

- 设置阈值函数：$o = f(net - \theta) = sgn(net - \theta)$

- 误差：$r = y_i - o_i$

- 权值更新：$w_{new} = w_{old} + \eta r x$　　(η是学习率)

感知机学习算法

基于误差的学习示例

- 假定学习率 $\eta = 0.1$，阈值 $\theta = 0$

➢ 输入第一个样本 x_1

$$net^1 = w^0 \cdot x_1 = (0.5)(-1) + (1)(1) + (-1)(-2) + (0)(0) = 2.5$$

$$o_1 = f(net^1) = sgn(2.5) = 1$$

$$w^1 = w^0 + \eta(y_1 - o_1)x_1 = (0.5,1,-1,0) + 0.1(-1-1)(-1,1,-2,0) =$$

$$(0.7,0.8,-0.6,0)$$

感知机学习算法

基于误差的学习示例

➢ 输入第二个样本 x_2

$$net^2 = w^1 \cdot x_2 = (0.7)(-1) + (0.8)(0) + (-0.6)(1.5) + (0)(-0.5) = -1.6$$

$$o_2 = f(net^2) = sgn(-1.6) = -1$$

$$w^2 = w^1 + \eta(y_2 - o_2)x_2 = (0.7,0.8,-0.6,0) + 0.1(-1-(-1))$$

$$(-1,0,1.5,-0.5) = (0.7,0.8,-0.6,0)$$

由于 $y_2 = o_2$ ，所以 $w^2 = w^1$

感知机学习算法

基于误差的学习示例

➢ 输入第三个样本 x_3

$$net^3 = w^2 \cdot x_3 = (0.7)(-1) + (0.8)(-1) + (-0.6)(1) + (0)(0.5) = -2.1$$

$$o_3 = f(net^3) = sgn(-2.1) = -1$$

$$w^3 = w^2 + \eta(y_3 - o_3)x_3 = (0.7, 0.8, -0.6, 0) + 0.1(1 - (-1))$$

$$(-1, -1, 1, 0.5) = (0.5, 0.6, -0.4, 0.1)$$

➢ 继续输入样本进行训练，直到 $r = d_p - o_p = 0$，$p = 1, 2, 3$

神经元模型总结

- “多输入——单输出”的结构特征
  - 众多的树突为神经信号提供输入通道
  - 单一的轴突为神经信号提供输出通道
- “整合——激发”的功能特征
  - 整合输入信号：时间整合与空间整合功能
  - 激发神经冲动：“全或无”式的兴奋与抑制功能
- 记忆和学习
  - 联结权重的存储与更新
- 人工神经元就是一个具有记忆功能的输入权值化的“多输入—单输出”“整合—激发”装置。
  - 是一个信息处理单元，一个自动机
  - 可以概括各类人工神经元

结构、功能、记忆和学习的方式形成不同的神经元

04

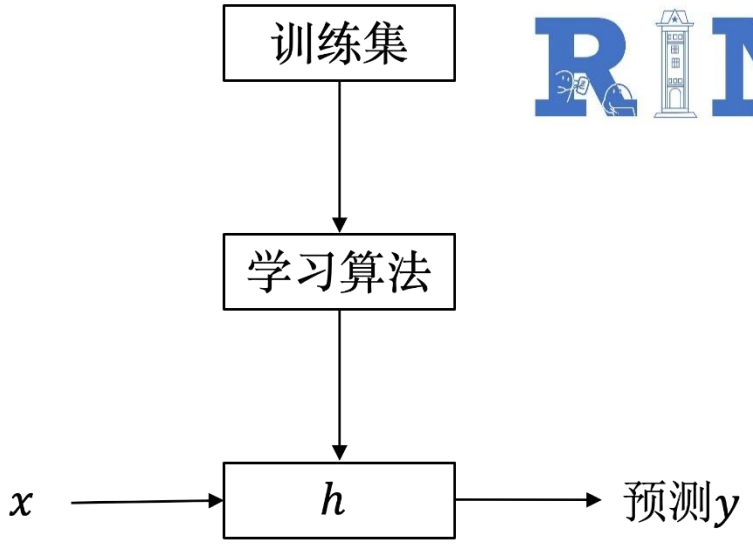
神经元的应用

单输入-单输出神经元应用

多输入-单输出神经元应用

单个神经元的局限

单个神经元的应用



回归和分类问题

定义：

- 用X表示输入空间，用Y表示输出空间，$X = Y = \mathbb{R}$。
- 给定训练集，学习一个映射：h: $X \mapsto Y$，使得h(x)是y的“好”预测。
- 习惯上，方程h被称为一个**假设**(hypothesis)
- 如果 y 取连续值（例如预测股票价格），则这个问题被称为**回归**问题(regression);
- 如果 y 取有限数量的离散值（例如预测股票是涨还是跌），这个问题被称为**分类**问题(classification)。

线性回归

以股票预测问题为例：

假设某只股票近二十天的股价记录如下：

日期	1	2	3	4	5	6	7	8	9	10
股价	1.0	0.5	1.9	2.5	2.6	3.2	3.4	4.4	4.3	5.8
日期	11	12	13	14	15	16	17	18	19	20
股价	5.6	6.3	7.0	7.0	7.2	9.0	8.6	8.7	9.6	10.4

如何通过日期来预测股票价格？

尝试建立一个描述日期与股价关系的方程。

线性回归

- 可以看出来股价和日期非常接近线性关系，不妨假设$h(x) = wx + b$。
- 这样线性关系的回归问题被称为**线性回归**(linear regression)。
- 其中w, b为所需求得的**参数**(parameter)。
- 现在，给定了训练集，为了求得参数还需要一个标准来评价预测$h(x)$是否足够“好”。

线性回归

- 线性回归的目标是要得到一条尽可能表示数据分布的线性函数。
- 可以用均方误差表示这一目标：
- 训练集$\{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$大小为$n$，如下公式

$$E = \frac{1}{n} \sum_{i=1}^{n} [h(x^{(i)}) - y^{(i)}]^2$$

被称为假设h的**均方误差**(mean squared error)。均方误差越小越“好”。

- 常用**最小二乘法**最小化均方误差。

线性回归

最小二乘法：

这样，问题就被化为求

$$\underset{w,b}{\operatorname{argmin}} E = \underset{w,b}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (wx_i + b - y_i)^2$$

要求使得E最小的w, b，可以用E对它们分别求偏导。

线性回归

$$\frac{\partial E}{\partial w} = \frac{2}{n}\left(w\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n}(b - y_i)x_i\right) \quad (1)$$

$$\frac{\partial E}{\partial b} = 2\left(nb + \sum_{i=1}^{n}(wx_i - y_i)\right) \quad (2)$$

令两式为零可求得w与b的**闭式解**(closed-form solution)

对(2)式：

$$nb + \sum_{i=1}^{n}(wx_i - y_i) = 0$$

$$\Longrightarrow b = \frac{\sum_{i=1}^{n}(y_i - wx_i)}{n}$$

线性回归

对(1)式：

$$w\sum_{i=1}^{n}x_i^2+\sum_{i=1}^{n}(b-y_i)x_i=0\Longrightarrow w\sum_{i=1}^{n}x_i^2+\sum_{i=1}^{n}(\frac{\sum_{j=1}^{n}(y_j-wx_j)}{n}-y_i)\,x_i=0$$

$$\Longrightarrow w\,[\sum_{i=1}^{n}x_i^2-\frac{1}{n}\sum_{i=1}^{n}(\sum_{j=1}^{n}x_j)\,x_i]=\sum_{i=1}^{n}(y_i-\frac{\sum_{j=1}^{n}y_j}{n})\,x_i$$

$$\Longrightarrow w=\frac{\sum_{i=1}^{n}x_i\,(y_i-\frac{\sum_{j=1}^{n}y_j}{n})}{\sum_{i=1}^{n}x_i^2-\frac{1}{n}(\sum_{i=1}^{n}x_i)^2}$$

其中$\bar{x}=\frac{\sum_{i=1}^{n}x_i}{n}$，$\bar{y}=\frac{\sum_{i=1}^{n}y_i}{n}$为$x,y$的均值。

线性回归

$$w = \frac{\sum_{i=1}^{n} x_i \left(y_i - \frac{\sum_{j=1}^{n} y_j}{n}\right)}{\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2} = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2}$$

$$b = \frac{\sum_{i=1}^{n} (y_i - w x_i)}{n} = \bar{y} - w\bar{x}$$

其中 $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$， $\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$ 为 x, y 的均值。

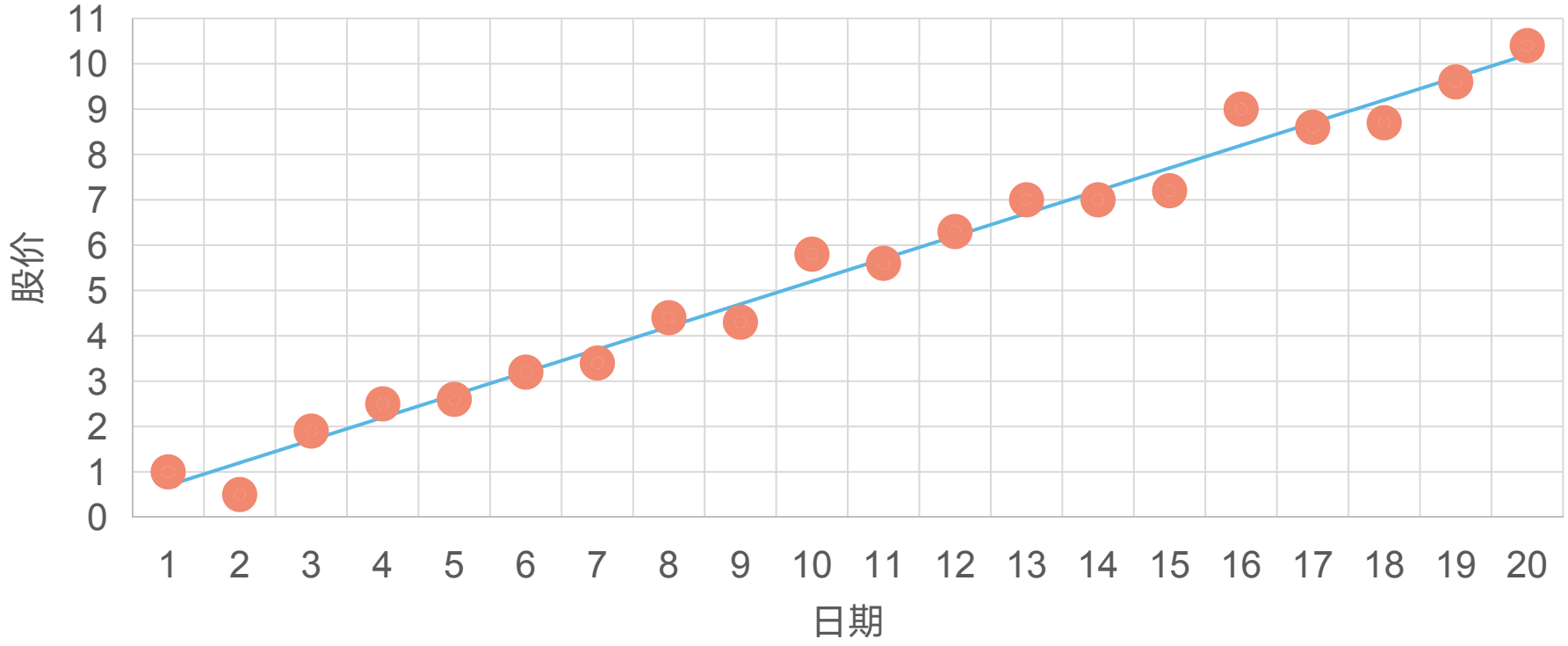
此外也可以化成

$$w = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

线性回归

将数据代入，可以求得$w = 0.5, b = 0.2$。

作直线$y = 0.5x + 0.2$与股票数据进行对比：



除了用最小二乘法还可以用**神经元**来解决这一问题

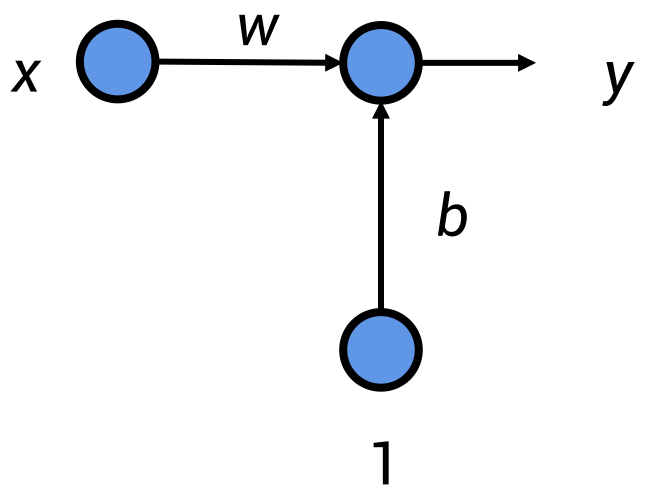
线性回归

设神经元模型为 $y = wx + b$

1）初始化权重 w，b

2）根据权重计算出一个解 $o_i = wx_i + b$

3）根据均方差求误差 $loss(w,b) = \frac{1}{2}(o_i - y_i)^2$



4）调整权值

$$\frac{\partial loss}{\partial w} = \frac{\partial loss}{\partial o_i}\frac{\partial o_i}{\partial w} = (o_i - y_i)\,x_i,\quad w_{new} = w_{old} - \eta\frac{\partial loss}{\partial w}$$

$$\frac{\partial loss}{\partial b} = \frac{\partial loss}{\partial o_i}\frac{\partial o_i}{\partial b} = (o_i - y_i),\quad b_{new} = b_{old} - \eta\frac{\partial loss}{\partial b}$$

5）是否满足终止条件？不满足则跳回（2）

线性回归

总结

- **最小二乘法**从损失函数求导，直接求得数学解析解；
- **神经元方法**是利用导数传递误差，再通过迭代方式一步一步（用近似解）逼近真实解。

多元线性回归

之前的问题中用于预测股票价格的输入为日期，可以认为特征的维度为1。

而有些时候每个样本有不止一个输入，也即有多个维度的特征，设特征的维度为d。

将其写作向量形式$\boldsymbol{x}_i = (x_{i1}, x_{i2}, ..., x_{id})^T$。

此时的线性回归可以写成：

$$\begin{aligned} h(\boldsymbol{x}_i) &= w_1 x_{i1} + w_2 x_{i2} + ... + w_d x_{id} + b \\ &= \boldsymbol{w}^T \boldsymbol{x}_i \end{aligned}$$

其中$\boldsymbol{w} = (w_1, w_2, ..., w_d)^T$

多元线性回归

此时，对于训练集$\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$，尝试习得

$$h(\boldsymbol{x}_i) = \boldsymbol{w}^T \boldsymbol{x}_i + b$$

使得

$$h(\boldsymbol{x}_i) \simeq y_i, i = 1,2, \ldots, n$$

则称为**多元线性回归**(multivariate linear regression)。

所求的$\boldsymbol{w}$又被称为特征的**权重**(weight)，b则称为**偏置**(bias)。

多元线性回归

类似地，使用最小二乘法来求解。

为书写方便，令

$$\hat{\boldsymbol{w}} = (\boldsymbol{w}; b)$$

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1^T & 1 \\ \boldsymbol{x}_2^T & 1 \\ \vdots & \vdots \\ \boldsymbol{x}_n^T & 1 \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1d} & 1 \\ x_{21} & x_{22} & \cdots & x_{2d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nd} & 1 \end{pmatrix}$$

$$\boldsymbol{y} = (y_1, y_2, \ldots, y_n)^T$$

也就是将偏置并入权重，并令其对应的输入恒为1，并将整个数据集合并为矩阵形式。

多元线性回归

此时的均方误差为

$$E = \frac{1}{n}\|\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}}\|^2$$

$\|\cdot\|$为欧几里得范数。

问题可以写成求

$$\begin{aligned} &\underset{\hat{\boldsymbol{w}}}{\operatorname{argmin}} \quad E \\ &= \underset{\hat{\boldsymbol{w}}}{\operatorname{argmin}} \, (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}})^T (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}}) \end{aligned}$$

多元线性回归

同样求导得到

$$\frac{\partial E}{\partial \hat{\boldsymbol{w}}} = 2\boldsymbol{X}^T(\boldsymbol{X}\hat{\boldsymbol{w}} - \boldsymbol{y})$$

令其为零可求得闭式解。

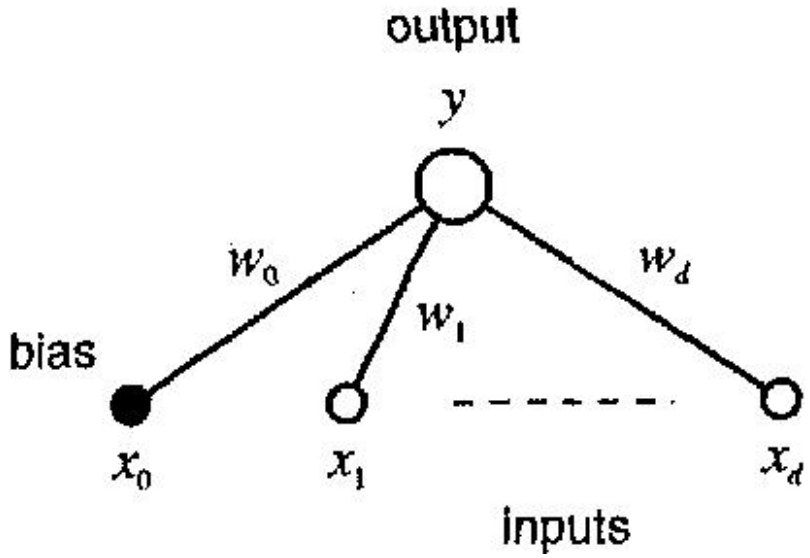
当$\boldsymbol{X}^T\boldsymbol{X}$满秩或正定时，解

$$\hat{\boldsymbol{w}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

若$\boldsymbol{X}^T\boldsymbol{X}$不满秩，例如当$d \geq n$时令$E$最小的结果不唯一，这里不作深入讨论。

多元线性回归

线性回归的线性方程也可以用如下神经元表示（输入神经元只负责传递数据）：



在神经元内只对输入进行加权求和，x_0和w_0表示线性方程中常数项。

这样神经元可用于解决多元线性回归问题。

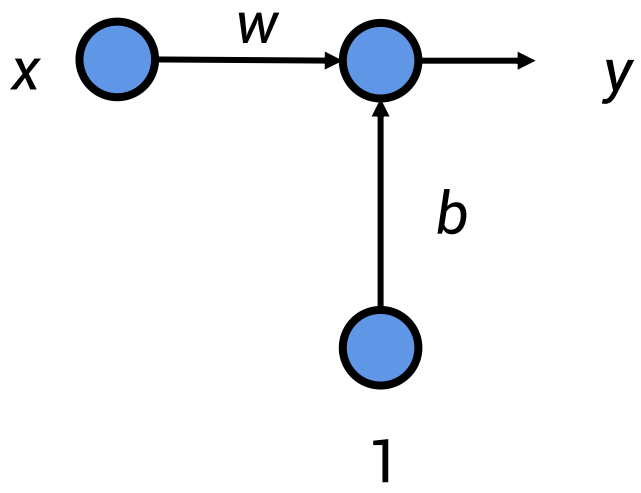
示例：线性回归预测股票价格

- 假如我们获得了股票20天的波动数据

股票20天波动数据					
日期	**1**	**2**	**3**	**4**	**5**
股价	55.22	56.34	55.52	55.53	56.94
日期	**6**	**7**	**8**	**9**	**10**
股价	58.88	58.18	57.09	58.38	38.54
日期	**11**	**12**	**13**	**14**	**15**
股价	57.72	58.02	57.81	58.71	60.84
日期	**16**	**17**	**18**	**19**	**20**
股价	61.08	61.74	62.16	60.80	60.87

单输入单输出

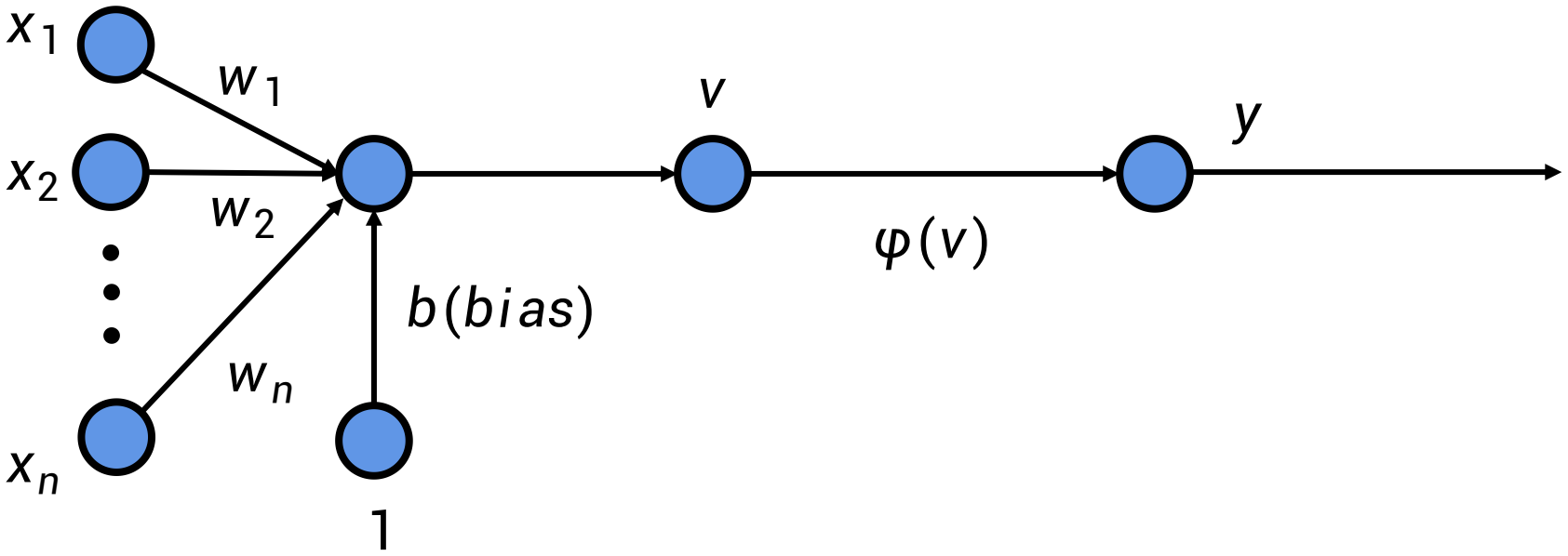
- 采用单输入的神经元进行拟合，模型$y = wx + b$
- 线性拟合的结果是$y = 0.3249x + 55.1073$



多输入单输出

- 若假设股票波动不仅与当前日期有关，还与多个前面多个日期相关
- 采用多输入的神经元进行拟合，预测模型为

$$y = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b$$



多输入单输出

- 采用多元线性回归可以求得：

  $w_1 = -0.2064,\ w_2 = 0.5659, w_3 = -0.5767, w_4 = 1.0369,\ b = 10.9260$

- 检测该模型的预测结果，第5天56.9746（预测值）\ 56.94（准确值），第10天59.3102 \ 58.54，第13天58.8806 \ 57.81，第20天60.4564 \ 60.87

- 相比单元预测模型的预测结果为 56.7318、58.3563、59.3310、61.6053，可以看出多元线性模型比单元线性模型的预测更为准确

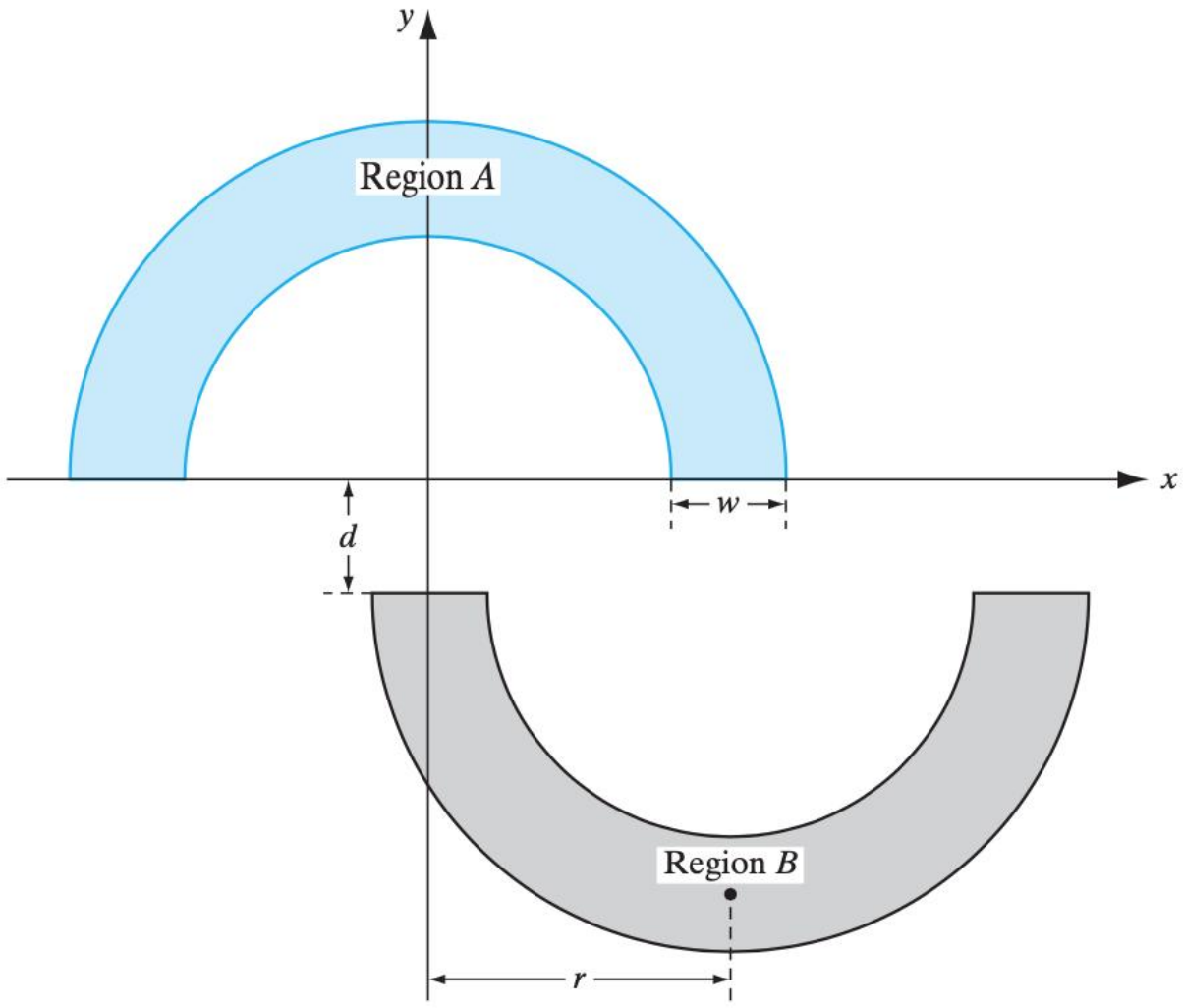
线性分类

以双月模型为例：

有两个“月亮”，标记为“区域A”的月亮关于y轴对称，标记为“区域B”的月亮在离y轴距离为r，离x轴距离为d的位置。

两个月亮大小相同，其中：半径为r，宽度为w。

- d的值越大两个月亮会更加分离
- d的值越小两个月亮会更加靠近



双月模型

- **应用：**

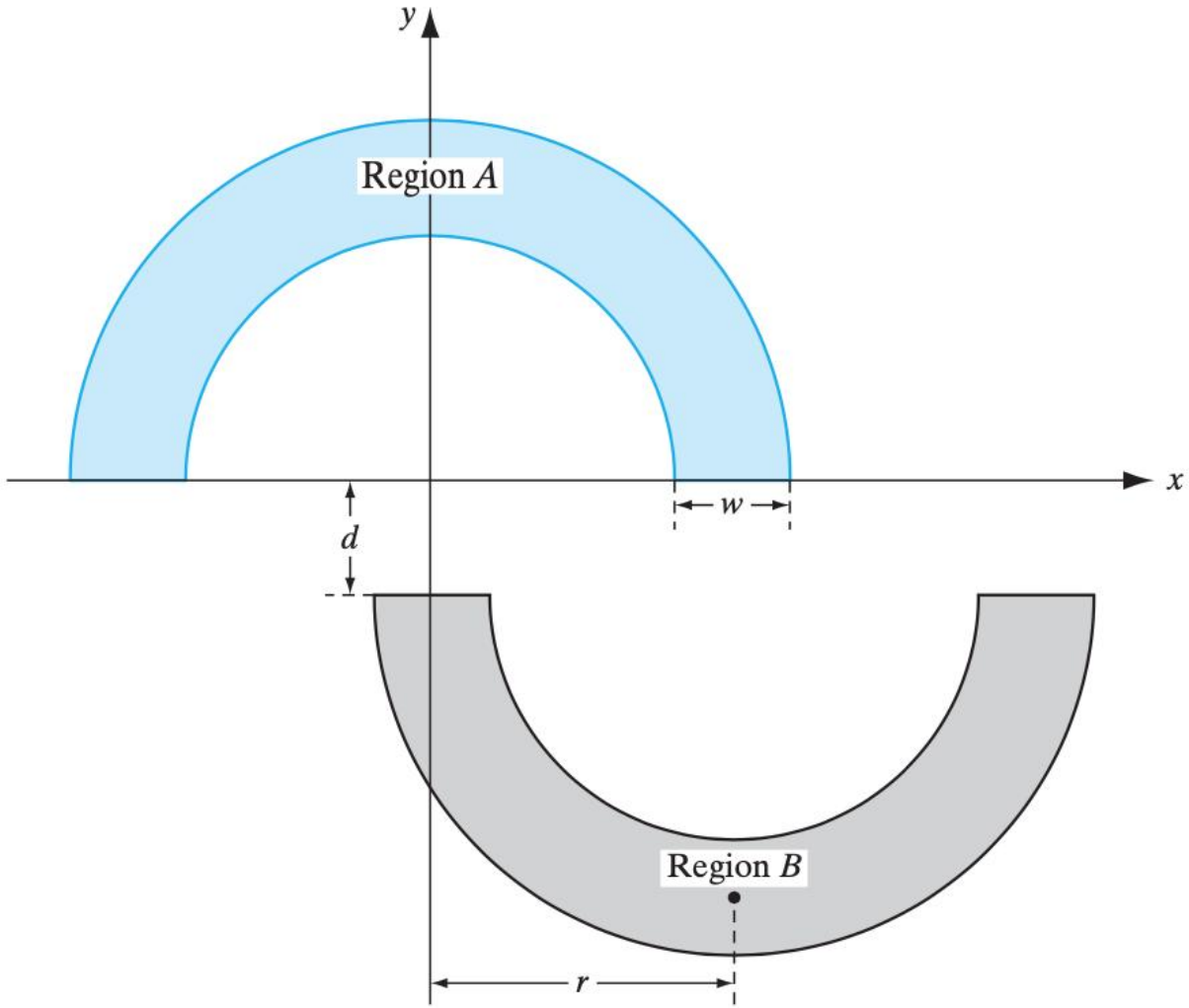
感知机神经元也是一种多输入单输出的神经元，它用于将这两个月亮分离。

- **数据：**

训练样本由 2000 个数据点组成，随机从区域A、B中各自选取1000个点。

测试样本由 4000个数据点组成，随机从区域A、B中各自选取2000个点。

其中标签 $y_i = \begin{cases} +1, & \text{点} \boldsymbol{x}_i \in \text{区域} A \\ -1, & \text{点} \boldsymbol{x}_i \in \text{区域} B \end{cases}$



双月模型

（1）对$\boldsymbol{w}_0$，b_0 进行初始化。

```
def __init__(self, shape):
    self.w = np.zeros(shape)      # 权重
    self.b = 0                    # 偏置
    self.activate_func = sgn      # 激活函数
```

（2）在训练集中选取数据（$\boldsymbol{x}_i$ ， y_i）计算输出。

```
def calclate(self, x):
    return self.activate_func(np.dot(self.w, x.T) + self.b)
```

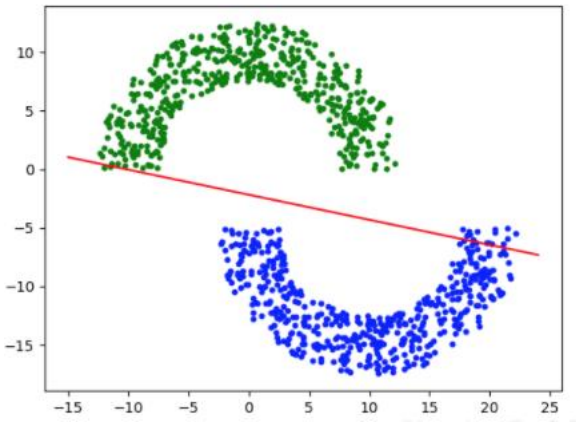
（3）这里使用基于误差的学习规则来更新权值。

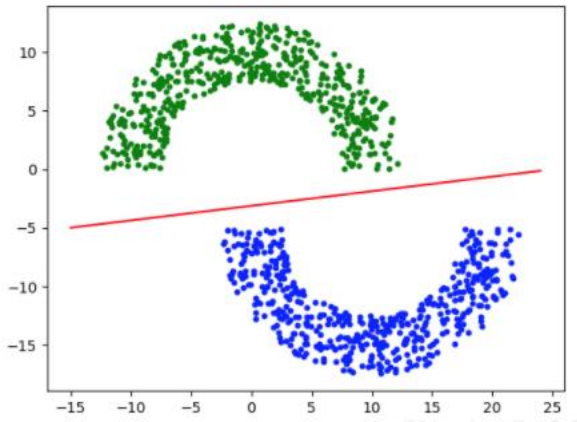
```
def update(self,x,y,out,learning_rate):
    self.w += learning_rate * x.T * (y - out)
    self.b += learning_rate * (y - out)
```

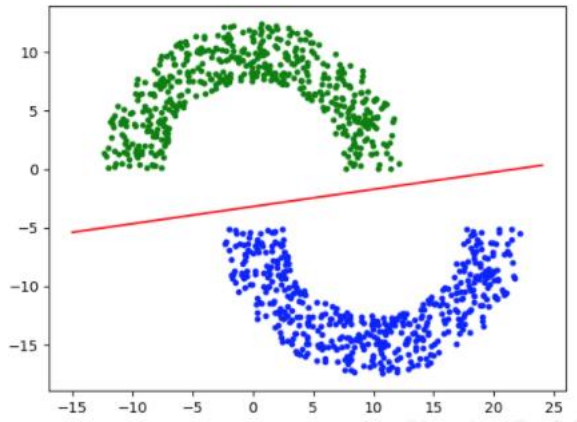
双月模型

（4）循环。

```
def train(self, x, y, epochs, learning_rate):
    for epoch in range(epochs):
        for i in range(x.shape[0]):
            out = self.calclate(x[i])
            self.update(x[i], y[i], out, learning_rate)
```





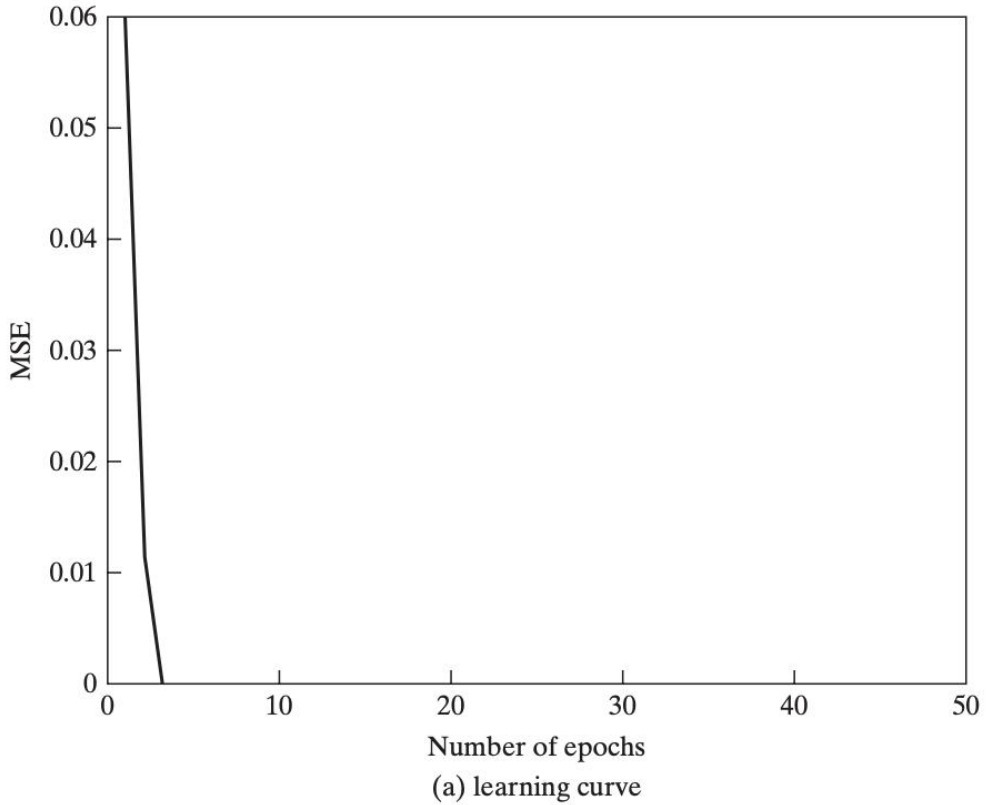


双月模型

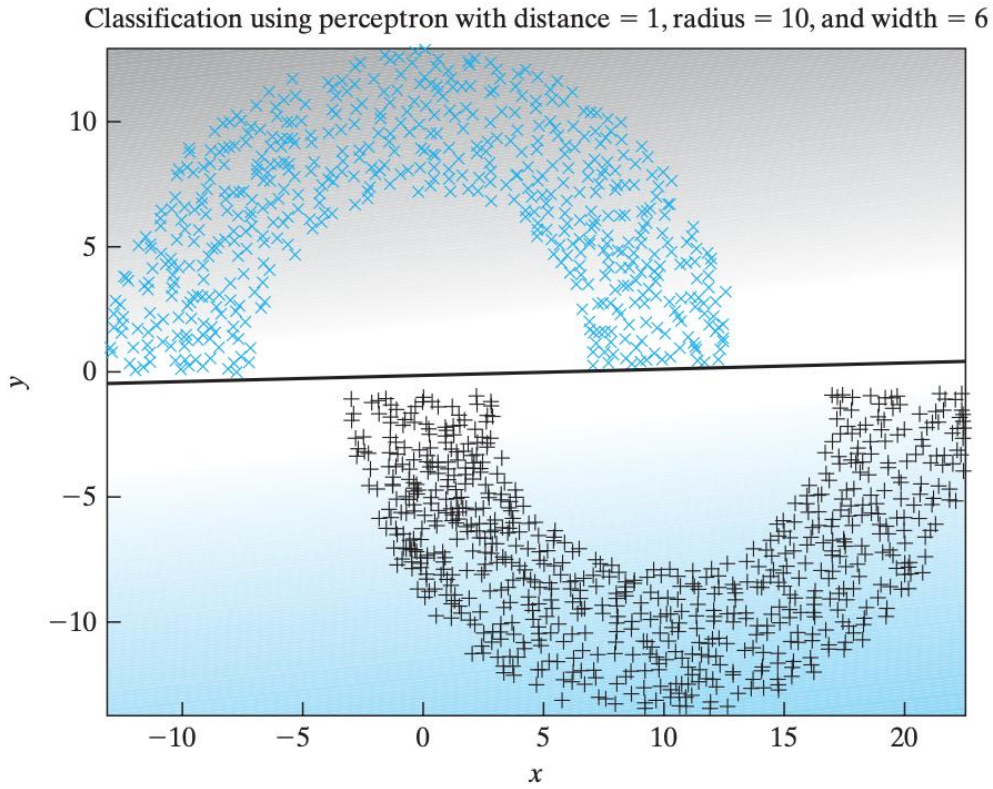
当 r=10，w=6 且 **d=1** 时：

图（a）是学习曲线，描述了均方误差MSE和迭代次数之间的关系，三步迭代就趋于收敛了。

图（b）是感知机训练后计算得到的决策边界，能够较好的将测试点分离。



(a) learning curve



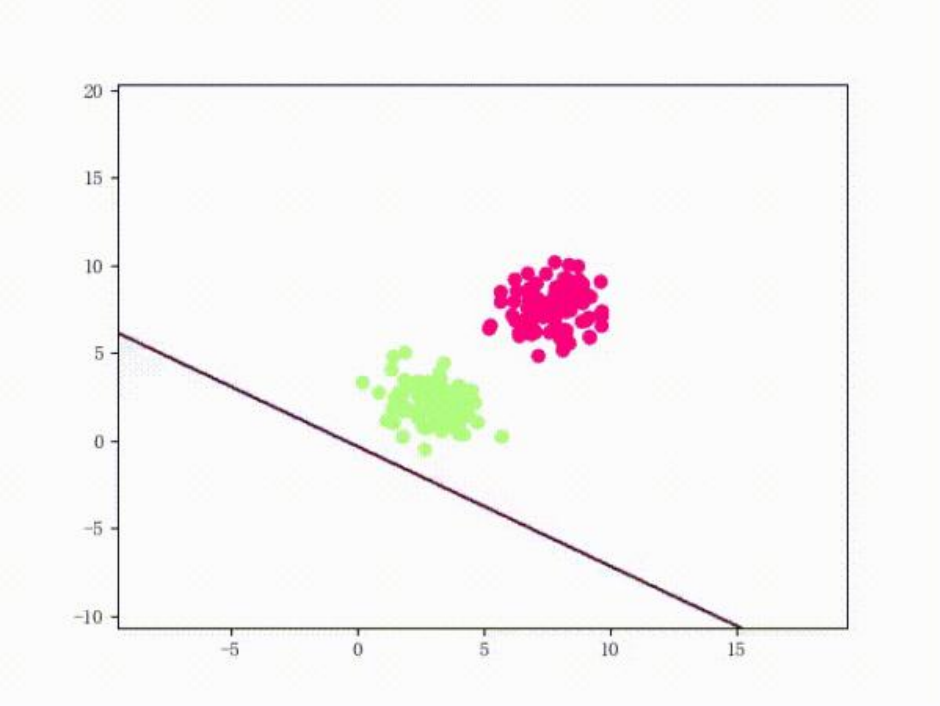
(b) testing result

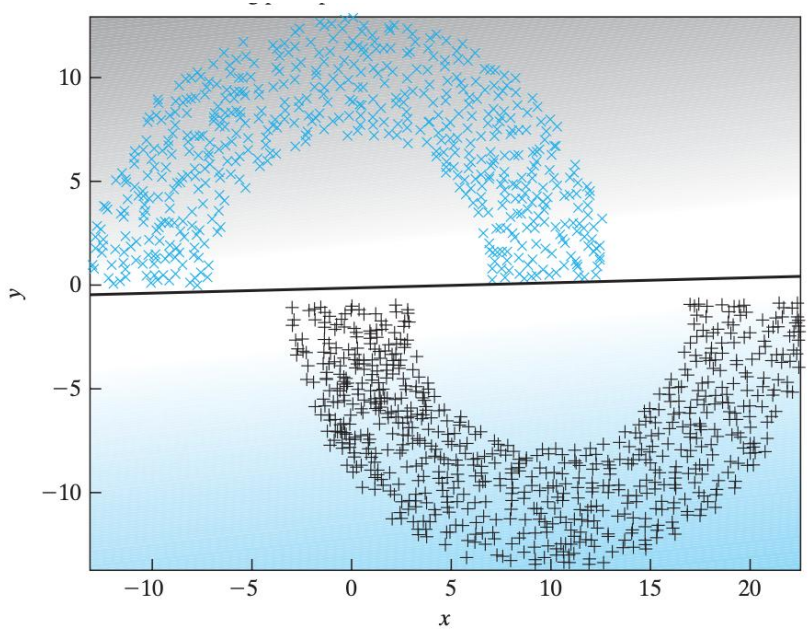
单个神经元的局限

线性可分

由$\boldsymbol{w}^T\boldsymbol{x}$=0决定的超平面将不同的类别分开，这样将不同类别分开的界限便被称为**决策边界**(decision boundary)，而能够由线性的决策边界进行划分的分类问题则被称为**线性可分**(linear separable)。

单个神经元就可以较好处理线性可分问题。



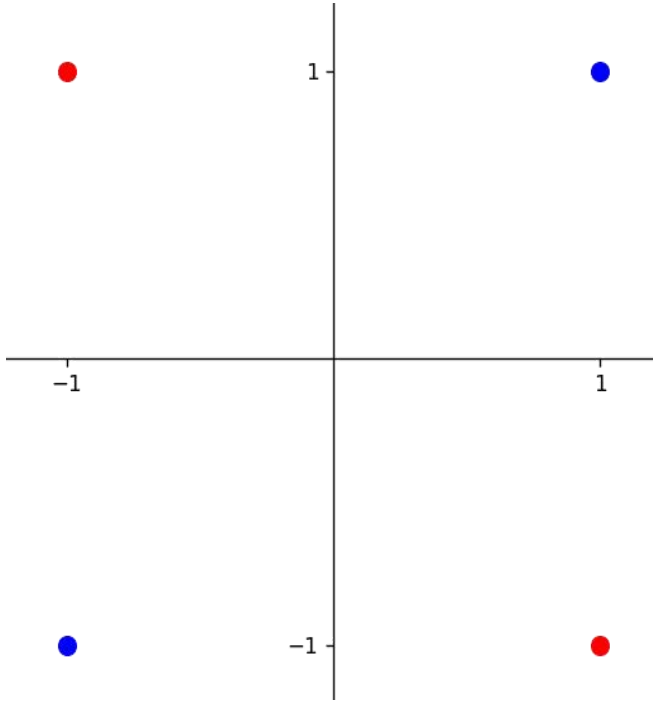


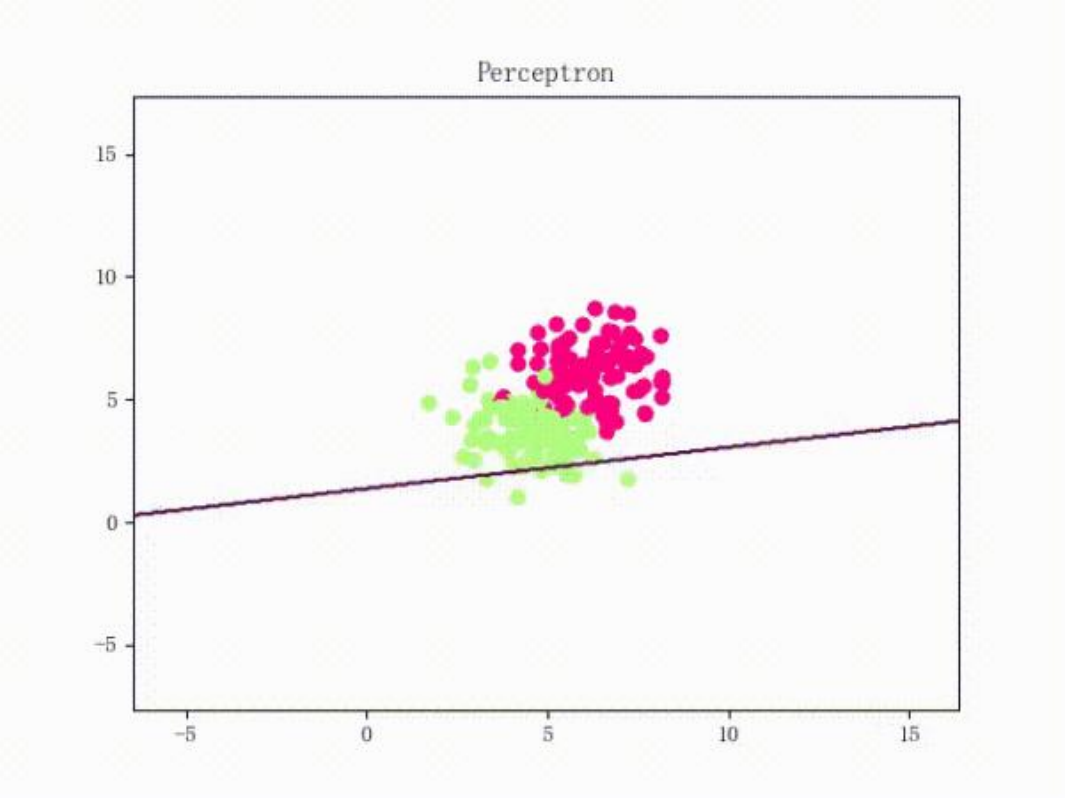
单个神经元的局限

线性不可分

而现实中的很多分类问题是**线性不可分**的，如XOR问题。

如果令+1为真， −1为假，则XOR问题如图所示。其中红色为真类，蓝色为假类。显然，无法找到一条直线划分两个类别。



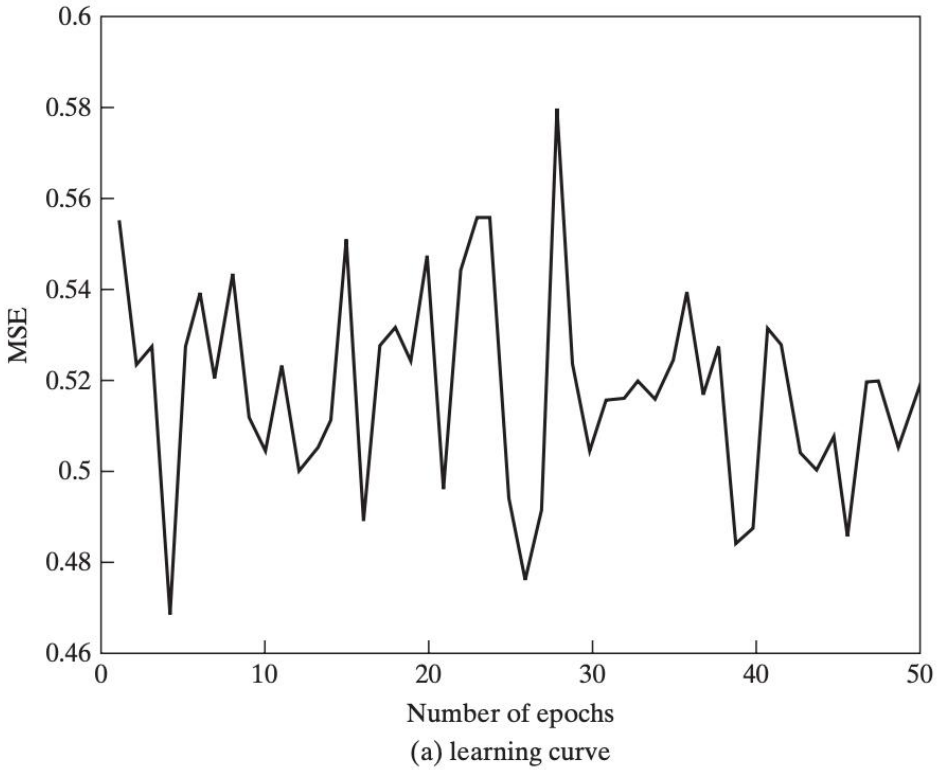


单个神经元的局限

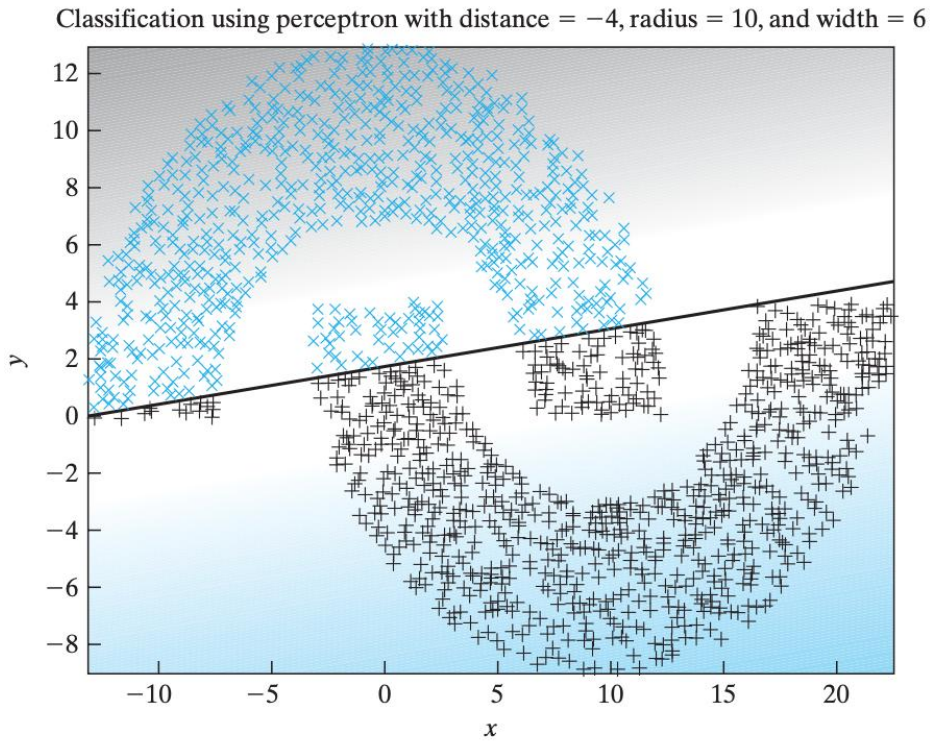
线性不可分

在双月模型中如果依然设置 r=10，w=6 ，但设置 **d=-4**，会得到下面的结果。

图（a）中的学习曲线持续波动，图（b）中的决策边界也不能很好的划分两个月亮



(a) learning curve



(b) testing result

05 思考題

- 怎样用 MP 神经元实现与、或、非逻辑运算？（分析 假设将逻辑运算表示为 x1&x2 == y，也就是将 (x1, x2) 视为输入，y 视为输出分类。我们希望MP神经元获得从 (x1, x2) 到 y 的映射关系。）
- 为什么要向神经元中引入激活函数，请再列举至少三种书中未介绍的激活函数，并给出其表达式。
- 异或问题是否可以通过单个感知机神经元实现？为什么？
- 除异或问题外，还有哪些问题直观上非常简单但使用单个感知机神经元无法解决，请给出一个实例并说明无法解决的原因。
- 尝试通过组合多个感知机神经元来解决异或问题，请画出所设计的网络结构（包括相关联结的权重）

Q&A

Questions and Answers