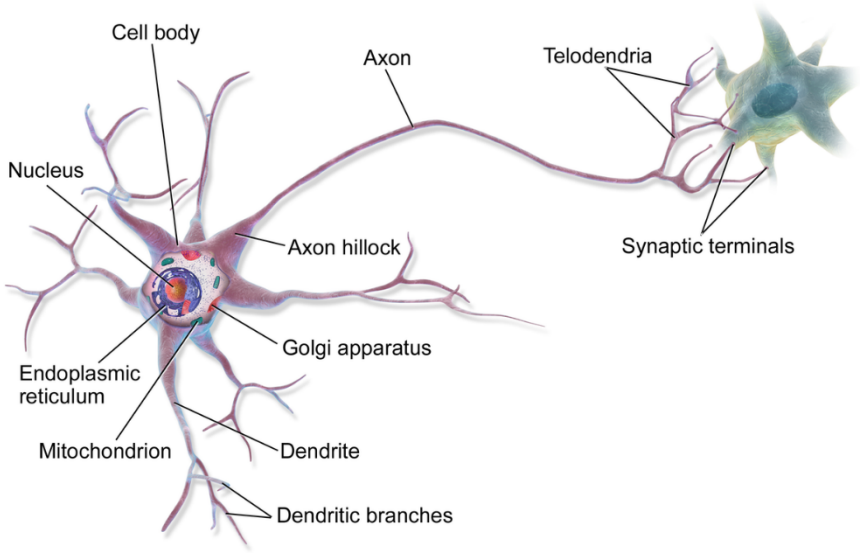
神经网络基础

主讲人：屠恩美

《机器学习与知识发现》

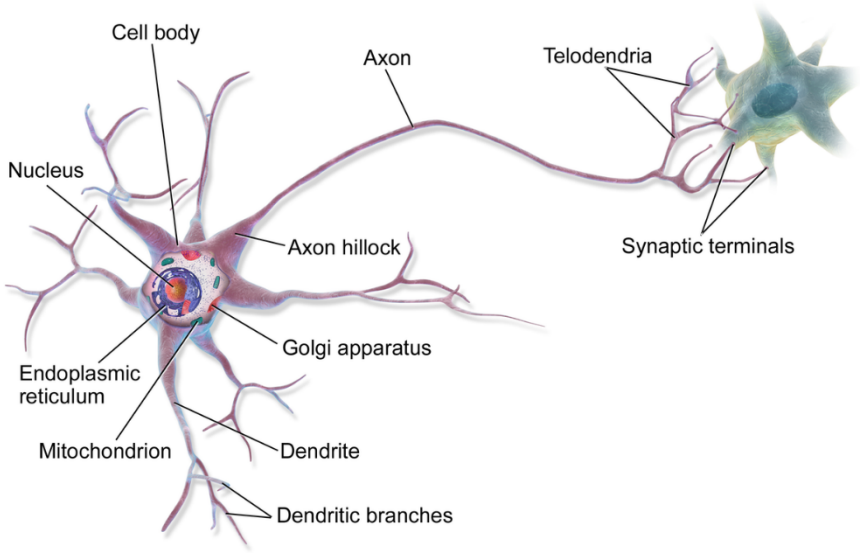
上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

神经元模型 – 生物与人工

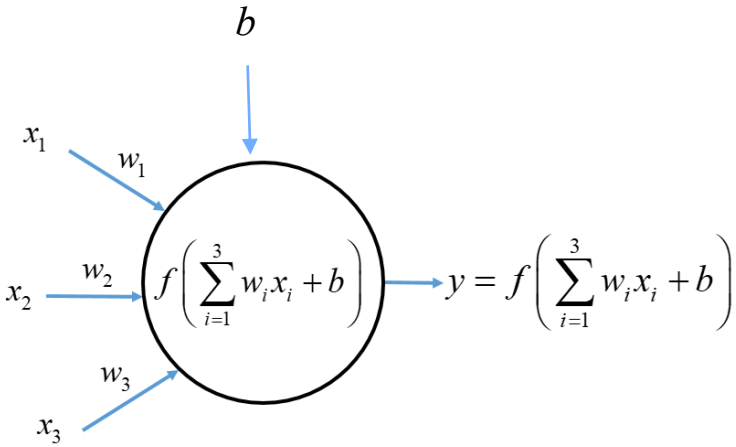


生物神经元结构

神经元模型 – 生物与人工



生物神经元结构



人工神经元模型

神经元模型 – 五要素

神经元五要素：

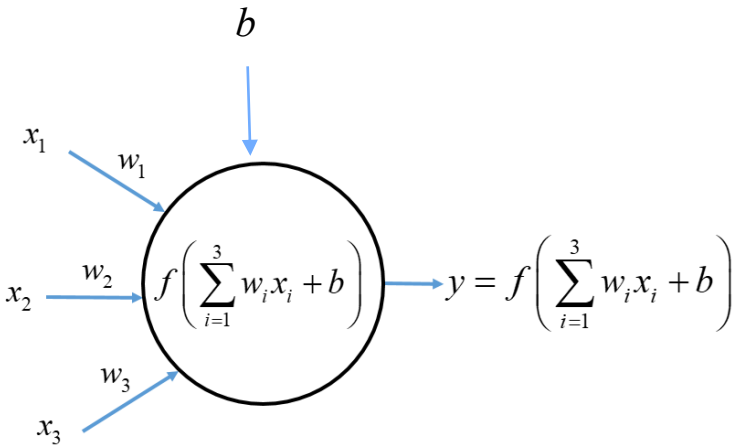
输入向量 $\mathbf{x}=(x_1,x_2,x_3)$

权值向量 $\mathbf{w}=(w_1,w_2,w_3)$

偏移标量 b

激活函数 $f(\bullet)$

输出目标 y



人工神经元模型

神经元模型 – 五要素

神经元五要素：

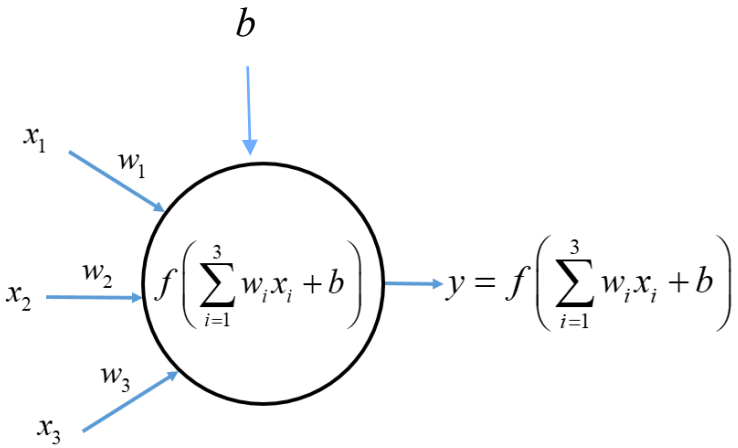
输入向量 $\mathbf{x}=(x_1,x_2,x_3)$ ， （已知）

权值向量 $\mathbf{w}=(w_1,w_2,w_3)$， （未知）

偏移标量 b ， （未知）

激活函数 $f(\bullet)$ ， （已知）

输出目标 y ， （已知）



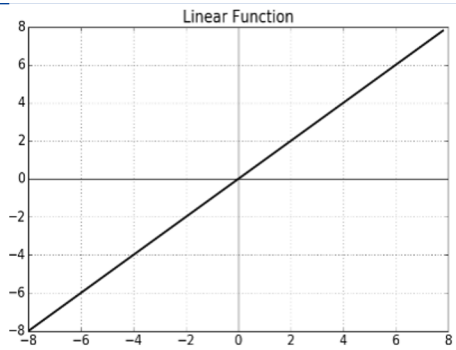
人工神经元模型

设计激活函数，学习未知变量

神经元模型 – 激活函数

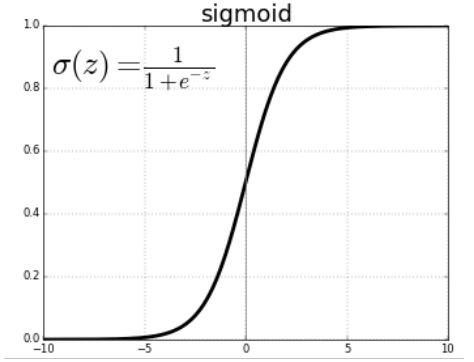
线性激活函数

$$f(s)=s$$



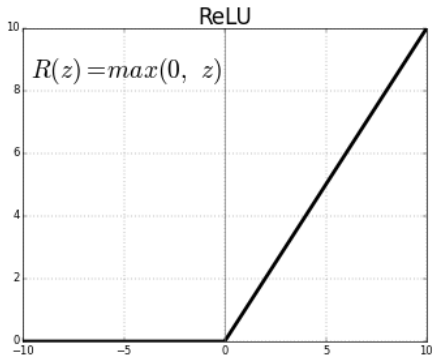
Sigmoid激活函数

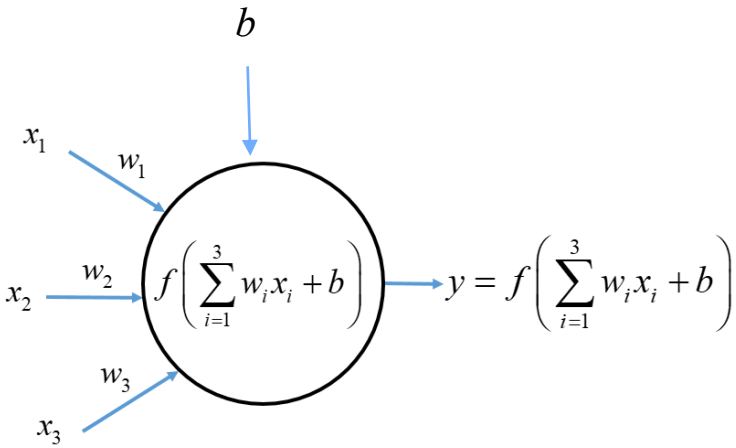
$$f(s)=\frac{1}{1+e^{-s}}=\frac{e^{s}}{1+e^{s}}$$



整流函数(ReLU)

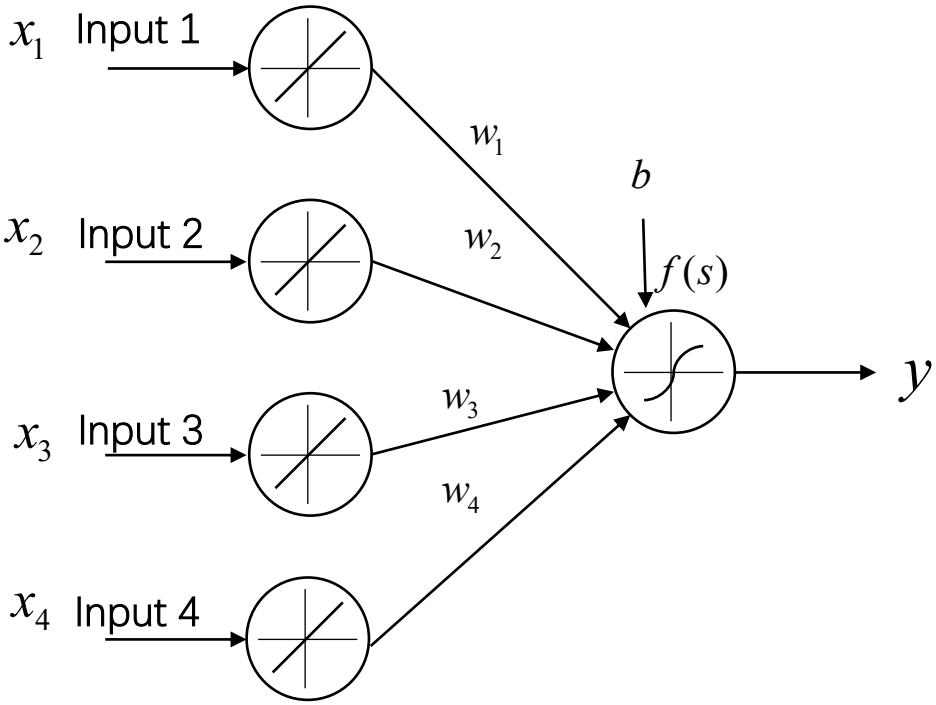
$$f(s)=\max(0,s)$$





人工神经元模型

两层神经网络



输入样本

$\mathbf{x} = (x_1, x_2, x_3, x_4)$

$$s = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$$

$$= \sum_{i=1}^{4} w_i x_i + b = \mathbf{w}^T\mathbf{x} + b$$

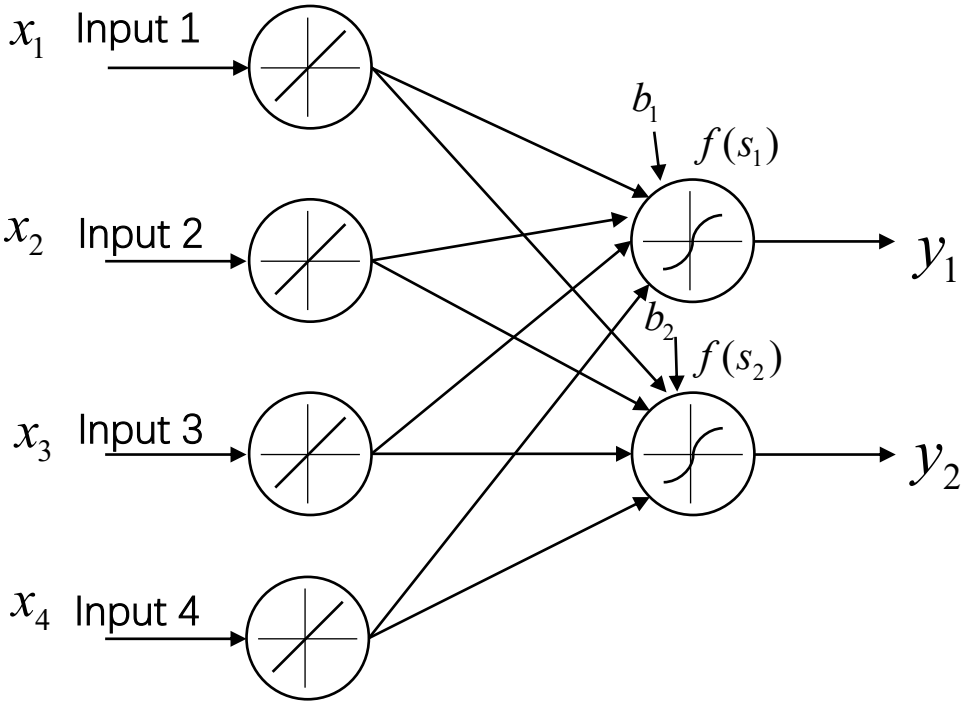
权值向量　　输入样本

$$y = f(s) = f(\mathbf{w}^T\mathbf{x} + b)$$

两层神经网络

输入样本

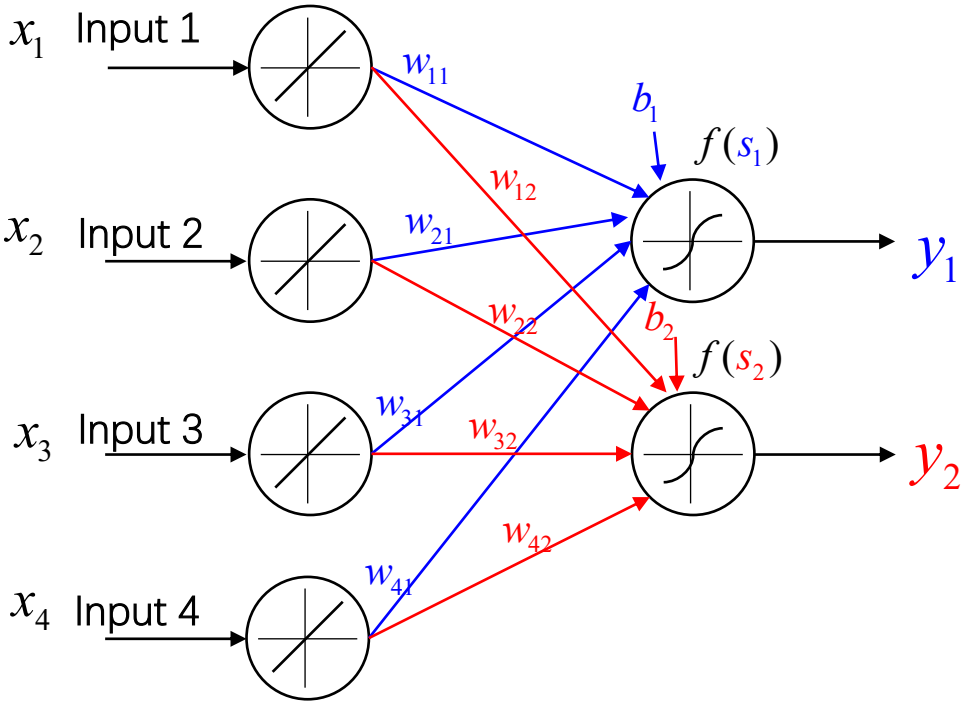
$\mathbf{x} = (x_1, x_2, x_3, x_4)$



两层神经网络

输入样本

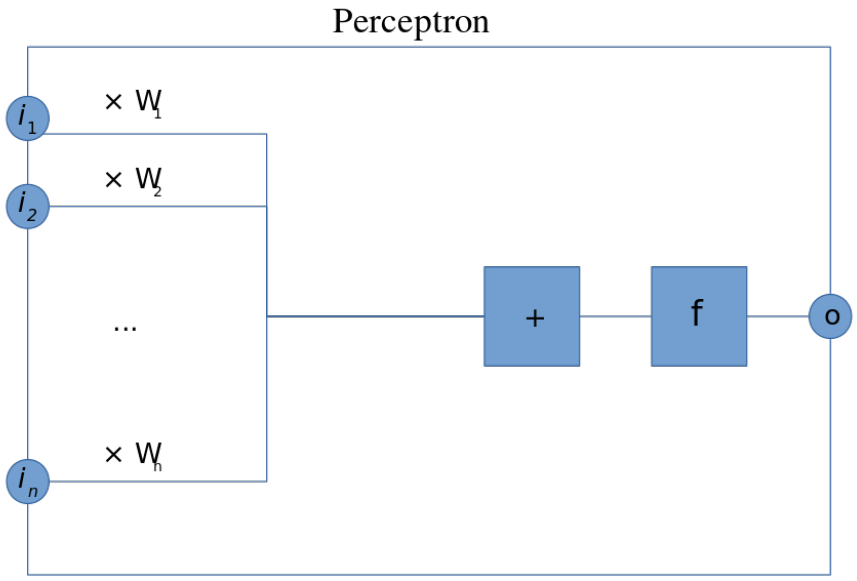
$\mathbf{x} = (x_1, x_2, x_3, x_4)$



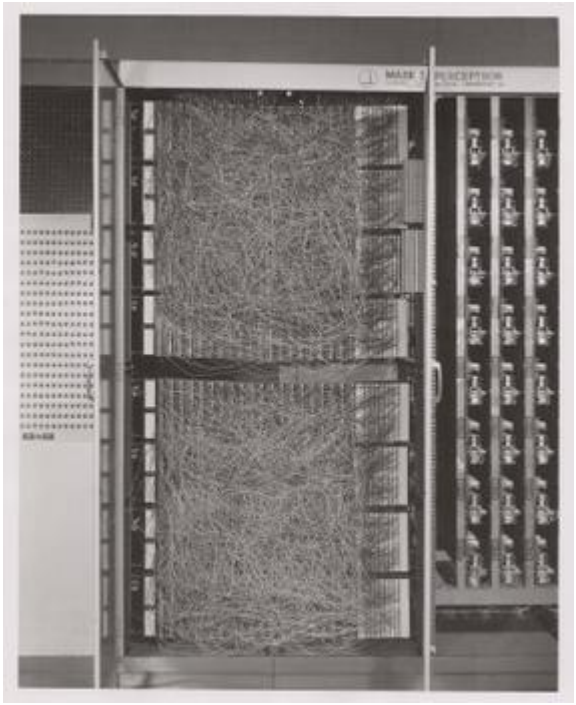
$$s_1 = \sum_{i=1}^{4} w_{i1} x_i + b_1 = \mathbf{w}_1^T \mathbf{x} + b_1 \qquad y_1 = f(s_1) = f(\mathbf{w}_1^T \mathbf{x} + b_1)$$

$$s_2 = \sum_{i=1}^{4} w_{i2} x_i + b_2 = \mathbf{w}_2^T \mathbf{x} + b_2 \qquad y_2 = f(s_2) = f(\mathbf{w}_2^T \mathbf{x} + b_2)$$

感知器Perceptron



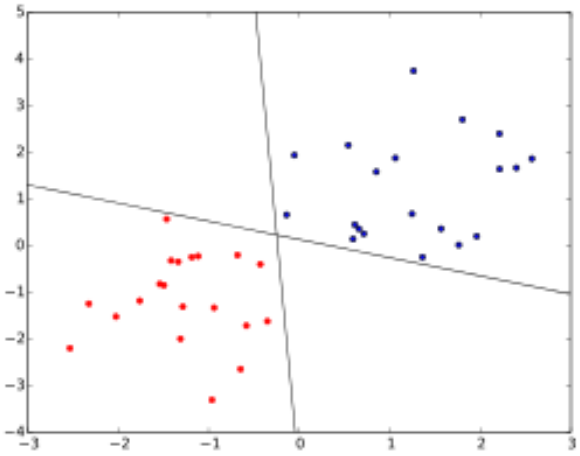
$$o = f\left(\sum_{k=1}^{n} i_k \cdot W_k\right)$$

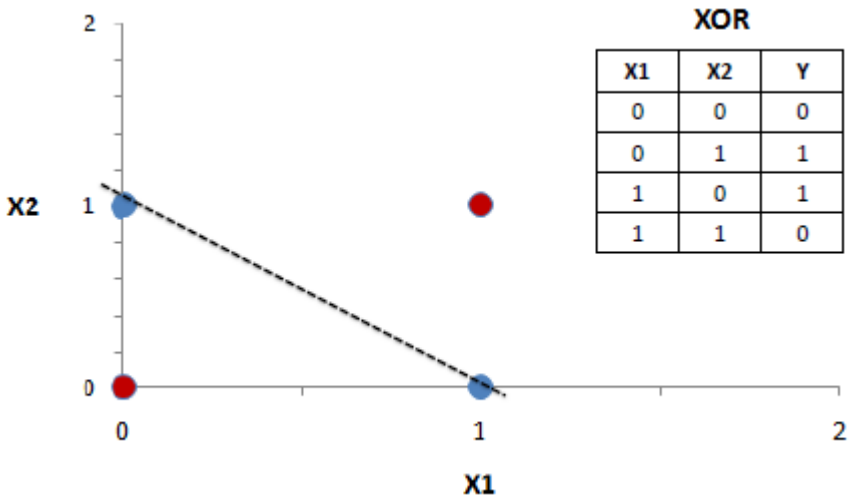


$$f(x) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

Frank Rosenblatt(1928.7.11 –1971.7.11)

感知器Perceptron





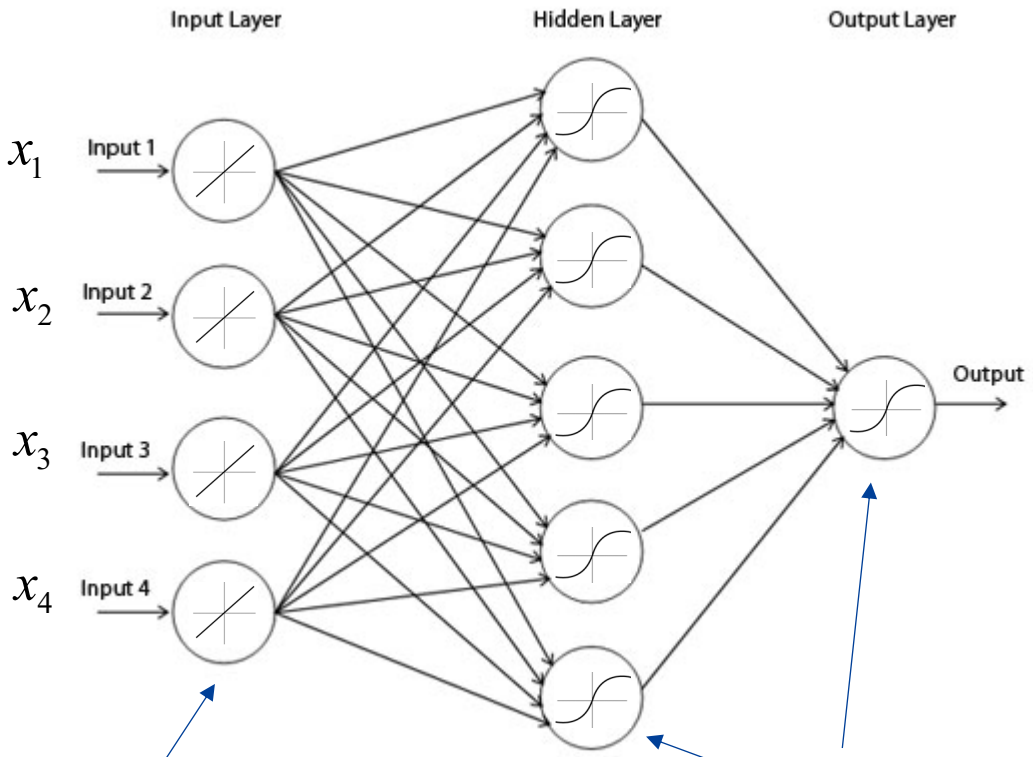
XOR

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0

多层神经网络 – 基本结构

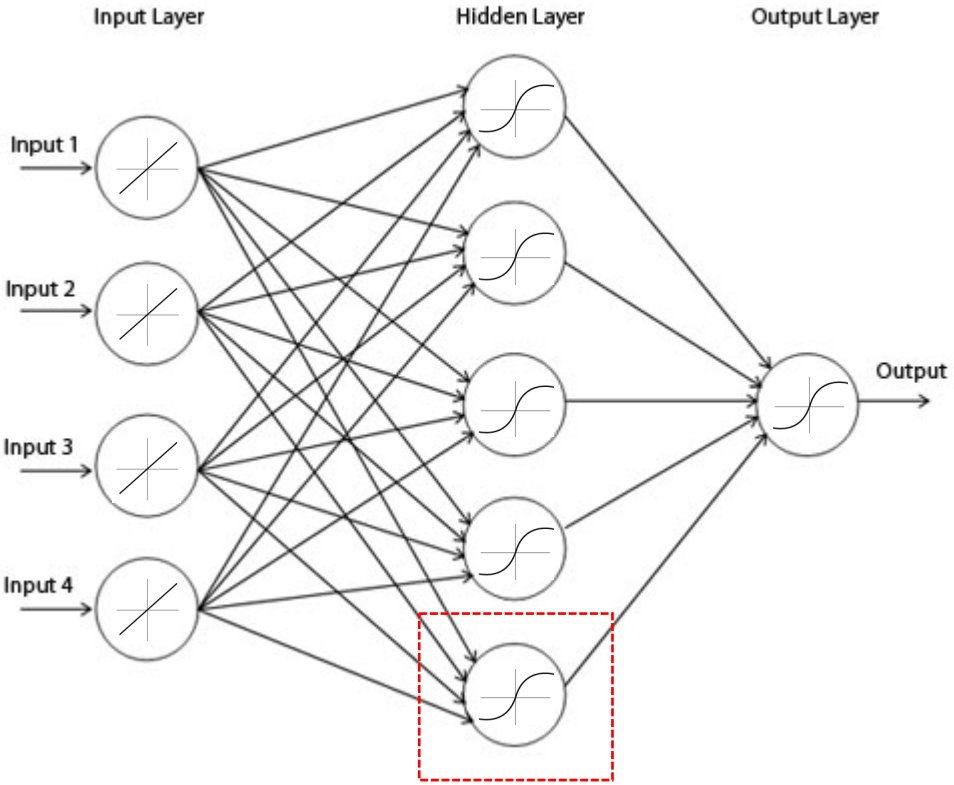
输入样本

$\mathbf{x} = (x_1, x_2, x_3, x_4)$



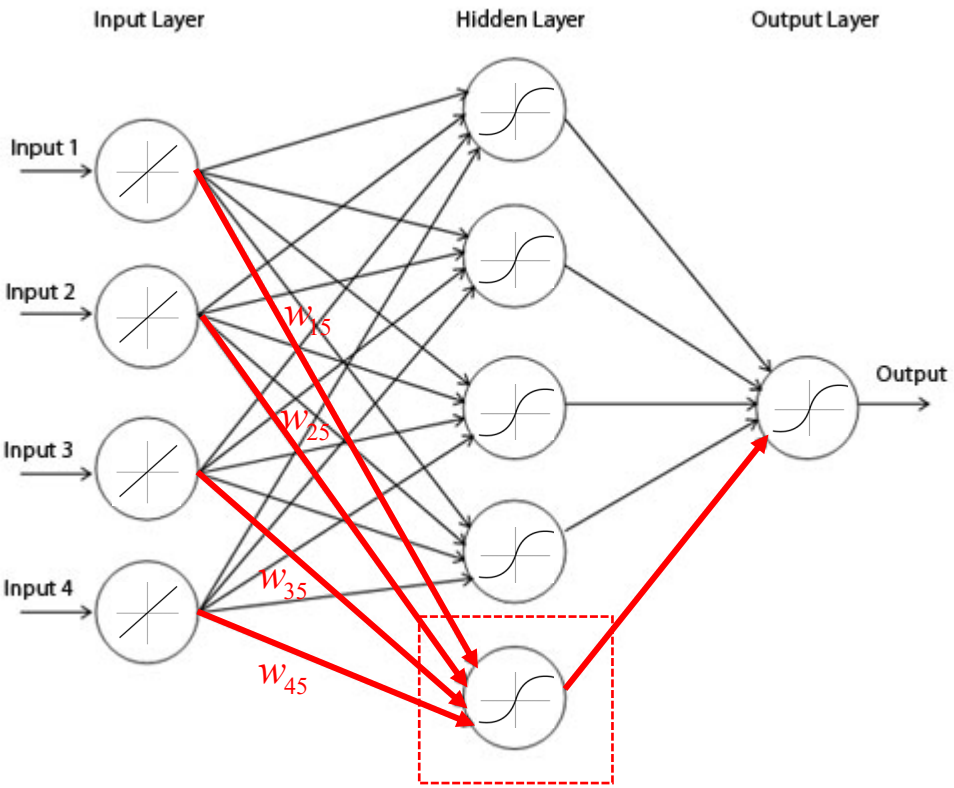
习惯上隐去每个神经元的偏移量 b

多层神经网络 – 输入输出关系



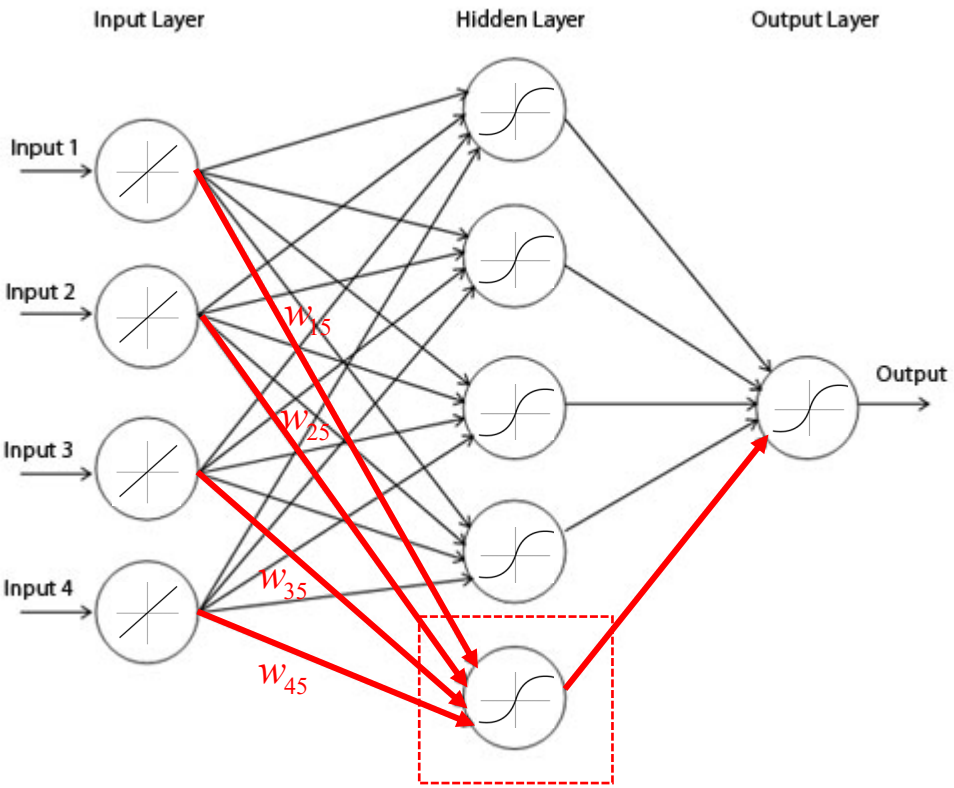
该神经元的输入、输出分别是什么？

多层神经网络 – 输入输出关系



输入 = （input 1, input 2, input 3, input 4）

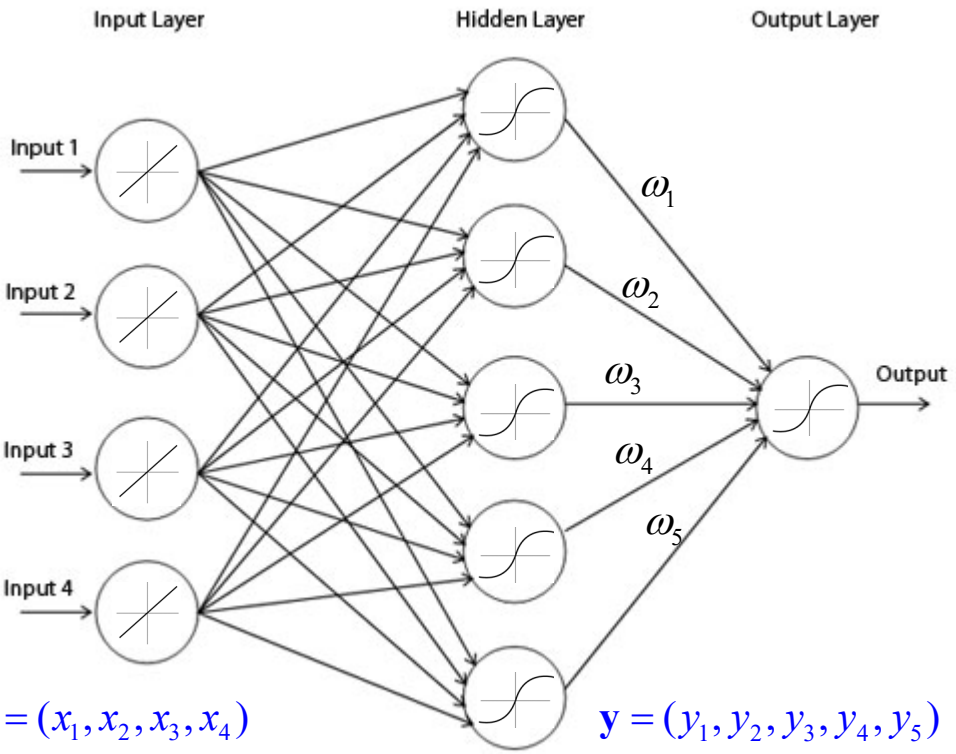
多层神经网络 – 输入输出关系



输入 = (input 1, input 2, input 3, input 4)

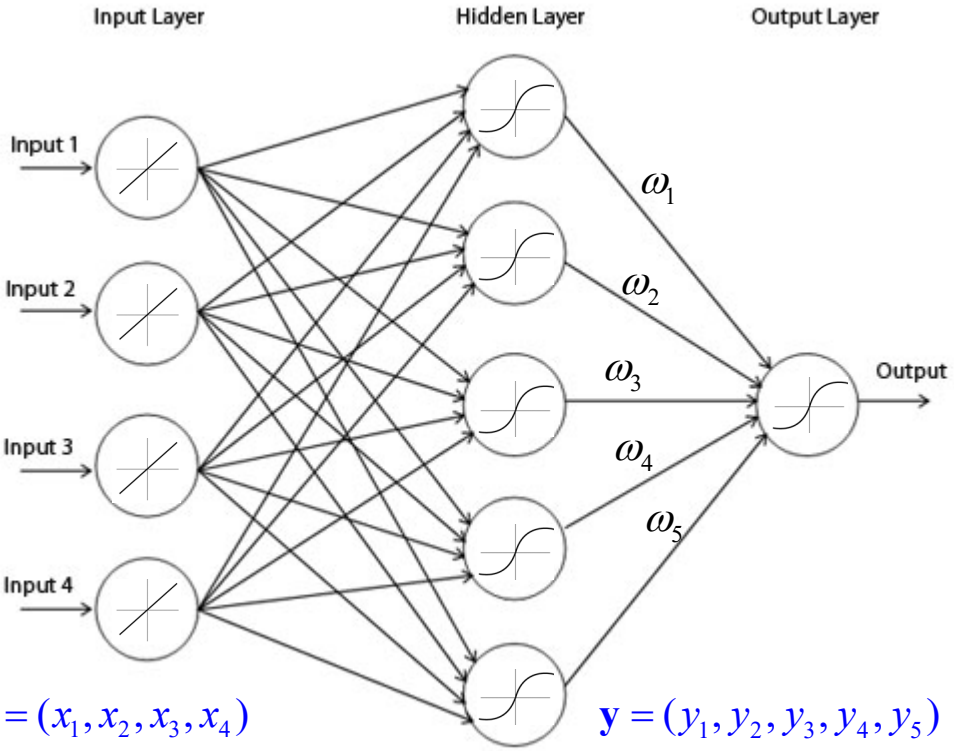
输出 = f (w_{15}*input 1 + w_{25}*input 2 + w_{35}*input 3 + w_{45}*input 4 + b)

多层神经网络 – 输入输出关系



总的输出 $\hat{z} = f\left(\sum_{i=1}^{5} \omega_i y_i + b_o\right)$

多层神经网络 – 输入输出关系



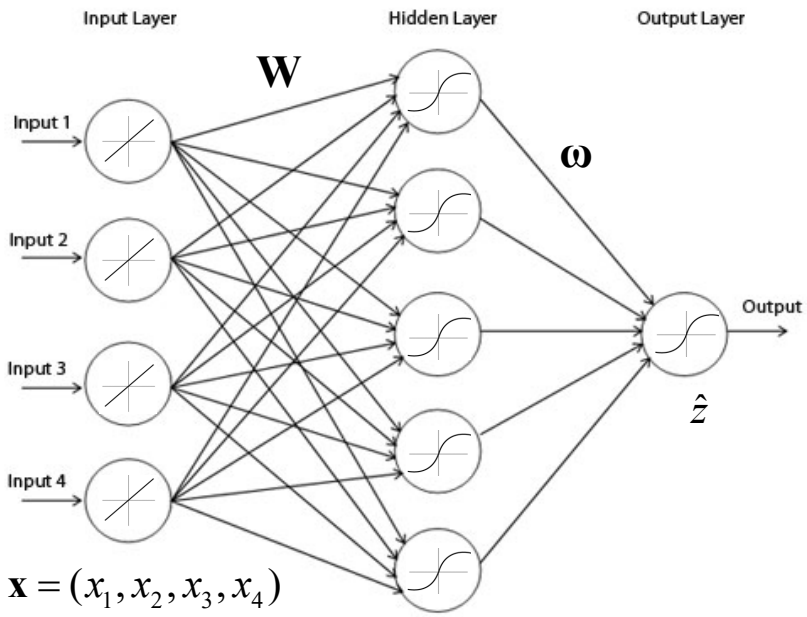
总的输出 $\hat{z} = f\left(\sum_{i=1}^{5} \omega_i y_i + b_o\right)$，其中隐含层的输出为 $y_i = f\left(\sum_{k=1}^{4} w_{ki} x_k + b_i\right), i = 1..5$

多层神经网络 – 输入输出关系

- 写成向量、矩阵形式

$$\begin{cases} \hat{z} = f\left(\sum_{i=1}^{5} \omega_i y_i + b_o\right) \\ y_i = f\left(\sum_{j=1}^{4} w_{ji} x_j + b_i\right), i = 1..5 \end{cases}$$

$$\boldsymbol{\omega} = (\omega_1, \omega_2, ..., \omega_4)^T$$
$$\mathbf{y} = (y_1, y_2, ..., y_5)^T$$
$$\mathbf{b} = (b_1, b_2, ..., b_5)^T$$
$$\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_5)$$



$$\begin{cases} \hat{z} = f\left(\boldsymbol{\omega}^T \mathbf{y} + b_o\right) & \text{网络输出} \\ \mathbf{y} = f\left(\mathbf{W}^T \mathbf{x} + \mathbf{b}\right) & \text{隐含层输出 (函数} f \text{元素运算函数)} \end{cases} \quad \begin{cases} \mathbf{y} = f(\mathbf{x}) \\ y_i = f(x_i) \end{cases}$$

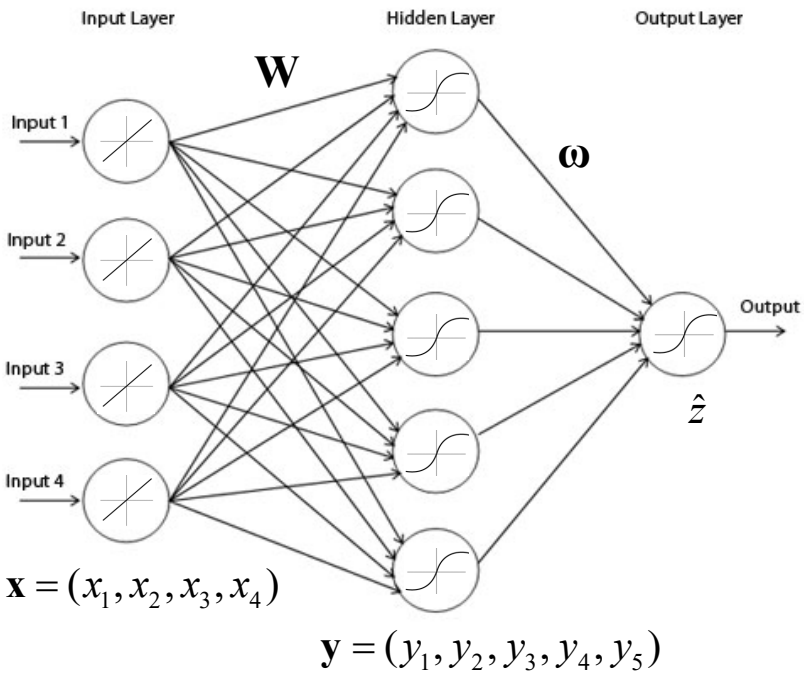
多层神经网络 – 学习目标

- 学习目标：网络对每个样本 $\mathbf{x}$ 的输出 $\hat{z}$ 尽可能与其对应的目标值 z 相同。
- 构建学习目标函数（误差函数）

$$J(\mathbf{x};z)=(\hat{z}-z)^2$$

其中

$$\begin{cases}\hat{z}=f\left(\sum_{i=1}^{5}\omega_i y_i+b_o\right) & \text{网络输出}\\ y_i=f\left(\sum_{j=1}^{4}w_{ji}x_j+b_i\right), i=1..5 & \text{隐含层输出}\end{cases}$$

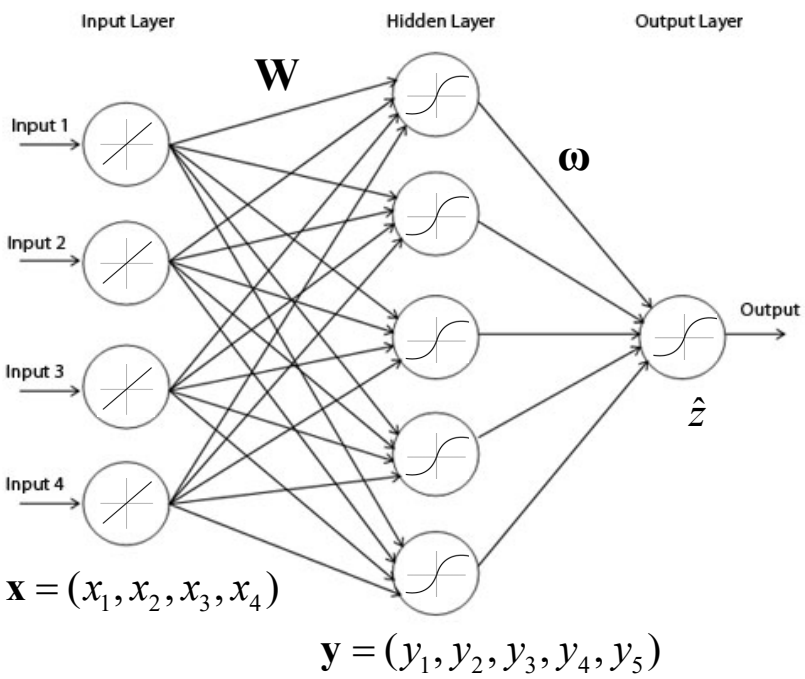


多层神经网络 – 学习目标

- 问题归纳：给定训练数据集 $D=\left\{\left(\mathbf{x}_i,z_i\right)\right\}_{i=1}^{n}$

$$\begin{cases} J(\mathbf{x};z)=\dfrac{1}{2}\left(\hat{z}-z\right)^2 & \text{误差函数} \\ \hat{z}=f\left(\boldsymbol{\omega}^T\mathbf{y}+b_o\right) & \text{网络输出} \\ \mathbf{y}=f\left(\mathbf{W}^T\mathbf{x}+\mathbf{b}\right) & \text{隐含层输出} \end{cases}$$

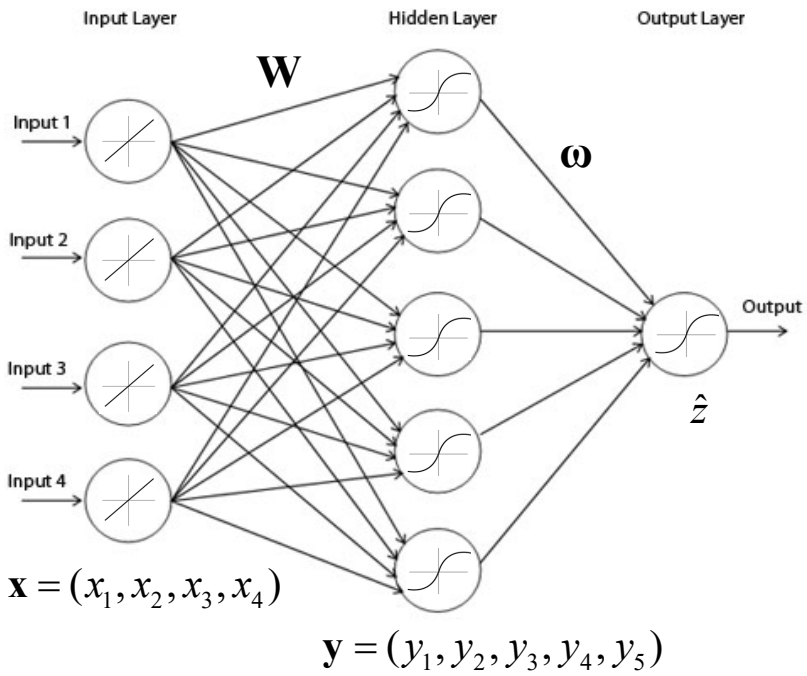
- 要学习的变量是哪些？



多层神经网络 – 学习目标

- 问题归纳：给定训练数据集 $D=\{(\mathbf{x}_i,z_i)\}_{i=1}^{n}$

$$\begin{cases} J(\mathbf{x};z)=\dfrac{1}{2}(\hat{z}-z)^2 & \text{误差函数} \\ \hat{z}=f\left(\boldsymbol{\omega}^T\mathbf{y}+b_o\right) & \text{网络输出} \\ \mathbf{y}=f\left(\mathbf{W}^T\mathbf{x}+\mathbf{b}\right) & \text{隐含层输出} \end{cases}$$



- 要学习的变量是哪些？

  输出层权值 $\boldsymbol{\omega}$， 输出层偏移 b_o，

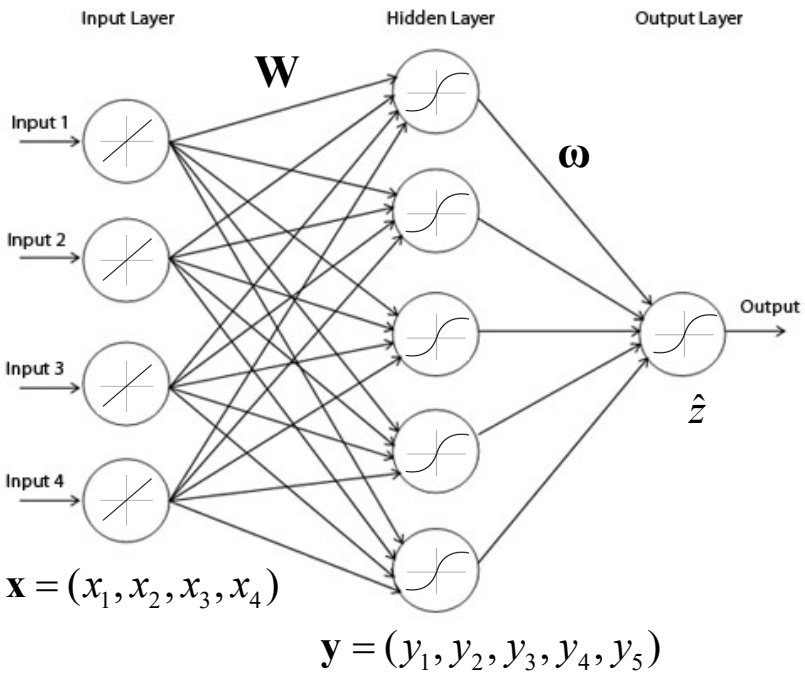
  隐含层权值 $\mathbf{W}$， 隐含层偏移 $\mathbf{b}$

- 怎么学习？

多层神经网络 – 学习目标

- 问题归纳：给定训练数据集 $D=\{(\mathbf{x}_i,z_i)\}_{i=1}^{n}$

$$\begin{cases} J(\mathbf{x};z)=\dfrac{1}{2}(\hat{z}-z)^2 & \text{误差函数} \\ \hat{z}=f\left(\boldsymbol{\omega}^T\mathbf{y}+b_o\right) & \text{网络输出} \\ \mathbf{y}=f\left(\mathbf{W}^T\mathbf{x}+\mathbf{b}\right) & \text{隐含层输出} \end{cases}$$



- 要学习的变量是哪些？

  输出层权值 $\boldsymbol{\omega}$，输出层偏移 b_o，

  隐含层权值 $\mathbf{W}$，隐含层偏移 $\mathbf{b}$

- 怎么学习？

给定一组标记样本，采用反向传播梯度下降算法最小化误差函数

↓

即求 J 最小(极小)值点

梯度下降

- 求函数 $y=f(x)$ 的最小（极小）值点

解析求解： $y'=\frac{df}{dx}=0 \Rightarrow x=x_0$。例如 $f(x)=2x^2+x+1$，则最小值位置为

$$\frac{df}{dx}=4x+1=0 \Rightarrow x=-\frac{1}{4}$$

梯度下降

- 求函数 $y=f(x)$ 的极小（最小）值位置

解析求解： $y'=\frac{df}{dx}=0 \Rightarrow x=x_0$。例如 $f(x)=2x^2+x+1$，则最小值位置为

$$\frac{df}{dx}=4x+1=0 \Rightarrow x=-\frac{1}{4}$$

数值求解：采用迭代逼近求解

再如 $f(x)=2x^2+e^x+x$，则

$$\frac{df}{dx}=4x+e^x+1=0 \not\Rightarrow x=x_0$$

梯度下降

- 求函数 $y=f(x)$ 的极小（最小）值位置

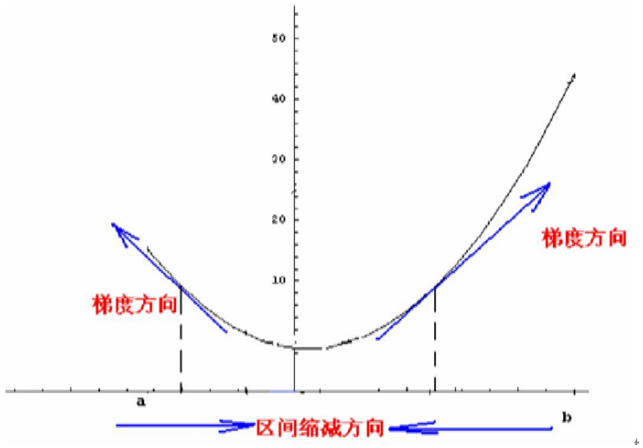
解析求解： $y'=\dfrac{df}{dx}=0 \quad \Rightarrow \quad x^*=x_0$。例如 $f(x)=2x^2+x+1$， 则最小值位置为

$$\frac{df}{dx}=4x+1=0 \quad \Rightarrow \quad x^*=-\frac{1}{4}$$

数值求解：采用迭代逼近求解

再如 $f(x)=2x^2+e^x+x$， 则

$$\frac{df}{dx}=4x+e^x+1=0 \quad \not\Rightarrow \quad x^*=x_0$$



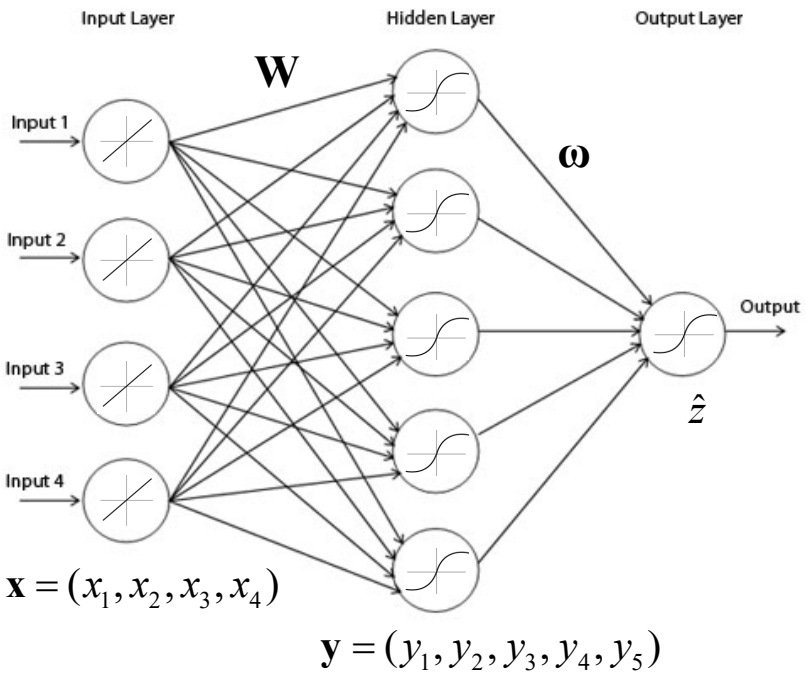
$$x^{(t+1)}=x^{(t)}-\lambda\Delta x=x^{(t)}-\lambda\left.\frac{df}{dx}\right|_{x=x^{(t+1)}}=x^{(t)}-\lambda\left(4x^{(t)}+e^{x^{(t)}}+1\right)$$

λ 称为学习步长或学习率。随着 $t\to\infty,\quad x^{(t)}\to x^*$

网络训练 – 梯度下降

- 对应我们的问题

$$\begin{cases} J(\mathbf{x};z)=\frac{1}{2}(\hat{z}-z)^2 & \text{目标函数} \\ \hat{z}=f\left(\boldsymbol{\omega}^T\mathbf{y}+b_o\right) & \text{网络输出} \\ \mathbf{y}=f\left(\mathbf{W}^T\mathbf{x}+\mathbf{b}\right) & \text{隐含层输出} \end{cases}$$



- 梯度下降算法

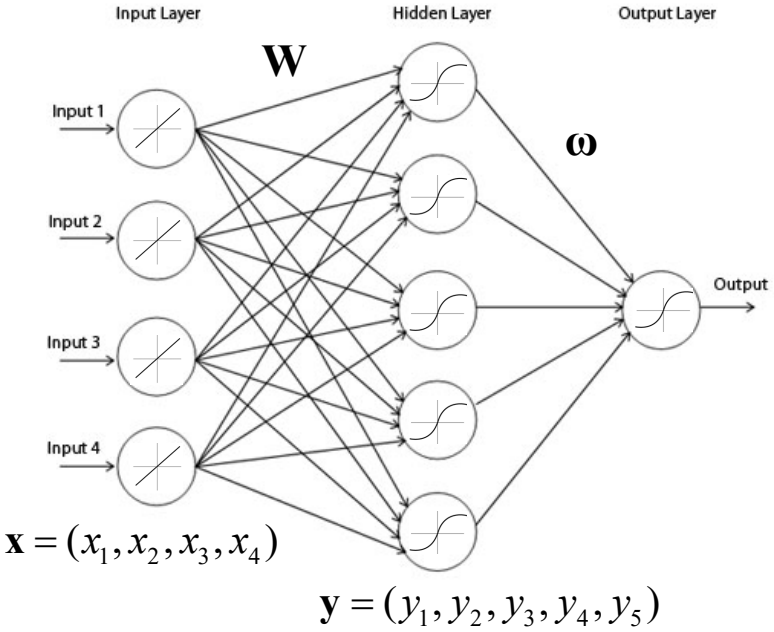
$$\begin{cases} \boldsymbol{\omega}^{(t+1)}=\boldsymbol{\omega}^{(t)}-\lambda\Delta\boldsymbol{\omega} \\ b_o^{(t+1)}=b_o-\lambda\Delta b_o \end{cases}$$

$$\begin{cases} \mathbf{W}^{(t+1)}=\mathbf{W}^{(t)}-\lambda\Delta\mathbf{W} \\ \mathbf{b}^{(t+1)}=\mathbf{b}-\lambda\Delta\mathbf{b} \end{cases}$$

所以关键就在于求梯度 $\Delta\boldsymbol{\omega},\Delta b_o,\Delta\mathbf{W},\Delta\mathbf{b}$

网络训练 – 输出层梯度计算

$$\begin{cases} J(\mathbf{x};z)=\frac{1}{2}\left(\hat{z}-z\right)^2 & \text{误差函数} \\ \hat{z}=f\left(\boldsymbol{\omega}^T\mathbf{y}+b_o\right) & \text{网络输出} \\ \mathbf{y}=f\left(\mathbf{W}^T\mathbf{x}+\mathbf{b}\right) & \text{隐含层输出} \end{cases}$$



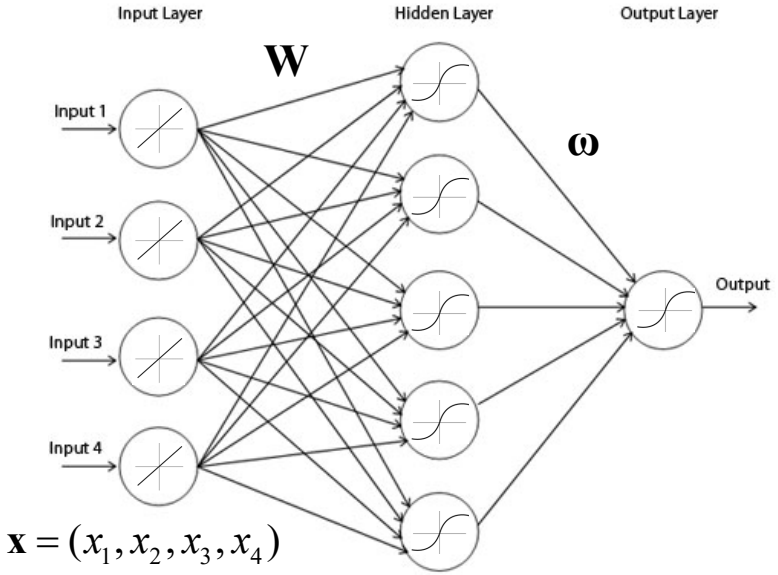
两层复合函数 $J(\hat{z}(\boldsymbol{\omega},b_o)$

根据复合函数链式求导法则

$$\begin{cases} \Delta\boldsymbol{\omega}=\frac{dJ}{d\hat{z}}\frac{\partial\hat{z}}{\partial\boldsymbol{\omega}}=\left(\hat{z}-z\right)f'\left(\boldsymbol{\omega}^T\mathbf{y}+b_o\right)\mathbf{y} \\ \Delta b_o=\frac{dJ}{d\hat{z}}\frac{\partial\hat{z}}{\partial b_o}=\left(\hat{z}-z\right)f'\left(\boldsymbol{\omega}^T\mathbf{y}+b_o\right) \end{cases}$$

网络训练 – 隐含层梯度计算

$$\begin{cases} J(\mathbf{x};z)=\frac{1}{2}\left(\hat{z}-z\right)^2 & \text{误差函数} \\ \hat{z}=f\left(\boldsymbol{\omega}^T\mathbf{y}+b_o\right) & \text{网络输出} \\ \mathbf{y}=f\left(\mathbf{W}^T\mathbf{x}+\mathbf{b}\right) & \text{隐含层输出} \end{cases}$$



三层复合函数 $J(\mathbf{W},\mathbf{b})=J(\hat{z}(\mathbf{y}(\mathbf{W},\mathbf{b})))$

根据复合函数链式求导法则

$$\begin{cases} \Delta\mathbf{W}=\frac{dJ}{d\hat{z}}\frac{\partial\hat{z}}{\partial\mathbf{W}}=\left(\hat{z}-z\right)\frac{\partial\hat{z}}{\partial\mathbf{W}} \\ \Delta\mathbf{b}=\frac{dJ}{d\hat{z}}\frac{\partial\hat{z}}{\partial\mathbf{b}}=\left(\hat{z}-z\right)\frac{\partial\hat{z}}{\partial\mathbf{b}} \end{cases}$$

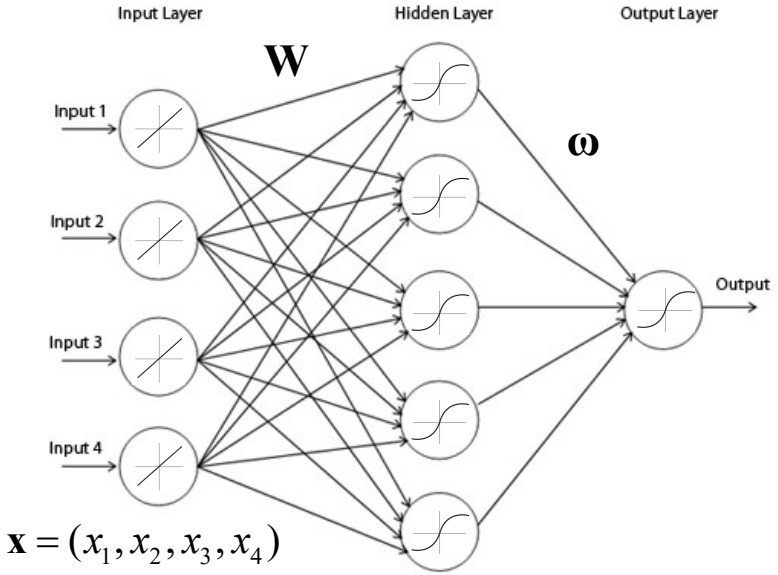
$$\begin{cases} \hat{z}=f(\mathbf{y}) \\ \mathbf{y}=f\left(\mathbf{w}\right) \end{cases}$$

元素运算函数 $y=f(w)$

$$\frac{\mathrm{d}\hat{z}}{\mathrm{d}\mathbf{w}}=\frac{\mathrm{d}\hat{z}}{\mathrm{d}\mathbf{y}}\odot\left.\frac{\mathrm{d}y}{\mathrm{d}w}\right|_{w\leftarrow\mathbf{w}}$$

网络训练 –隐含层梯度计算

$$\begin{cases} J(\mathbf{x};z)=\dfrac{1}{2}\left(\hat{z}-z\right)^2 & \text{误差函数} \\ \hat{z}=f\left(\boldsymbol{\omega}^T\mathbf{y}+b_o\right) & \text{网络输出} \\ \mathbf{y}=f\left(\mathbf{W}^T\mathbf{x}+\mathbf{b}\right) & \text{隐含层输出} \end{cases}$$



三层复合函数 $J(\mathbf{W},\mathbf{b})=J(\hat{z}(\mathbf{y}(\mathbf{W},\mathbf{b})))$

根据复合函数链式求导法则

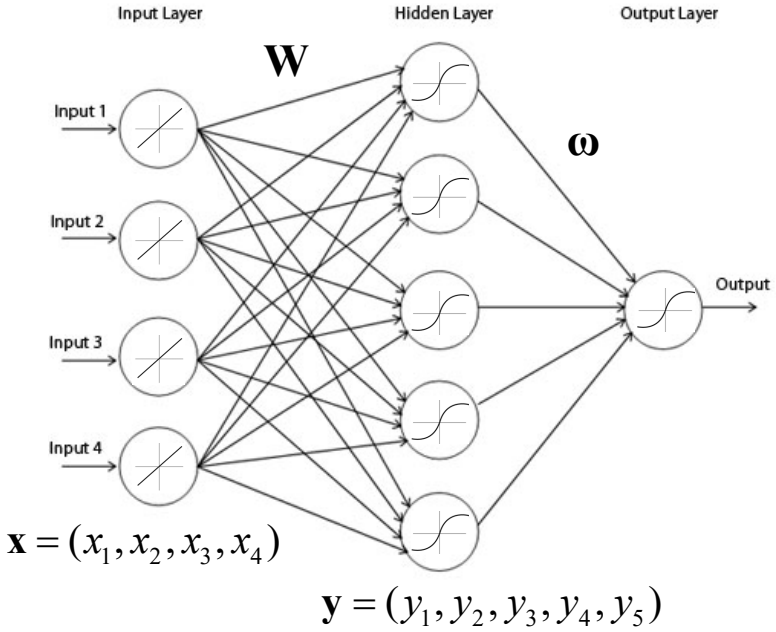
$$\begin{cases} \Delta\mathbf{W}=\dfrac{dJ}{d\hat{z}}\dfrac{\partial\hat{z}}{\partial\mathbf{W}}=\left(\hat{z}-z\right)\dfrac{\partial\hat{z}}{\partial\mathbf{W}} \\ \Delta\mathbf{b}=\dfrac{dJ}{d\hat{z}}\dfrac{\partial\hat{z}}{\partial\mathbf{b}}=\left(\hat{z}-z\right)\dfrac{\partial\hat{z}}{\partial\mathbf{b}} \end{cases}$$

$$\begin{cases} \hat{z}=f\left(\boldsymbol{\omega}^T\mathbf{y}+b_o\right) \\ \mathbf{y}=f\left(\mathbf{W}^T\mathbf{x}+\mathbf{b}\right) \end{cases}$$

$$\frac{\mathrm{d}\hat{z}}{\mathrm{d}\mathbf{W}}=\left[f'\left(\boldsymbol{\omega}^T\mathbf{y}+b_o\right)\boldsymbol{\omega}\odot f'\left(\mathbf{W}^T\mathbf{x}+\mathbf{b}\right)\right]\mathbf{x}^T$$

网络训练 –隐含层梯度计算

$$\begin{cases} J(\mathbf{x};z)=\frac{1}{2}\left(\hat{z}-z\right)^2 & \text{误差函数} \\ \hat{z}=f\left(\boldsymbol{\omega}^T\mathbf{y}+b_o\right) & \text{网络输出} \\ \mathbf{y}=f\left(\mathbf{W}^T\mathbf{x}+\mathbf{b}\right) & \text{隐含层输出} \end{cases}$$

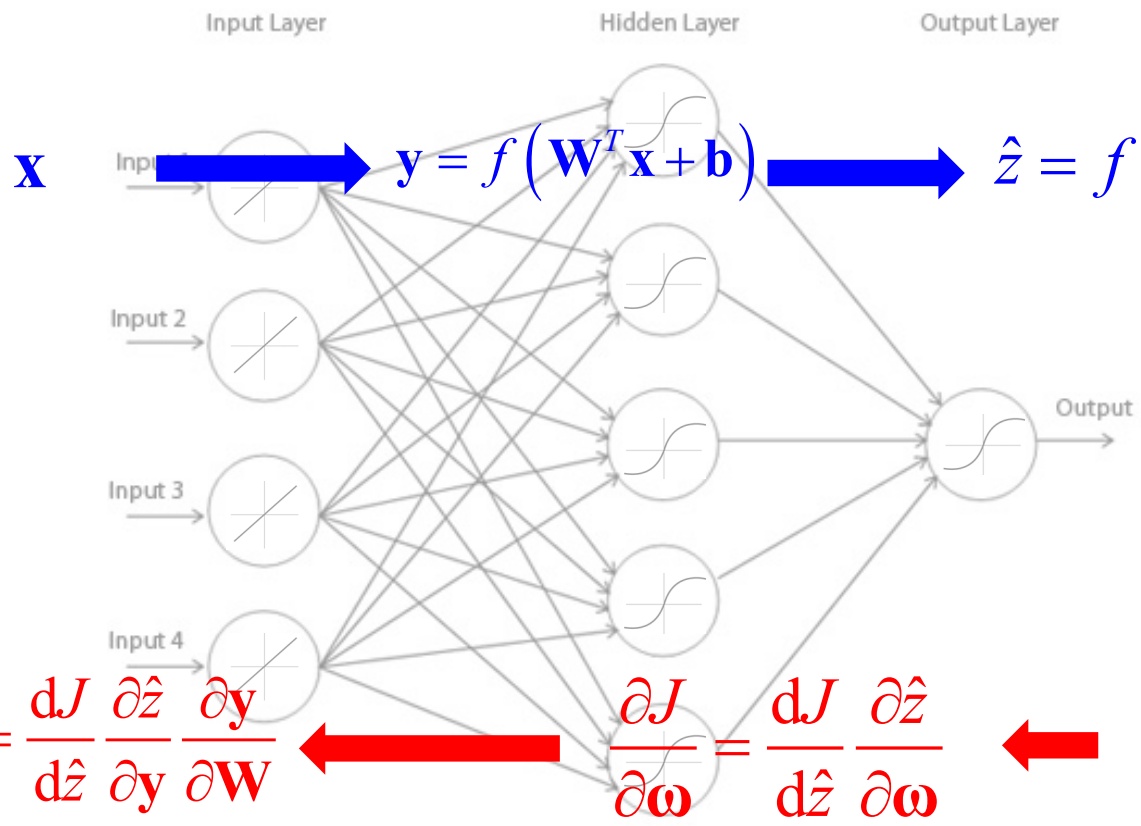


三层复合函数 $J(\mathbf{W},\mathbf{b})=J(\hat{z}(\mathbf{y}(\mathbf{W},\mathbf{b}))$

根据复合函数链式求导法则

$$\begin{cases} \Delta\mathbf{W}=\frac{dJ}{d\hat{z}}\frac{\partial\hat{z}}{\partial\mathbf{W}}=\left(\hat{z}-z\right)f'\left(\boldsymbol{\omega}^T\mathbf{y}+b_o\right)\boldsymbol{\omega}\odot f'\left(\mathbf{W}^T\mathbf{x}+\mathbf{b}\right)\mathbf{x}^T \\ \Delta\mathbf{b}=\frac{dJ}{d\hat{z}}\frac{\partial\hat{z}}{\partial\mathbf{b}}=\left(\hat{z}-z\right)f'\left(\boldsymbol{\omega}^T\mathbf{y}+b_o\right)\boldsymbol{\omega}\odot f'\left(\mathbf{W}^T\mathbf{x}+\mathbf{b}\right) \end{cases}$$

网络数据流向



$$\mathbf{x} \rightarrow \mathbf{y} = f\left(\mathbf{W}^T\mathbf{x} + \mathbf{b}\right) \rightarrow \hat{z} = f\left(\boldsymbol{\omega}^T\mathbf{y} + b_o\right)$$

$$J = \left(\hat{z} - z\right)^2$$

$$\frac{\mathrm{d}J}{\mathrm{d}\hat{z}} = (\hat{z} - z)$$

$$\frac{\partial J}{\partial \boldsymbol{\omega}} = \frac{\mathrm{d}J}{\mathrm{d}\hat{z}}\frac{\partial \hat{z}}{\partial \boldsymbol{\omega}}$$

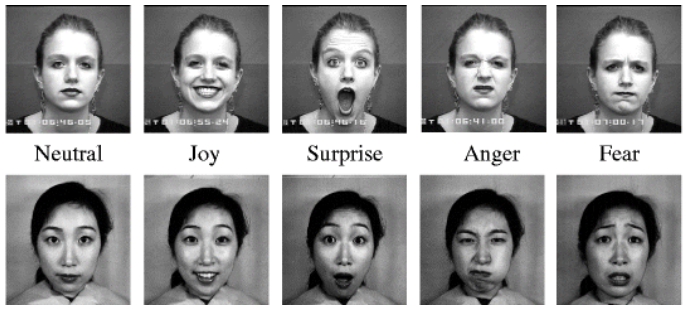
$$\frac{\partial J}{\partial \mathbf{W}} = \frac{\mathrm{d}J}{\mathrm{d}\hat{z}}\frac{\partial \hat{z}}{\partial \mathbf{y}}\frac{\partial \mathbf{y}}{\partial \mathbf{W}}$$

误差反向传播算法(Error back-propagation algorithm)

网络训练 – 标记数据

- 假设训练数据：$(\mathbf{x}_1, z_1),(\mathbf{x}_2, z_2),...,(\mathbf{x}_n, z_n)$，　$\mathbf{x}_i$ 称为样本，z_i 称为其对应的类标签

$(\mathbf{x}_1, z_1=\text{Anna}),(\mathbf{x}_2, z_2=\text{Anna}),...,(\mathbf{x}_5, z_5=\text{Anna})$



$(\mathbf{x}_6, z_6=\text{Lina}),(\mathbf{x}_7, z_7=\text{Lina}),...,(\mathbf{x}_{10}, z_{10}=\text{Lina})$

人脸识别

$(\mathbf{x}_1, z_1=0),(\mathbf{x}_2, z_2=0),...,(\mathbf{x}_{10}, z_{10}=0)$

$(\mathbf{x}_{91}, z_{91}=9),(\mathbf{x}_{92}, z_{92}=9),...,(\mathbf{x}_{100}, z_{100}=9)$

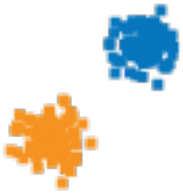
手写数字识别

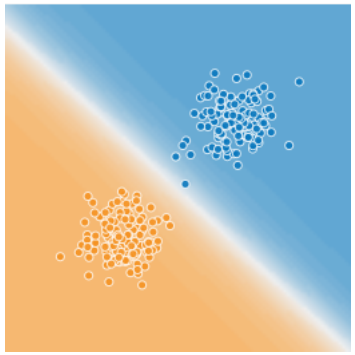
网络训练 – BP学习算法流程

	输入：训练数据集 $(\mathbf{x}_1,z_1),(\mathbf{x}_2,z_2),\ldots,(\mathbf{x}_n,z_n)$，学习步长 λ
1	随机初始化 $\boldsymbol{\omega},b_o,\mathbf{W},\mathbf{b}$，设置计数器 $t=0$
2	随机选择训练数据 $(\mathbf{x}_i,z_i)$
3	前向计算 $\mathbf{y}=f\left(\mathbf{W}^T\mathbf{x}_i+\mathbf{b}\right)$，$\hat{z}_i=f\left(\boldsymbol{\omega}^T\mathbf{y}+b_o\right)$，$J(\mathbf{x}_i;z_i)=\frac{1}{2}\left(\hat{z}_i-z_i\right)^2$
4	反向计算 $\Delta\boldsymbol{\omega},\Delta b_o,\Delta\mathbf{W},\Delta\mathbf{b}$
5	变量更新 $\begin{cases}\boldsymbol{\omega}^{(t+1)}=\boldsymbol{\omega}^{(t)}-\lambda\Delta\boldsymbol{\omega}\\ b_o^{(t+1)}=b_o-\lambda\Delta b_o\end{cases}$，$\begin{cases}\mathbf{W}^{(t+1)}=\mathbf{W}^{(t)}-\lambda\Delta\mathbf{W}\\ \mathbf{b}^{(t+1)}=\mathbf{b}-\lambda\Delta\mathbf{b}\end{cases}$
6	如未满足终止条件，$t=t+1$, 重复第2步

常用终止条件：最大迭代次数，目标函数值减少量，验证数据错误率等

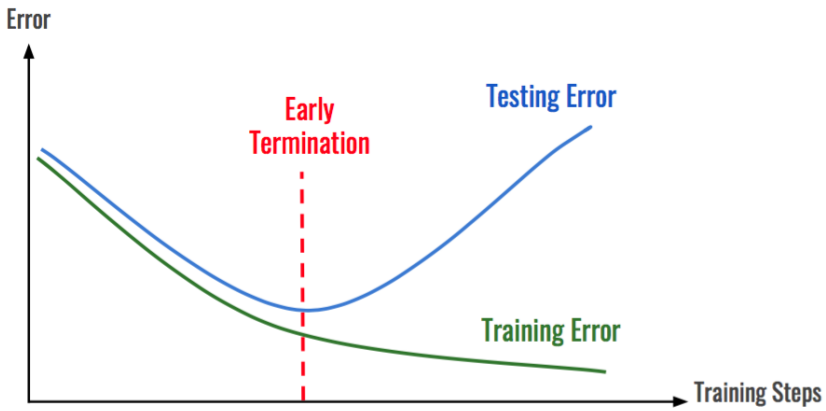
实验演示





多层神经网络

- 多层(三层)神经网络优势：
  - 可以证明：只需要一个包含足够多神经元的隐层，多层前馈神经网络就能以任意精度逼近**任意复杂度**的连续函数。
- 多层前馈网络局限
  - 强大的拟合能力使网络经常遭遇过拟合：训练误差持续降低，但测试误差却可能上升

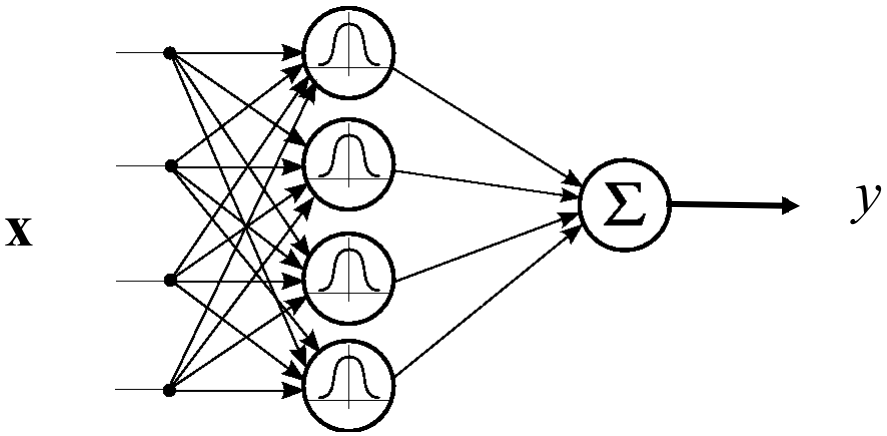


  - 如何设置隐层神经元的个数仍然是个未决问题，实际应用中通常使用“试错法”调整

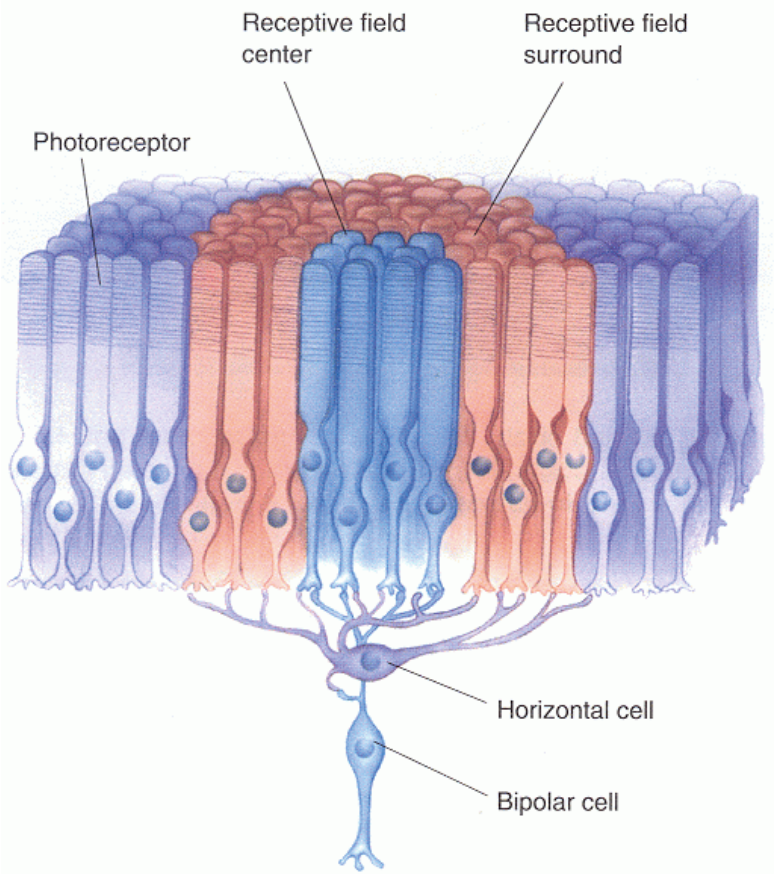
RBF网络

- RBF 网络是一种单隐层前馈神经网络，它使用径向基函数作为隐层神经元激活函数，而输出层则是隐层神经元输出的线性组合

$$\rho_i(\mathbf{x}\,|\,\mathbf{c}_i,\beta_i)$$



$$\begin{cases} \rho_i(\mathbf{x}\,|\,\mathbf{c}_i,\beta_i) = e^{-\beta_i\|\mathbf{x}-\mathbf{c}_i\|^2} \\ y = \sum_{i=1}^{q} w_i\rho_i(\mathbf{x}\,|\,\mathbf{c}_i,\beta_i) \end{cases}$$



RBF网络

- RBF网络性能

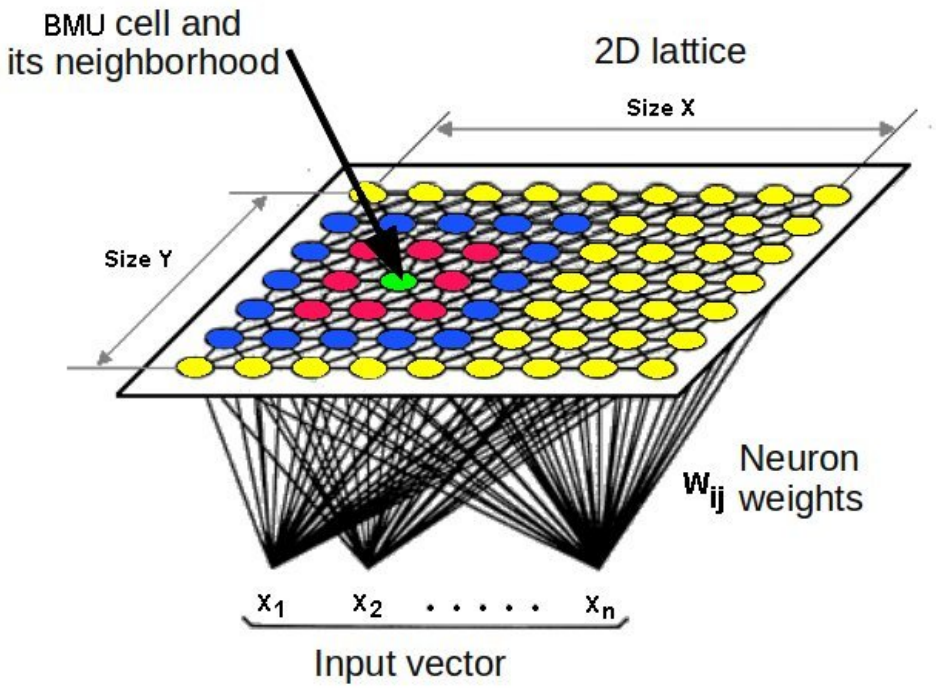
  具有足够多隐层神经元RBF 神经网络能以任意精度逼近任意连续函数.

  [Park and Sandberg，1991]

- RBF网络训练
  - Step1:确定神经元中心，常用的方式包括随机采样、聚类等
  - Step2:利用BP算法等确定连接权重 w 和激活函数参数 β_i

自组织映射网络

- 自组织网络映射(Self Organization Map, SOM) 网络是一种竞争型的无监督神经网络
- 能将高维数据映射到低维空间（通常为2维），同时保持输入数据在高维空间的拓扑结构。



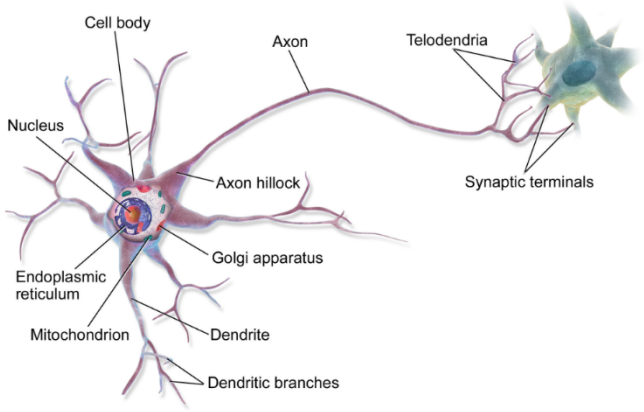
自组织映射网络

- SOM的优势
  - SOM 能将高维数据映射到低维空间（通常为2维），同时保持输入数据在高维空间的拓扑结构，即将高维空间中相似的样本点映射到网络输出层中邻近神经元。
  - 采用权值迭代更新，无需BP算法训练
- 训练过程
  - Step1:接受到一个训练样本后，每个输出层神经元计算该样本与自身权向量之间的距离，**距离最近**的神经元成为竞争获胜者
  - Step2:最佳匹配单元 u (Best Match Unit, BMU) 及其近邻神经元 v 的权值**w**将被调整，使得这些权向量与当前输入样本 **x** 的**距离缩小**

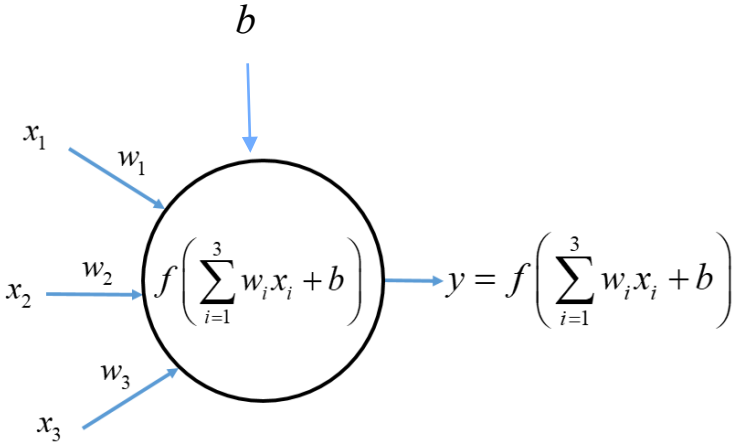
$$\mathbf{w}_v^{(t+1)} = \mathbf{w}_v^{(t)} + \lambda h(v,u)\left(\mathbf{x} - \mathbf{w}_v^{(t)}\right)$$

$\lambda, h(v,u)$ 为学习率和邻域函数

神经元模型 – 生物与人工



?

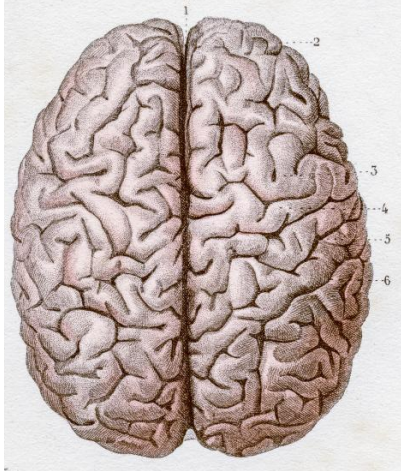


生物神经元结构

人工神经元模型

非常简单、非常粗糙的近似！

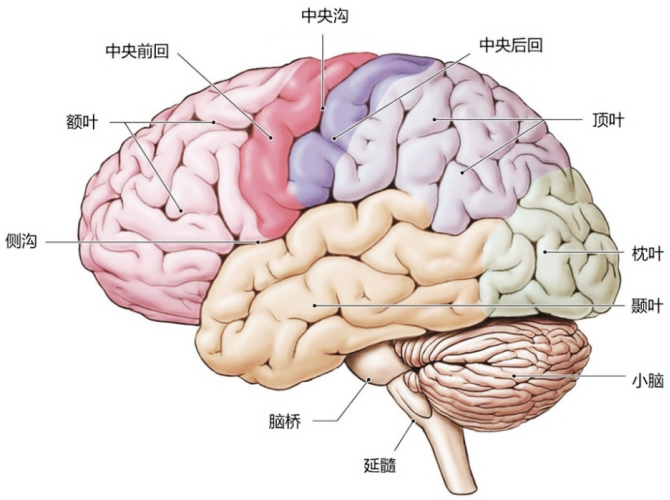
人脑结构

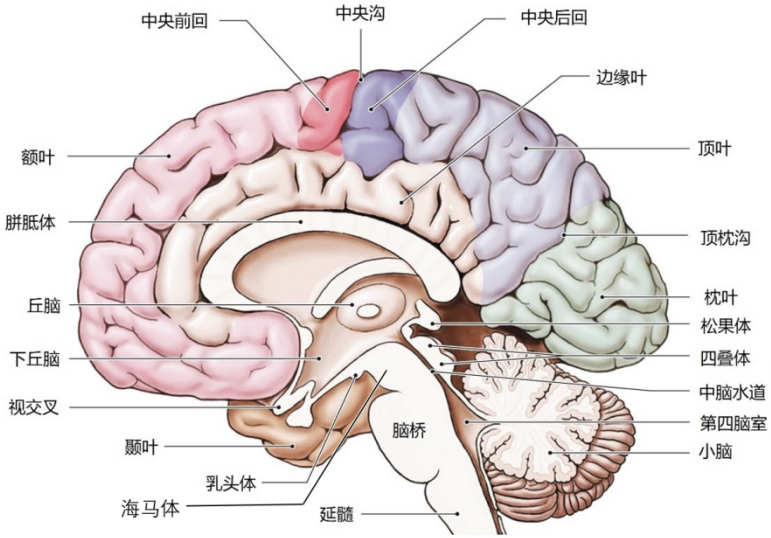


已知自然界最复杂、高度精致的系统

- 约160亿个神经元
- 至少125万亿个突触连接 （约银河系恒星数量的1500倍）
- 神经信号约480公里/小时的速度在神经元上传播

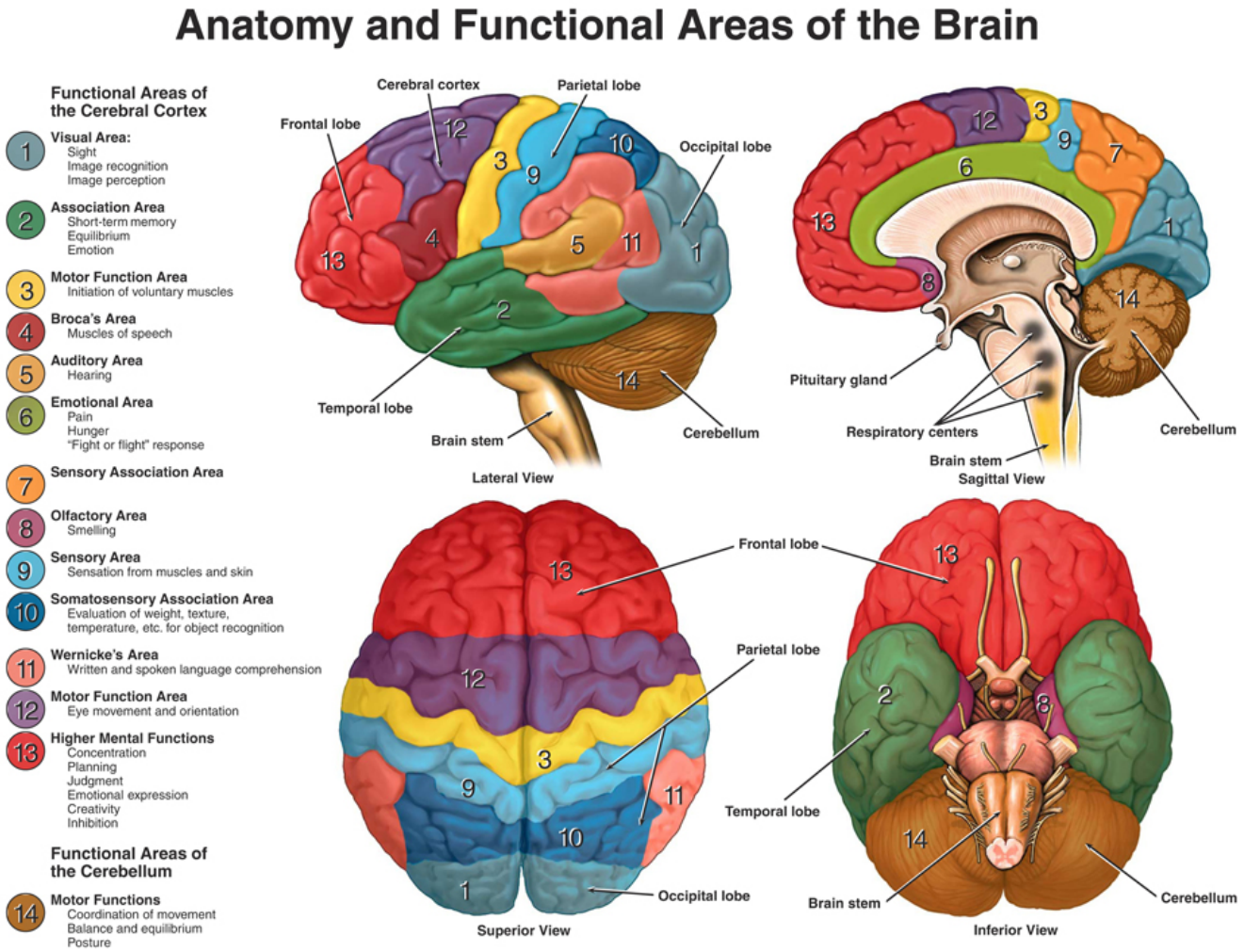
人脑结构



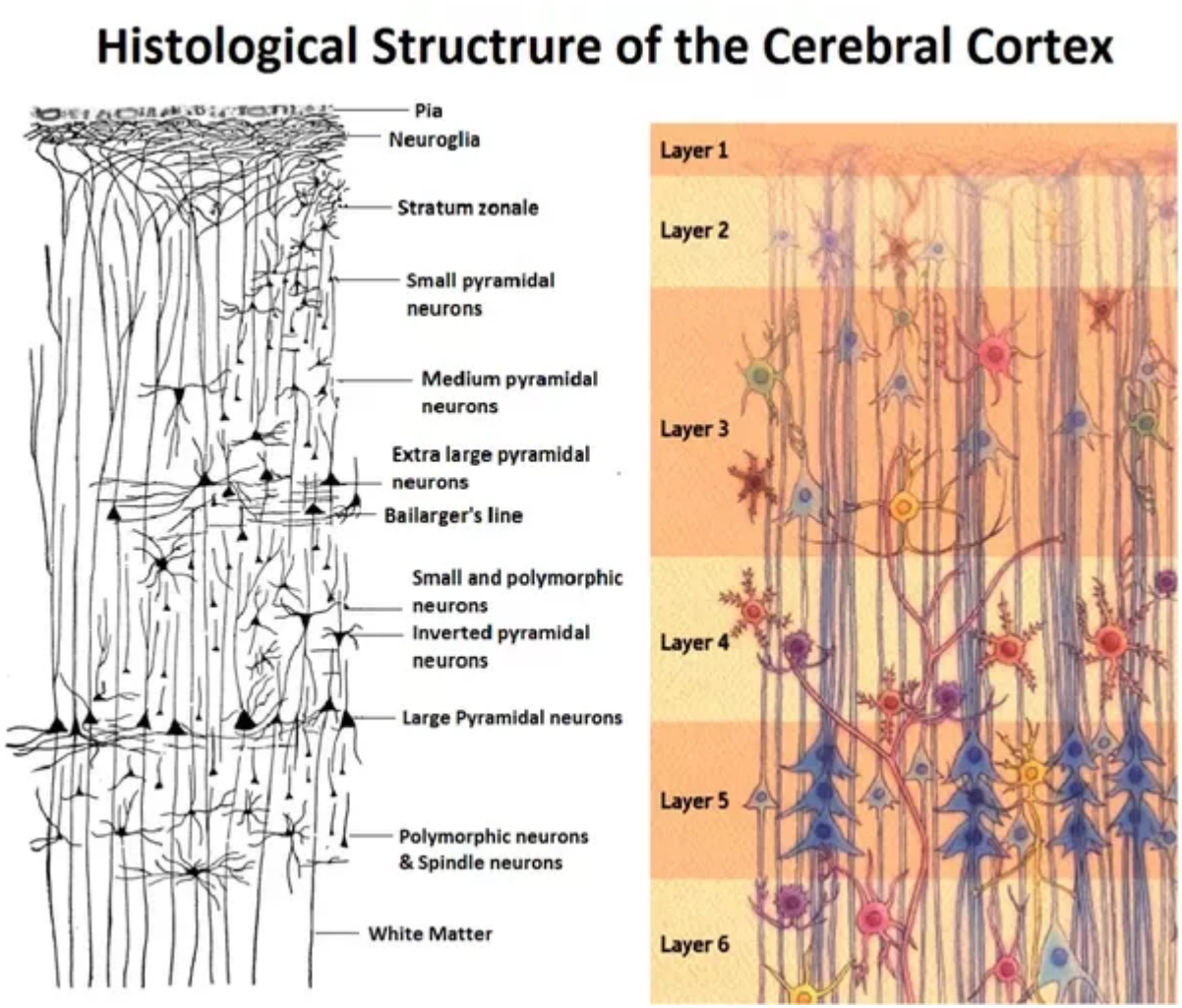


- **五个叶：**额叶，顶叶，枕叶，颞叶，岛叶
- **大脑皮质（也称灰质）：**是神经活动的最高中枢，厚度约为2-4毫米
- **白质（也称为髓质）：**是神经细胞的轴突所在，负责大脑皮质不同区域间的信息交流和共享。

功能分区

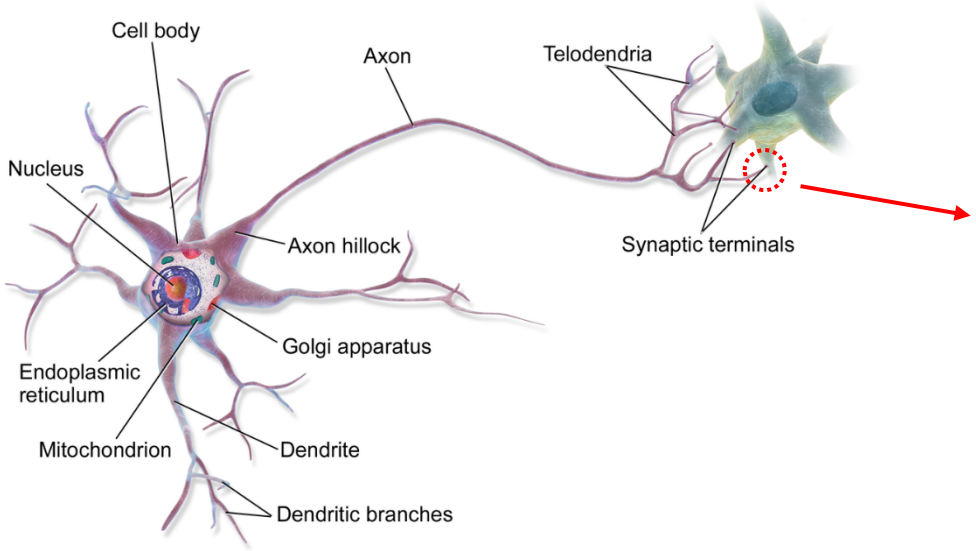

Anatomy and Functional Areas of the Brain
Functional Areas of the Cerebral Cortex
1 Visual Area: Sight, Image recognition, Image perception
2 Association Area: Short-term memory, Equilibrium, Emotion
3 Motor Function Area: Initiation of voluntary muscles
4 Broca's Area: Muscles of speech
5 Auditory Area: Hearing
6 Emotional Area: Pain, Hunger, "Fight or flight" response
7 Sensory Association Area
8 Olfactory Area: Smelling
9 Sensory Area: Sensation from muscles and skin
10 Somatosensory Association Area: Evaluation of weight, texture, temperature, etc. for object recognition
11 Wernicke's Area: Written and spoken language comprehension
12 Motor Function Area: Eye movement and orientation
13 Higher Mental Functions: Concentration, Planning, Judgment, Emotional expression, Creativity, Inhibition
Functional Areas of the Cerebellum
14 Motor Functions: Coordination of movement, Balance and equilibrium, Posture
Cerebral cortex
Parietal lobe
Frontal lobe
Occipital lobe
Temporal lobe
Brain stem
Cerebellum
Lateral View
Pituitary gland
Respiratory centers
Brain stem
Cerebellum
Sagittal View
Frontal lobe
Parietal lobe
Occipital lobe
Superior View
Temporal lobe
Brain stem
Cerebellum
Inferior View

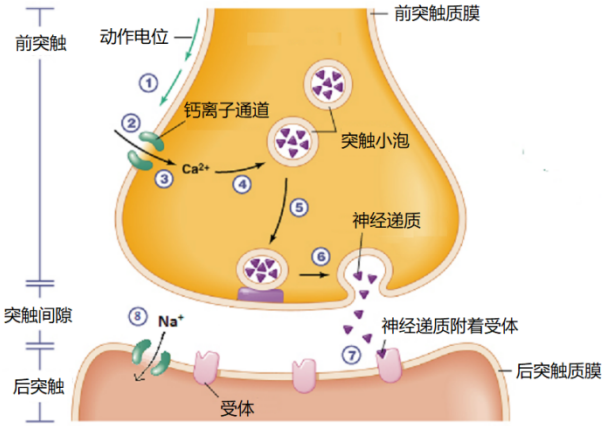
大脑脑皮层结构



从表面往深层依次是：
分子层、
外颗粒层、
外椎体细胞层、
内颗粒层、
内锥体细胞层
多形细胞层。

生物神经元

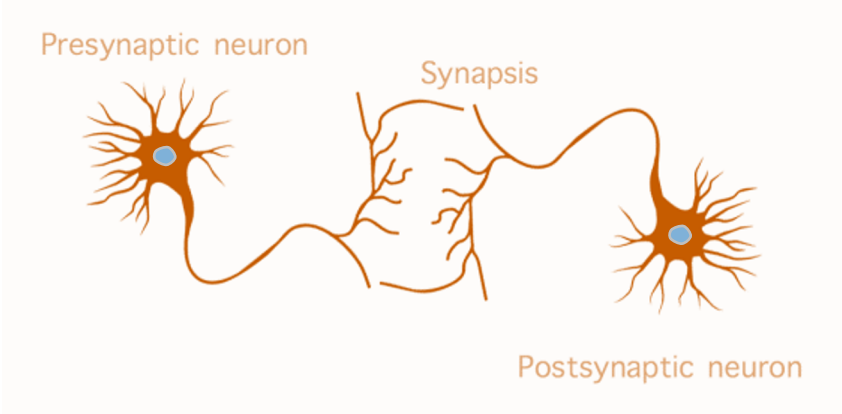


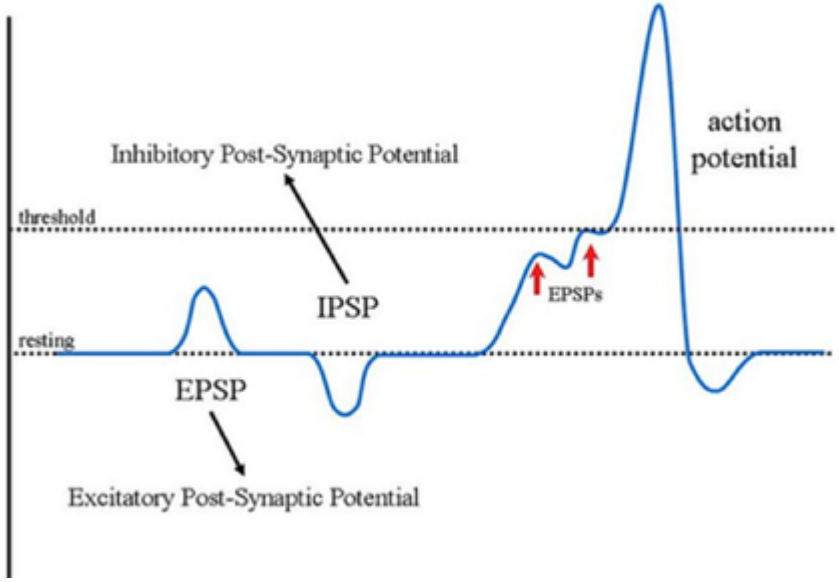


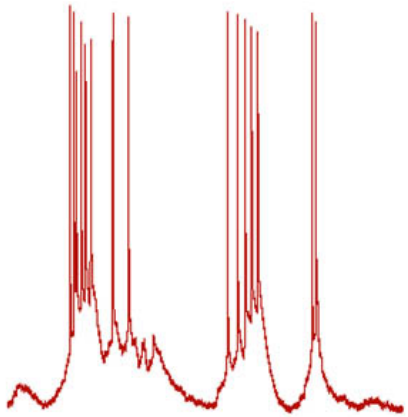
前突触动作电位→钙通道打开→钙离子进入→突触小泡前移→突触质膜融合→释放深递质→后突触递质受体接收→钠离子通道打开→钠离子进入→后突触极化反应→动作电位

动作电位传递过程

神经元动作电位

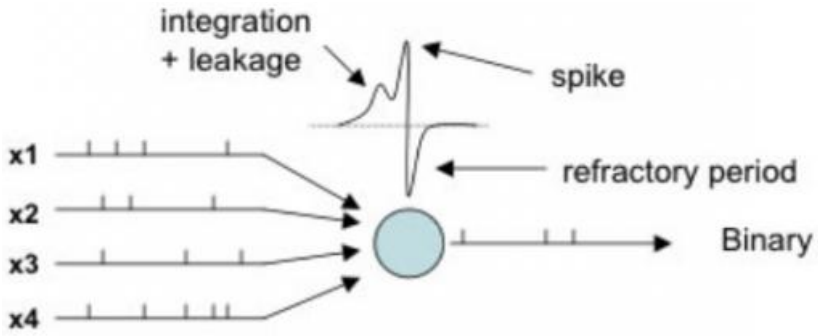

Presynaptic neuron
Synapsis
Postsynaptic neuron


Inhibitory Post-Synaptic Potential
action potential
threshold
IPSP
EPSPs
resting
EPSP
Excitatory Post-Synaptic Potential

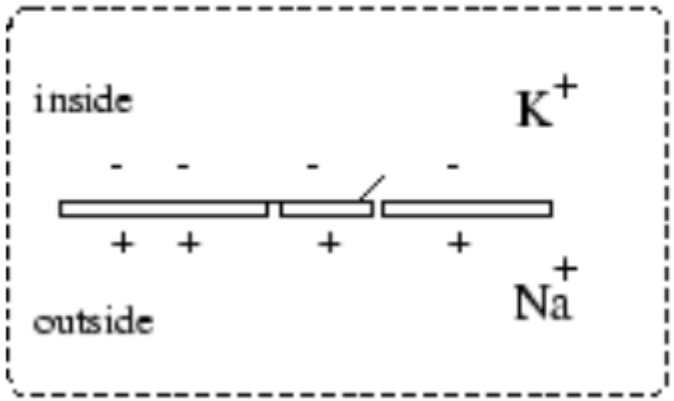


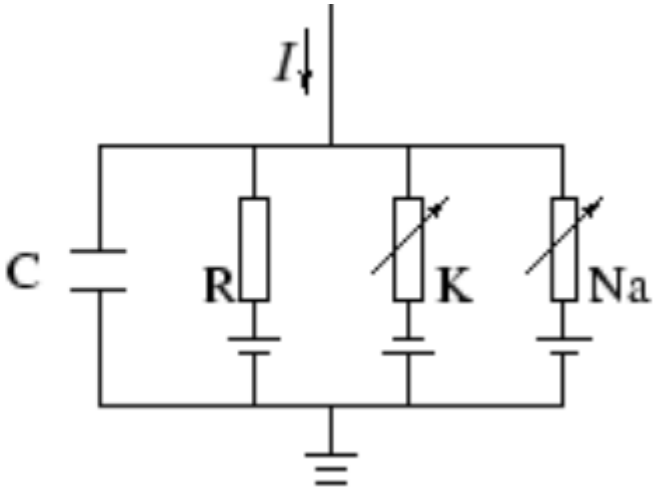
脉冲神经元

- 更接近于真实的生物神经模型
- 输入输出全是脉冲信号 (01001101011…)
- 天然地适合计算机硬件编程处理
- 生物神经信号不用转换可直接处理分析



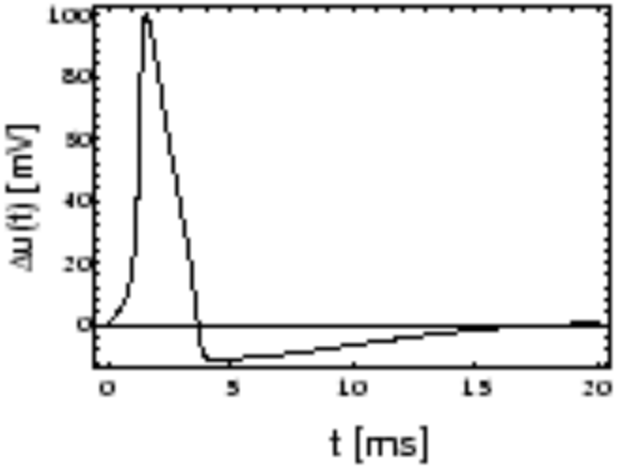
Hodgkin-Huxley Model



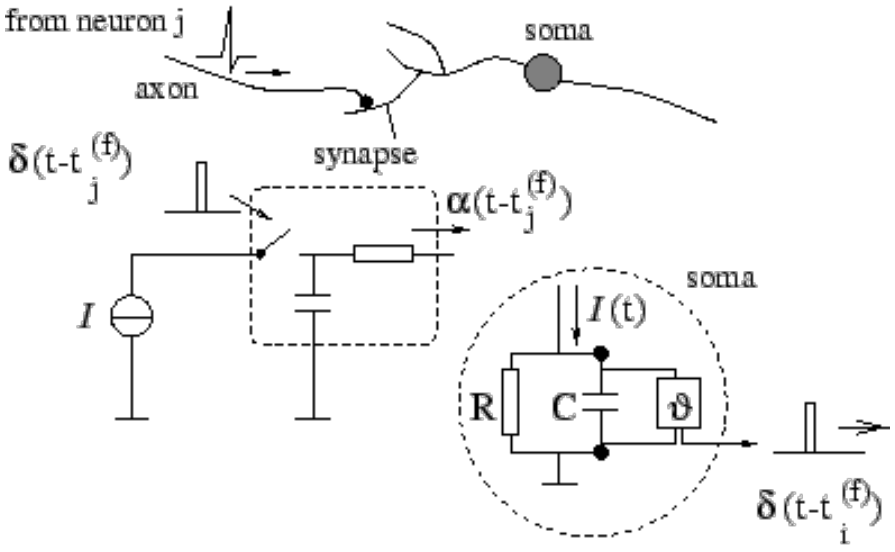


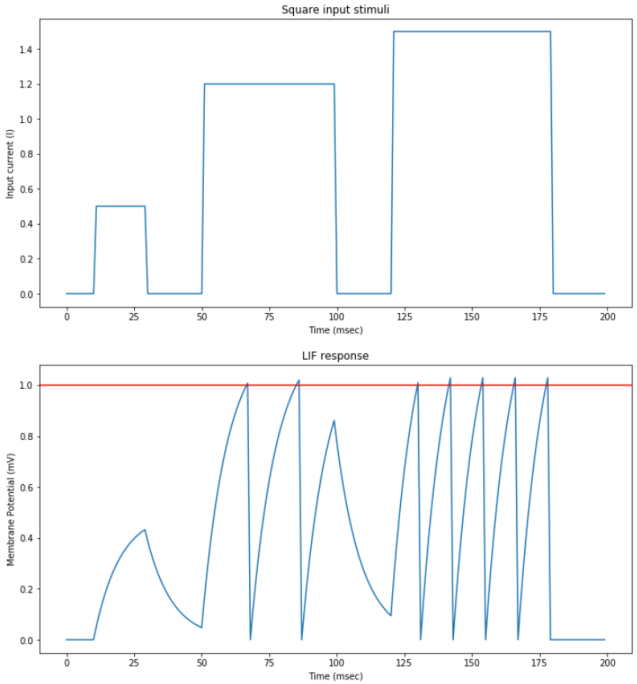
$$I(t) = I_C(t) + \sum_k I_k(t)$$

$$C\frac{du}{dt} = -\sum_k I_k(t) + I(t)$$



Leaky Integrate-and-Fire Model

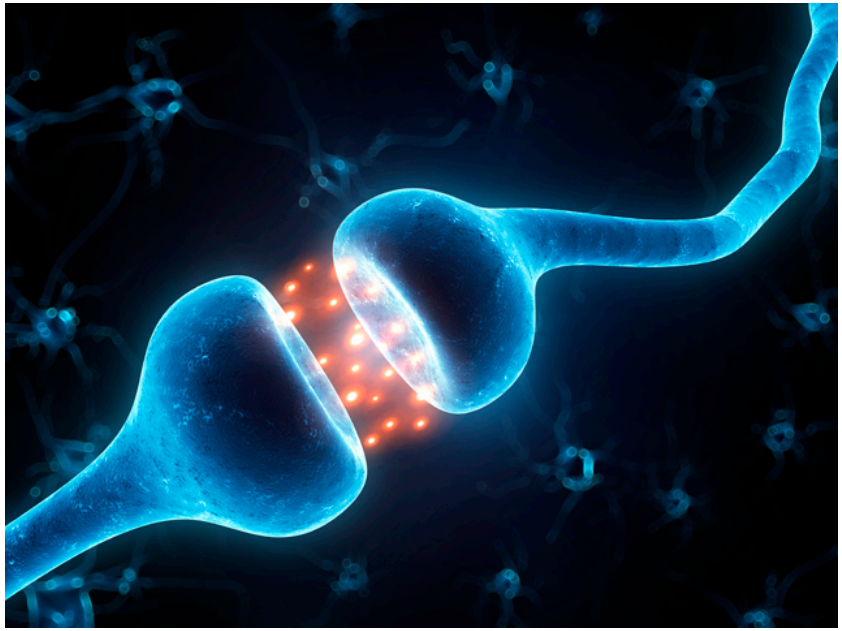


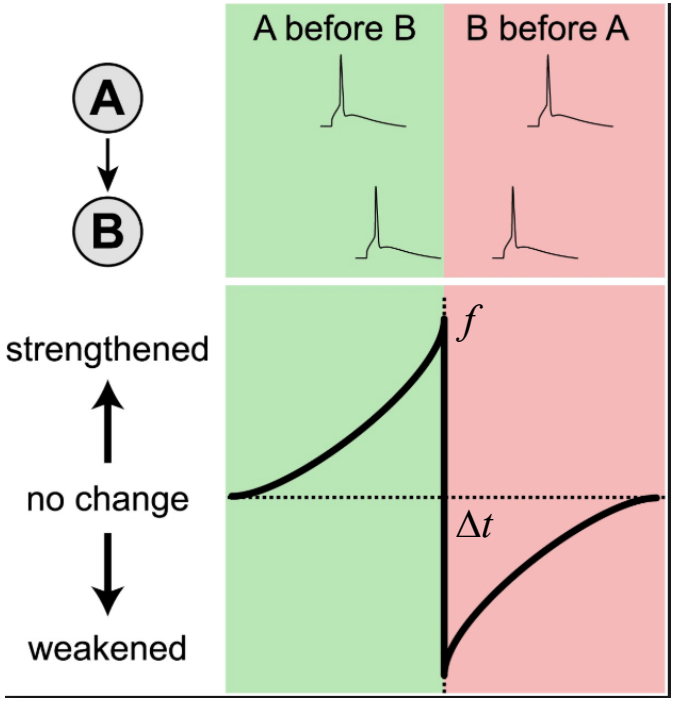


$$\tau_m \frac{du}{dt} = -u(t) + RI(t)$$

STDP 学习准则

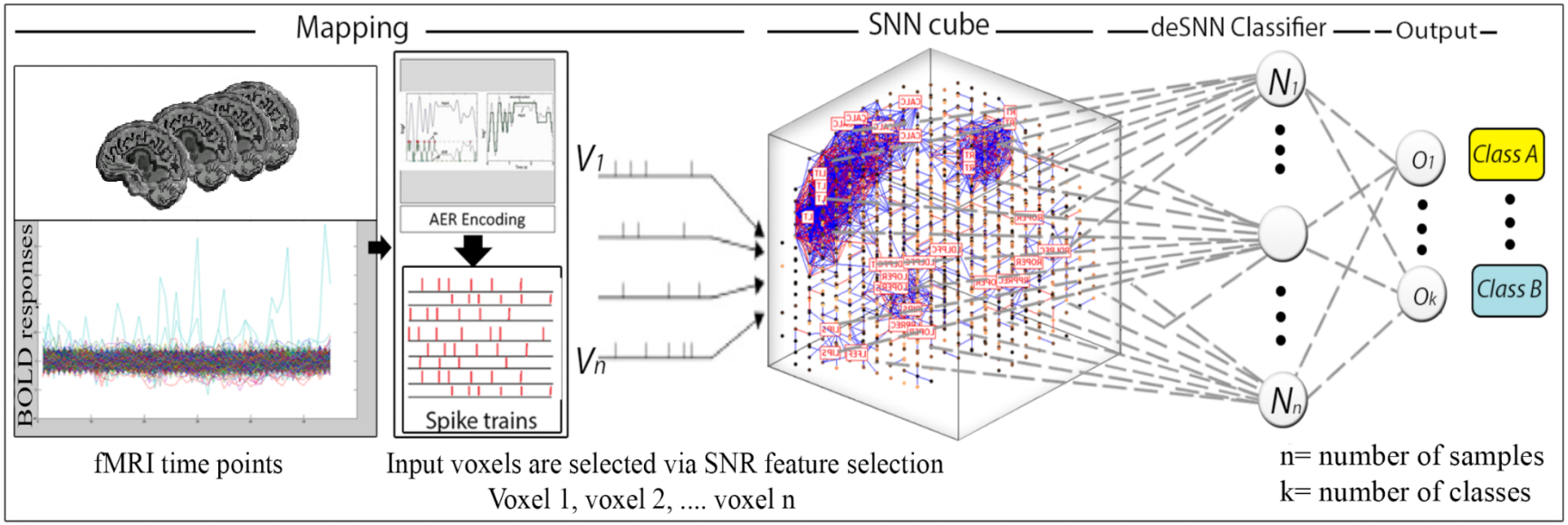
- Hebb理论：When one cell repeatedly assists in firing another, the axon of the first cell develops synaptic knobs (or enlarges them if they already exist) in contact with the soma of the second cell.





$$w^{(t+\Delta t)} = w^{(t)} + f(\Delta t)$$

脉冲神经网络


Mapping
SNN cube
deSNN Classifier
Output
BOLD responses
AER Encoding
Spike trains
V1
Vn
N1
Nn
O1
Ok
Class A
Class B
fMRI time points
Input voxels are selected via SNR feature selection
Voxel 1, voxel 2, ... voxel n
n= number of samples
k= number of classes