BP神经网络

广州泰迪智能科技有限公司

张敏

泰迪智能科技
TipDM Intelligent Technology

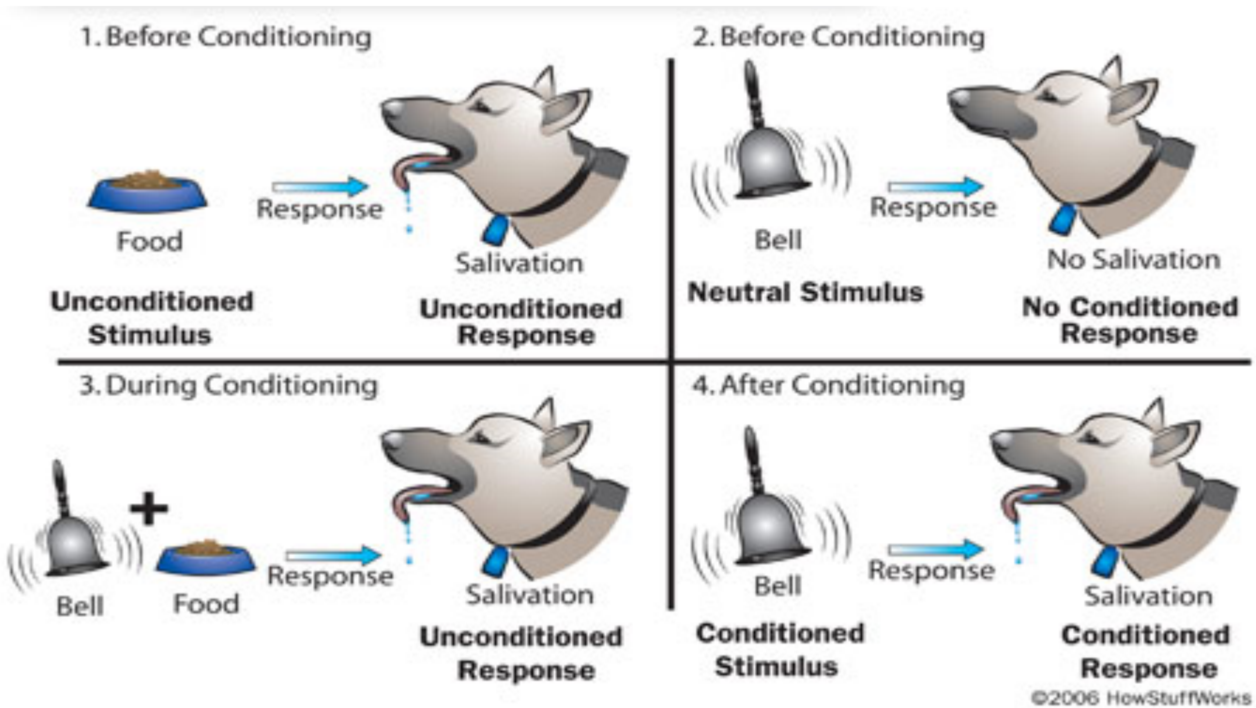
从鸢尾花数据集开始

类别列和前4列到底有何关系，能否通过已知样本构造出如下映射关系 f？

$$class = f(Sepal\_length, Sepal\_width, Petal\_length, Petal.width)$$

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	class
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
7	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.3	3.3	6	2.5	virginica
5.8	2.7	5.1	1.9	virginica
6.5	3	5.8	2.2	?
6.2	2.9	4.3	1.3	?

巴普洛夫关于神经反射的实验



$$Sepal\_length$$
$$Sepal\_width$$
$$Petal\_length$$
$$Petal.width$$

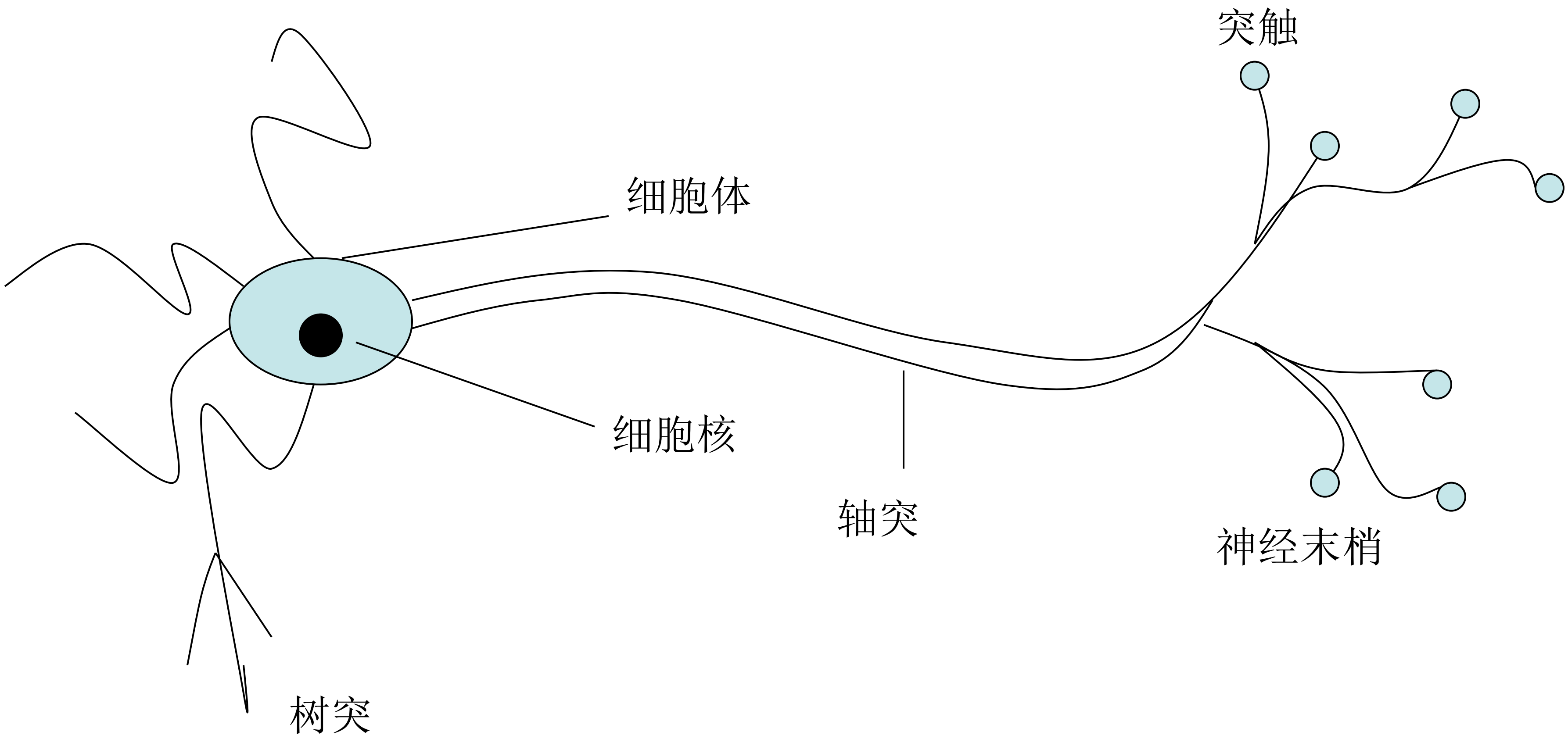
模型 / 系统

$$class$$

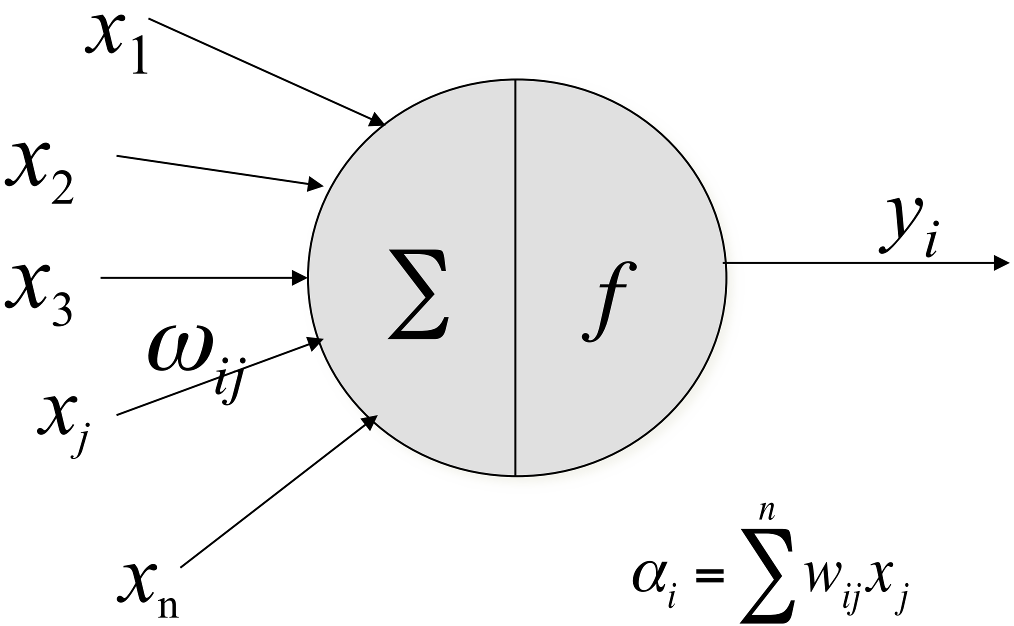
$$class = f(Sepal\_length, Sepal\_width, Petal\_length, Petal.width)$$

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	class
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
7	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.3	3.3	6	2.5	virginica
5.8	2.7	5.1	1.9	virginica
6.5	3	5.8	2.2	?
6.2	2.9	4.3	1.3	?

神经元结构



神经元结构

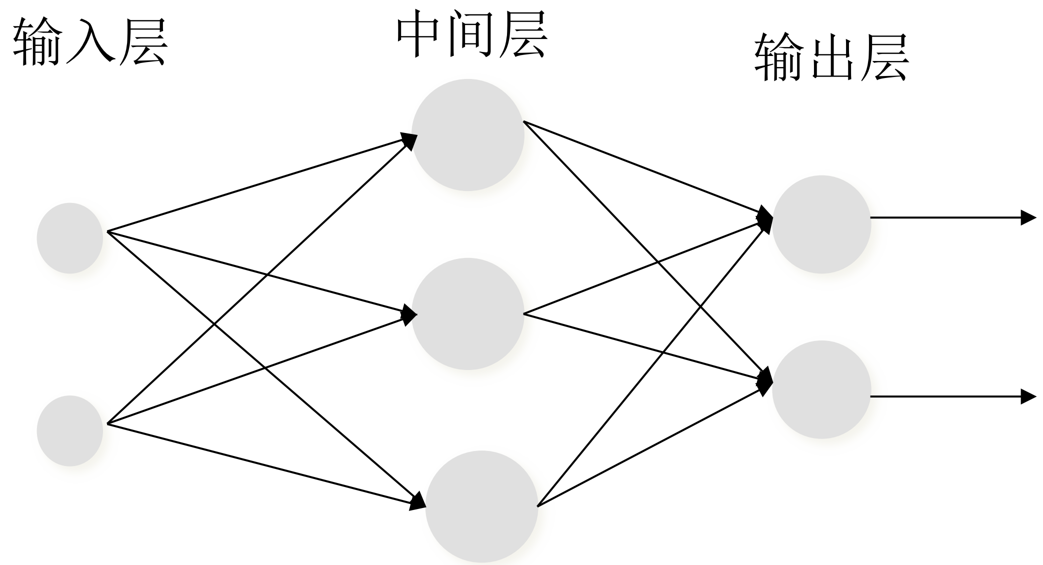


$$\alpha_i = \sum_{j=1}^{n} w_{ij} x_j \qquad y_i = f(\sum_{j=1}^{n} w_{ij} x_j - \theta)$$

x_j为输入信号， f为传递函数，$w_{i,j}$表示与神经元x_j 连接的权值，y_i表示输出值，θ表示阈值

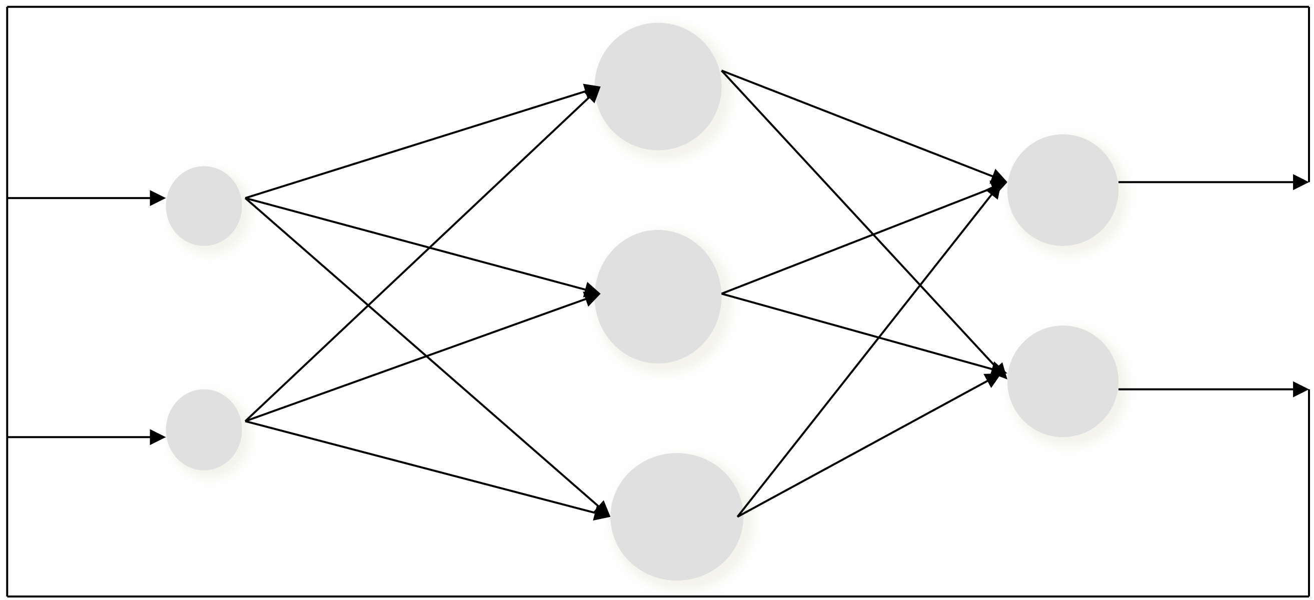
前向网络

特点： 每层只接受前一层的信息，没有反馈，如感知器网络。



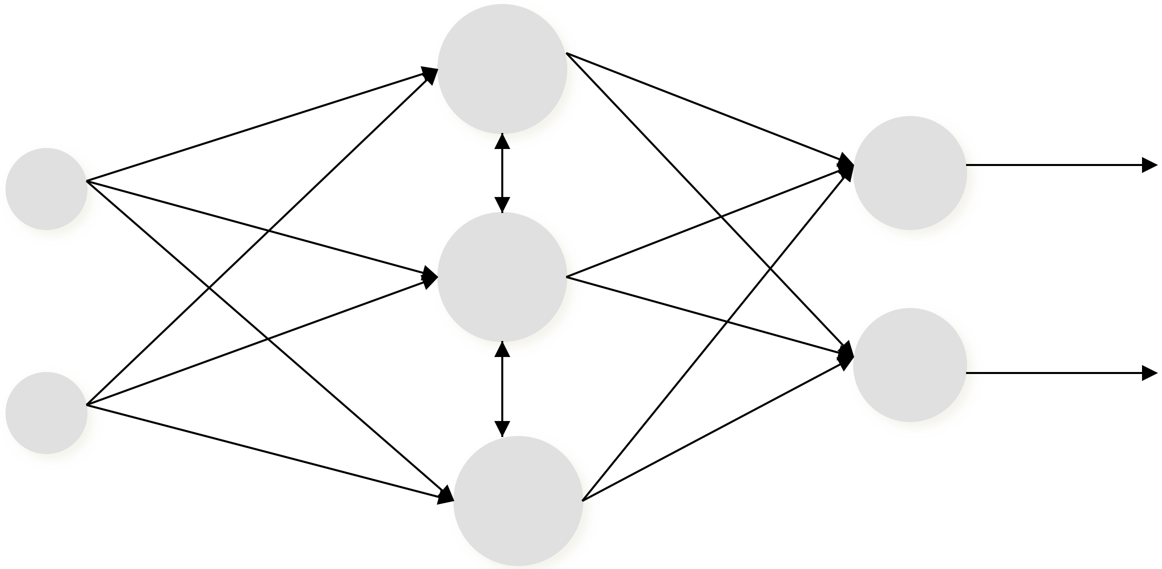
有反馈的前向神经网络

特点：输出层对输入层有反馈信息，如认知机和回归BP网络。



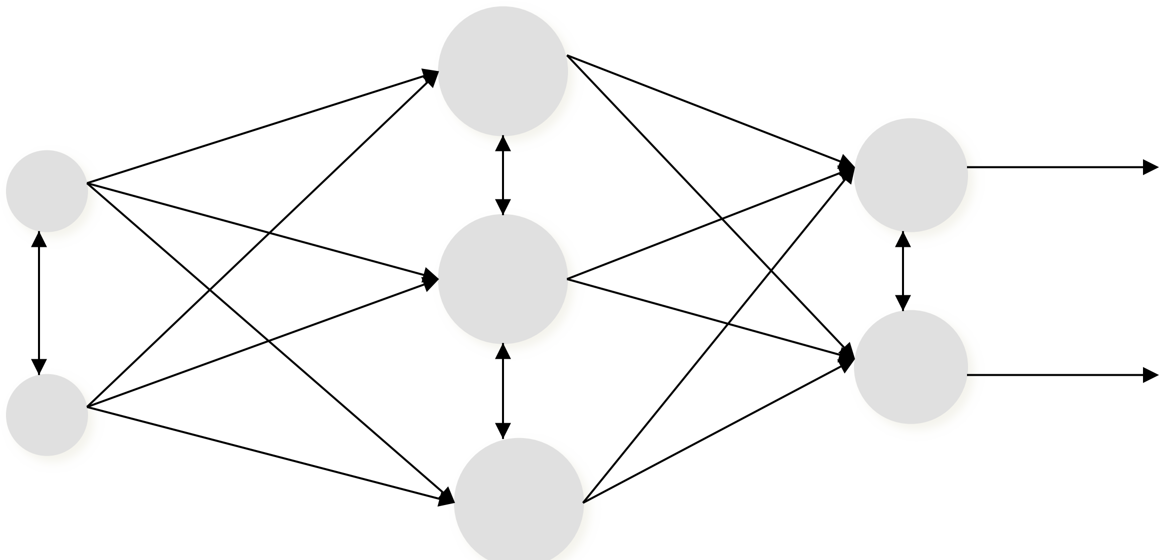
层内有互相结合的前向网络

特点：可以实现同一层内神经元之间的横向抑制或兴奋作用。

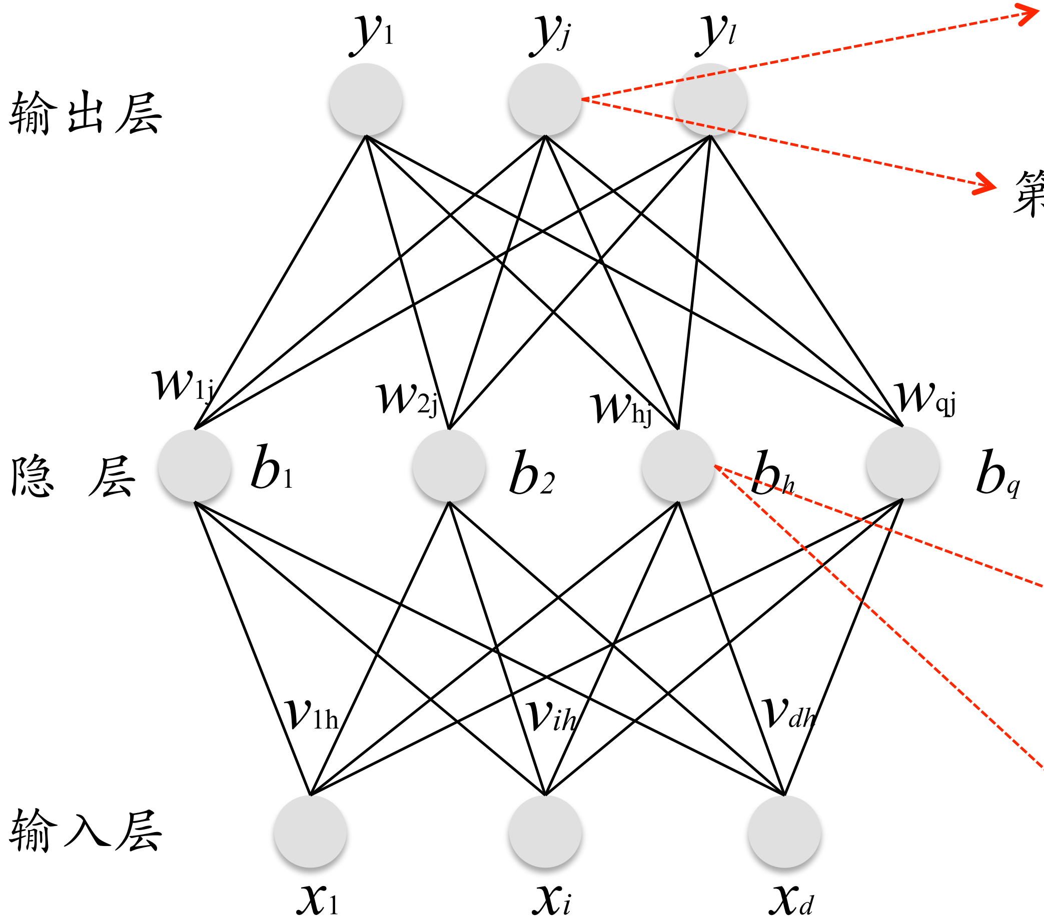


相互结合型网络

特点：任意两个神经元之间都可能有联系。



网络结构



第j个输出层神经元的输出：

$$\hat{y}_j = f(\beta_j - \theta_j)$$

第j个输出层神经元的输入：

$$\beta_j = \sum_{h=1}^{q} w_{hj} b_h$$

第h个隐层神经元的输出：

$$b_h = f(\alpha_h - \gamma_h)$$

第h个隐层神经元的输入：

$$\alpha_h = \sum_{i=1}^{d} v_{ih} x_i$$

传递函数的类型

阈值型

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

线性型

$$f(x_i)=\begin{cases}1 & x_i \geq x_2 \\ ax_i+b & x_1 \leq x_i < x_2 \\ 0 & x_i < x_1\end{cases}$$

S型

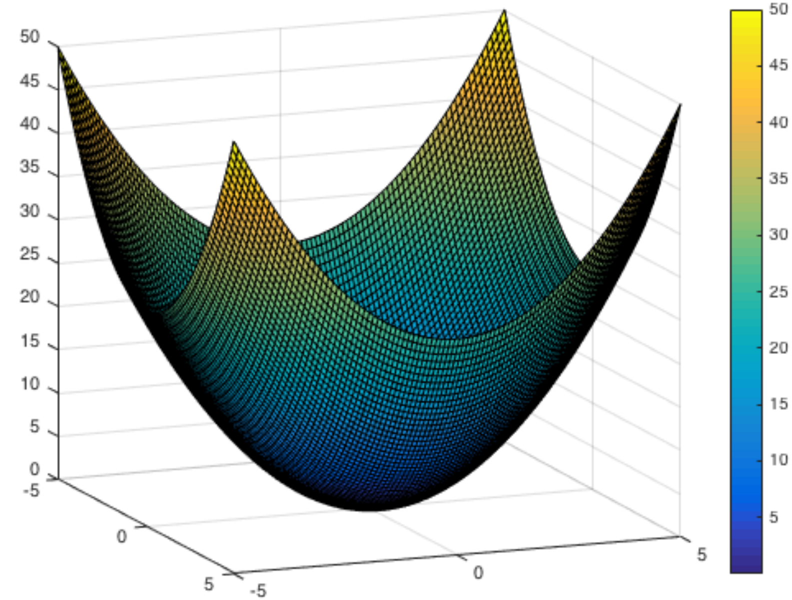
$$f(x)=sigmoid(x)=\frac{1}{1+e^{-x}}$$

泰迪智能科技 TipDM Intelligent Technology

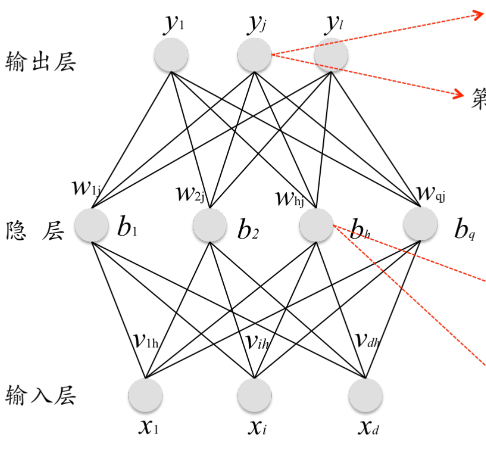
网络结构

	x_1	x_2	y
0	0.29	0.23	0.14
1	0.50	0.62	0.64
2	0.00	0.53	0.28
3	0.21	0.53	0.33
4	0.10	0.33	0.12
5	0.06	0.15	0.03
6	0.13	0.03	0.02
7	0.24	0.23	0.11
8	0.28	0.03	0.08
9	0.38	0.49	?
10	0.29	0.47	?

$$y = x_1^2 + x_2^2$$



网络结构



第j个输出层神经元的输出：

$$\hat{y}_j = f(\beta_j - \theta_j)$$

第j个输出层神经元的输入：

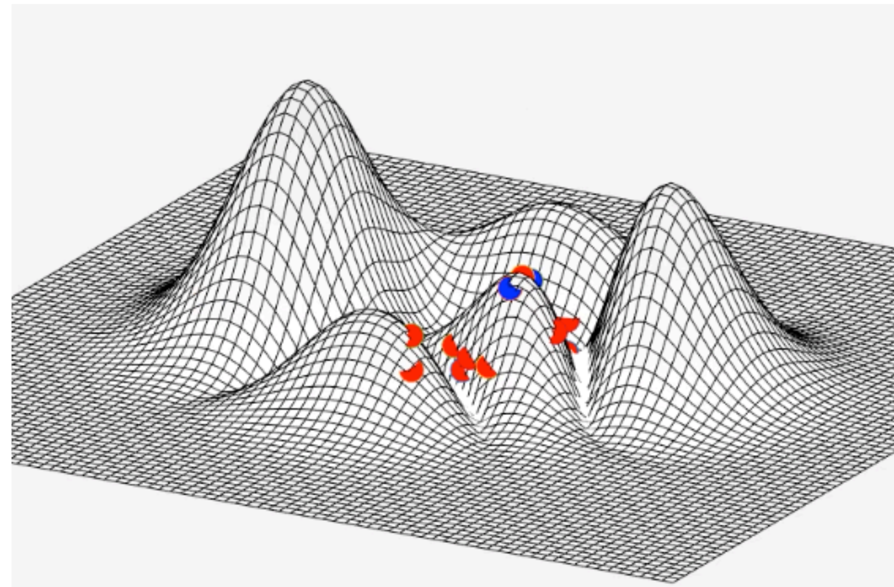
$$\beta_j = \sum_{h=1}^{q} w_{hj} b_h$$

第h个隐层神经元的输出：

$$b_h = f(\alpha_h - \gamma_h)$$

第h个隐层神经元的输入：

$$\alpha_h = \sum_{i=1}^{d} v_{ih} x_i$$



网络结构

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

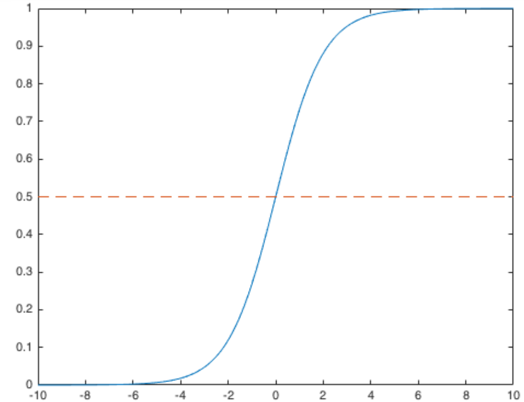
$$f^{'}(x) = f(x)(1-f(x))$$

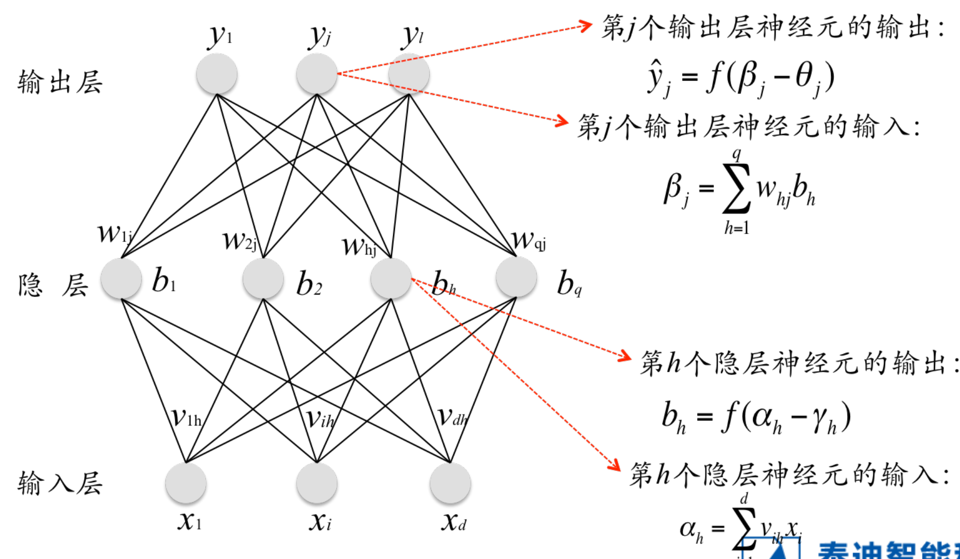
$$\hat{y}_j = f(\beta_j - \theta_j)$$

$$E = \frac{1}{2}\sum_{j=1}^{l}(\hat{y}_j - y_j)^2 \dashrightarrow \frac{\partial E}{\partial \hat{y}_j} = \hat{y}_j - y_j$$

$$\Delta w_{hj} = -\eta \frac{\partial E}{\partial w_{hj}}$$

$$\frac{\partial E}{\partial w_{hj}} = \frac{\partial E}{\partial \hat{y}_j} \cdot \frac{\partial \hat{y}_j}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}}$$





网络结构

$$\frac{\partial E}{\partial w_{hj}} = \frac{\partial E}{\partial \hat{y}_j} \cdot \frac{\partial \hat{y}_j}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}}$$

$$\frac{\partial \beta_j}{\partial w_{hj}} = b_h \qquad \frac{\partial E}{\partial \hat{y}_j} = \hat{y}_j - y_j$$

$$\frac{\partial \hat{y}_j}{\partial \beta_j} = f'(\beta_j - \theta_j)$$

$$f'(x) = f(x)(1 - f(x))$$

$$= f(\beta_j - \theta_j)(1 - f(\beta_j - \theta_j))$$

$$= \hat{y}_j(1 - \hat{y}_j)$$

$$g_j = -\frac{\partial E}{\partial \hat{y}_j} \cdot \frac{\partial \hat{y}_j}{\partial \beta_j}$$

$$= -(\hat{y}_j - y_j)\hat{y}_j(1 - \hat{y}_j)$$

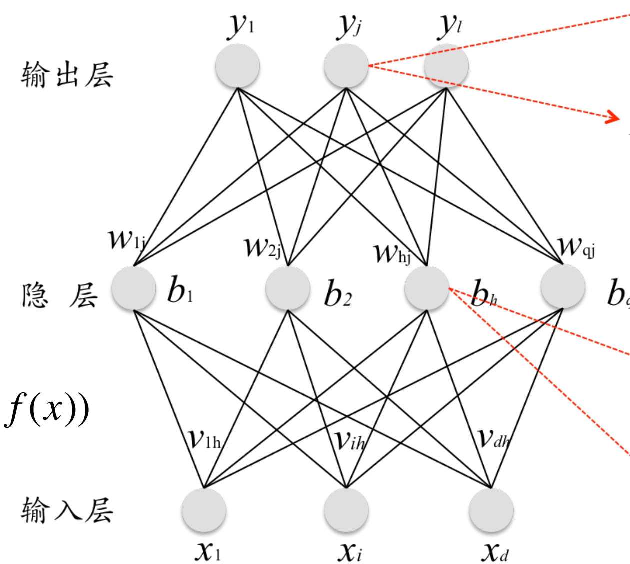
$$= \hat{y}_j(1 - \hat{y}_j)(y_j - \hat{y}_j)$$

$$\Delta w_{hj} = -\eta \frac{\partial E}{\partial w_{hj}}$$

$$= -\eta \frac{\partial E}{\partial \hat{y}_j} \cdot \frac{\partial \hat{y}_j}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}}$$

$$= \eta g_j b_h$$

$$\Delta w_{hj} = \eta \hat{y}_j(1 - \hat{y}_j)(y_j - \hat{y}_j)b_h$$



第j个输出层神经元的输出：

$$\hat{y}_j = f(\beta_j - \theta_j)$$

第j个输出层神经元的输入：

$$\beta_j = \sum_{h=1}^{q} w_{hj} b_h$$

第h个隐层神经元的输出：

$$b_h = f(\alpha_h - \gamma_h)$$

第h个隐层神经元的输入：

$$\alpha_h = \sum_{i=1}^{d} v_{ih} x_i$$

网络结构

$$\Delta w_{hj} = -\eta \frac{\partial E}{\partial w_{hj}}$$

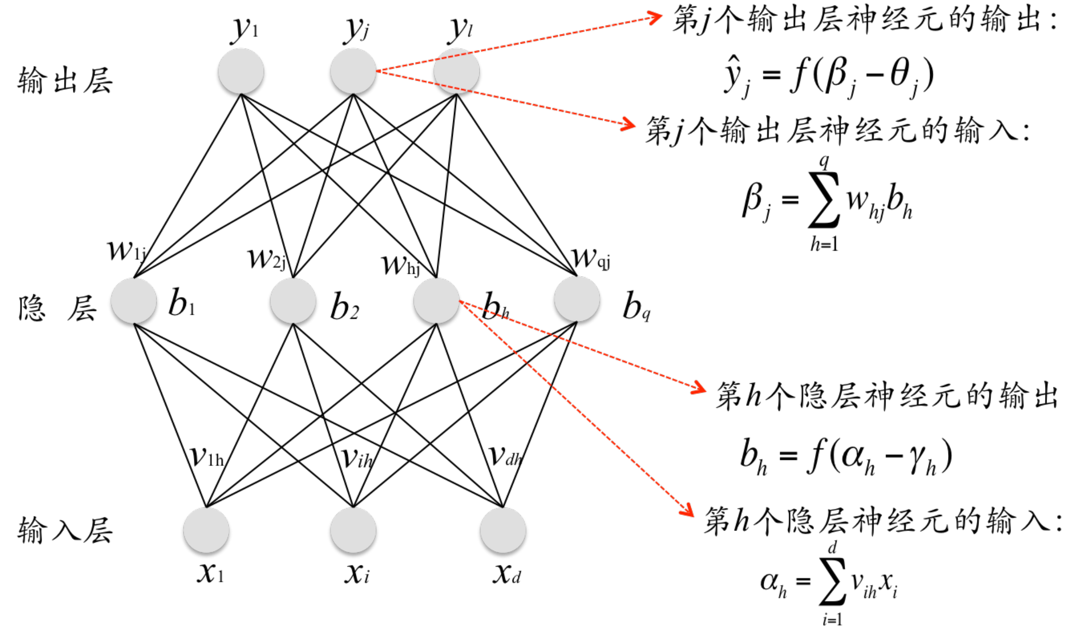
$$= -\eta \frac{\partial E}{\partial \hat{y}_j} \cdot \frac{\partial \hat{y}_j}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}}$$

$$= \eta g_j b_h$$

$$= \eta \hat{y}_j (1 - \hat{y}_j)(y_j - \hat{y}_j) b_h$$

$$\nabla \theta_j = -\eta g_j$$

$$= -\eta \hat{y}_j (1 - \hat{y}_j)(y_j - \hat{y}_j)$$



网络结构

$$\nabla v_{ih} = \eta e_h x_i$$

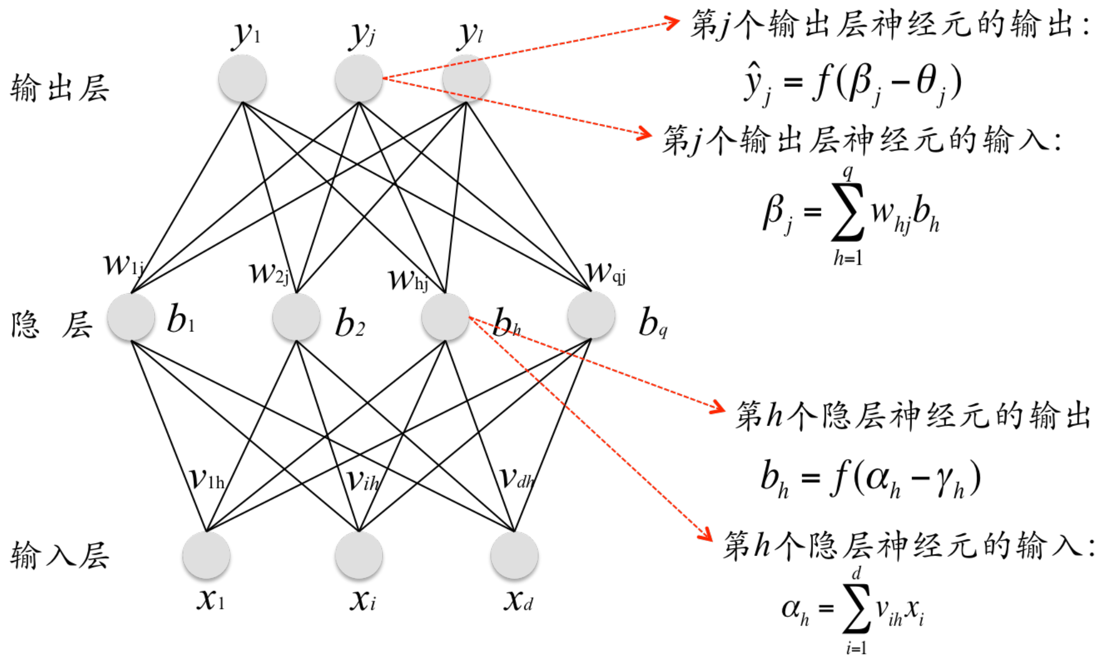
$$= -\eta \frac{\partial E}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h} x_i$$

$$= \eta b_h (1 - b_h) \sum_{j=1}^{l} w_{hj} g_j x_i$$

$$\nabla \gamma_h = -\eta e_h$$

$$= \eta \frac{\partial E}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h}$$

$$= -\eta b_h (1 - b_h) \sum_{j=1}^{l} w_{hj} g_j$$



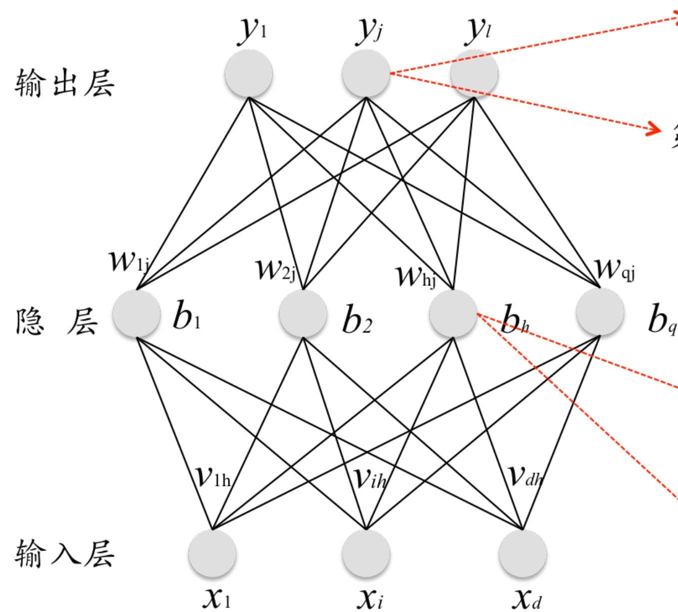
网络结构

$$\Delta w_{hj} = \eta \hat{y}_j (1-\hat{y}_j)(y_j-\hat{y}_j) b_h$$

$$\nabla \theta_j = -\eta \hat{y}_j (1-\hat{y}_j)(y_j-\hat{y}_j)$$

$$\nabla v_{ih} = \eta b_h (1-b_h) \sum_{j=1}^{l} w_{hj} g_j x_i$$

$$\nabla \gamma_h = -\eta b_h (1-b_h) \sum_{j=1}^{l} w_{hj} g_j$$



第j个输出层神经元的输出：

$$\hat{y}_j = f(\beta_j - \theta_j)$$

第j个输出层神经元的输入：

$$\beta_j = \sum_{h=1}^{q} w_{hj} b_h$$

第h个隐层神经元的输出：

$$b_h = f(\alpha_h - \gamma_h)$$

第h个隐层神经元的输入：

$$\alpha_h = \sum_{i=1}^{d} v_{ih} x_i$$

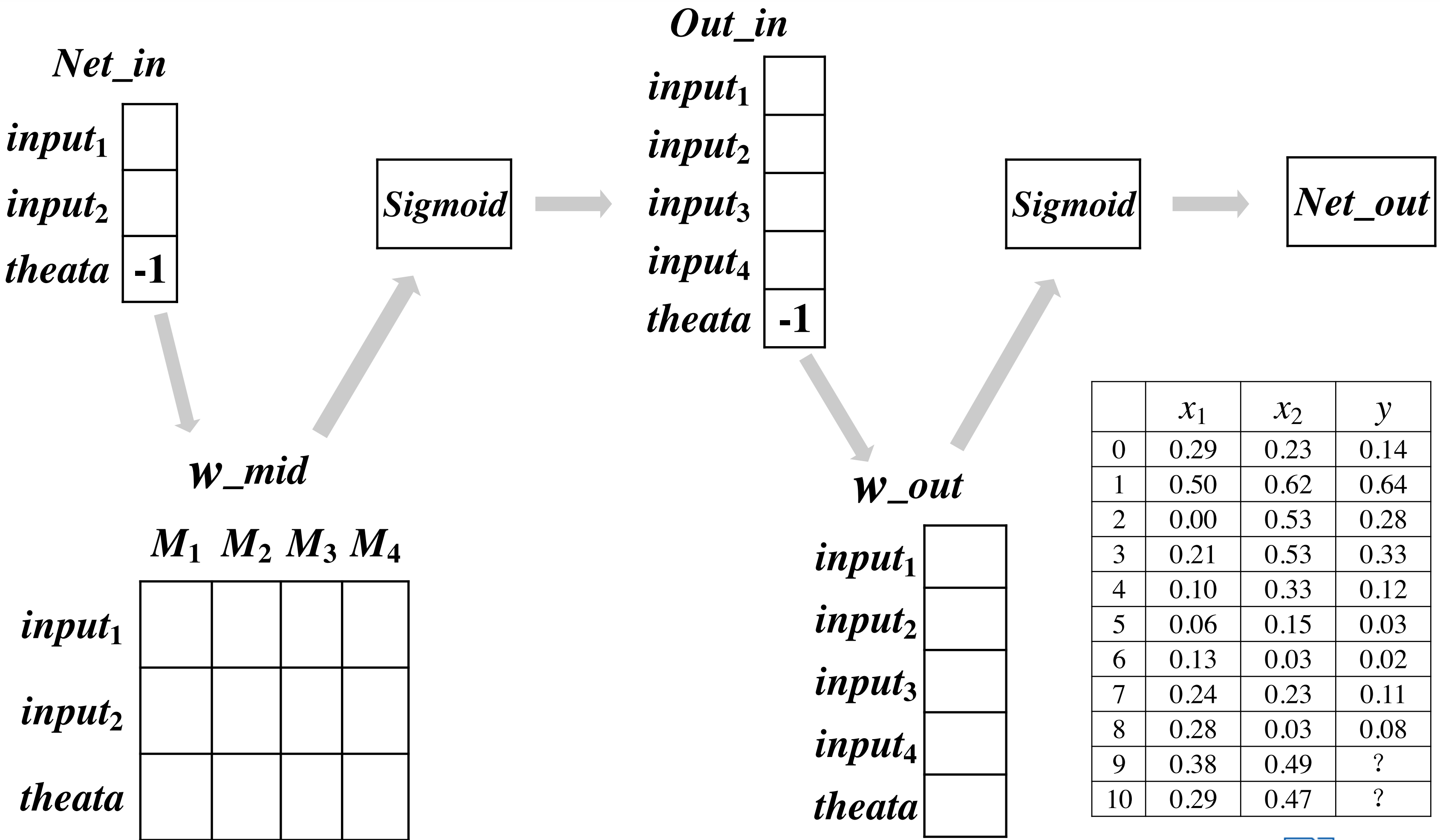
网络训练过程

输入：训练集数据、学习速率*yita*

过程：

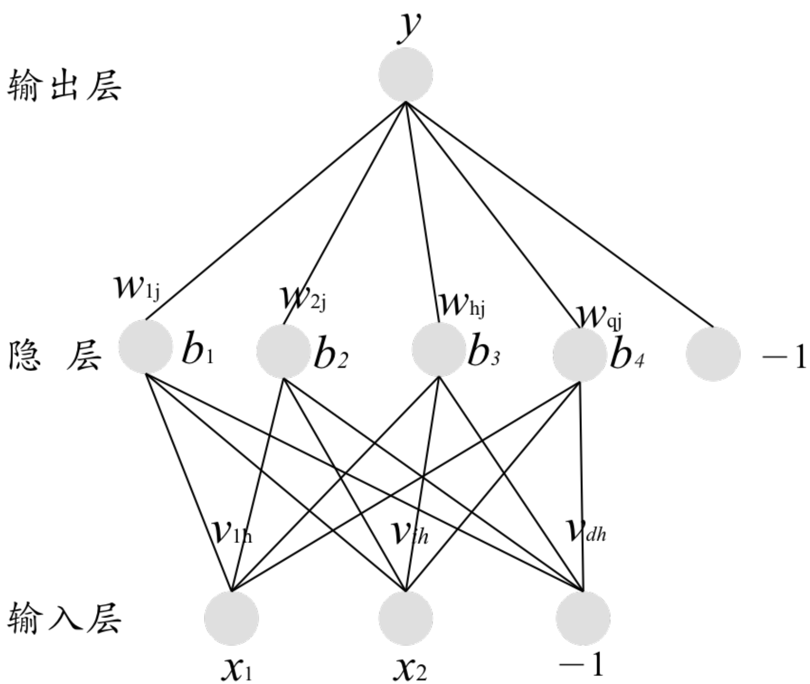
- 在(0,1)范围内随机初始化网络中所有连接权和阈值
- repeat
  - 根据网络输入和当前参数计算网络输出值y
  - 计算输出层神经元梯度项g_j
  - 计算隐层神经元梯度项恶e_h
  - 跟新连接权值和阈值
- until达到停止条件
- 输出：连接权值和阈值

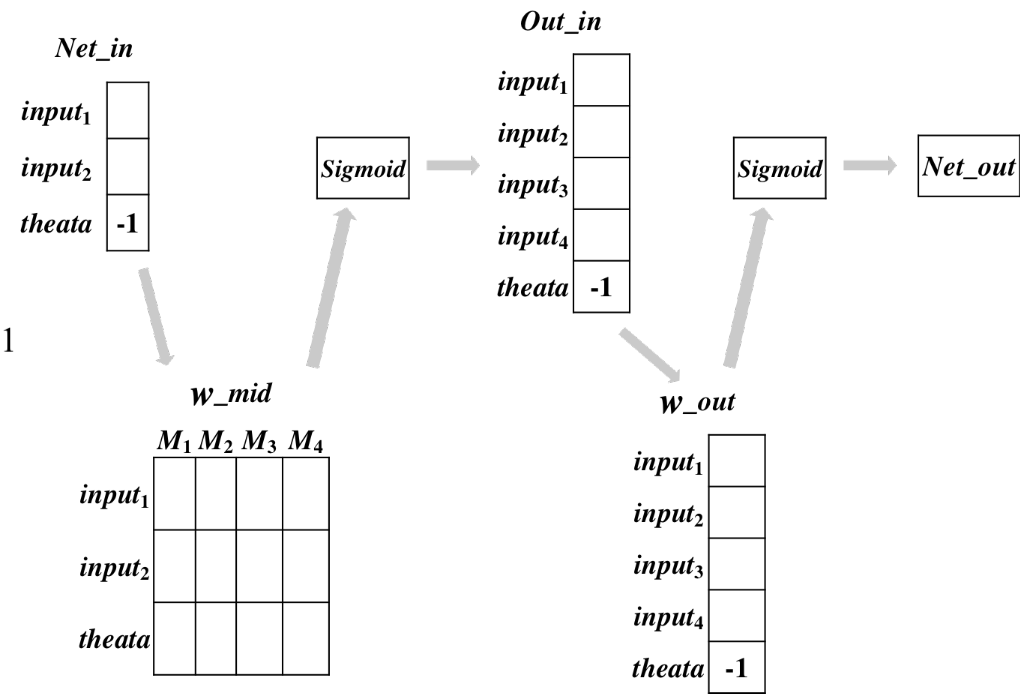
网络训练过程



	x_1	x_2	y
0	0.29	0.23	0.14
1	0.50	0.62	0.64
2	0.00	0.53	0.28
3	0.21	0.53	0.33
4	0.10	0.33	0.12
5	0.06	0.15	0.03
6	0.13	0.03	0.02
7	0.24	0.23	0.11
8	0.28	0.03	0.08
9	0.38	0.49	?
10	0.29	0.47	?

网络训练过程





$$y_i = f\left(\sum_{j=1}^{n} w_{ij} x_j - \theta\right)$$

应用：对iris数据集进行分类

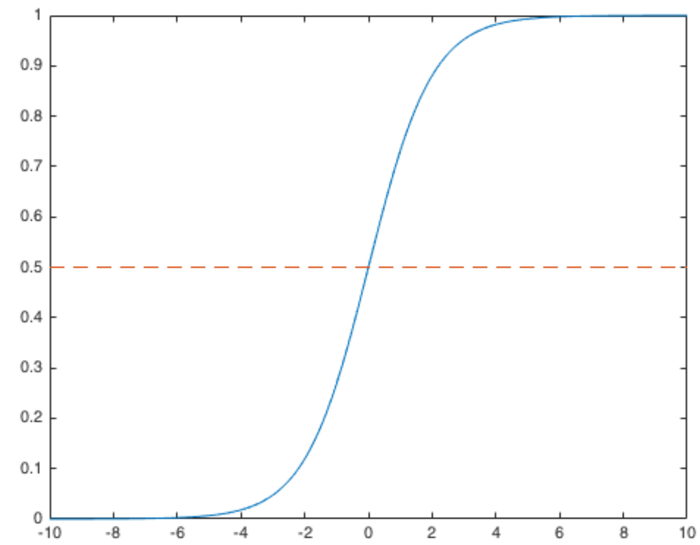
- 训练集：80%，分层抽样
- 测试集：20%，余下

1. 利用80%训练集样本构建一个网络
2. 将20%样本的前4列作为网络输入，输出每个样本的类别
3. 将模型输出结果与实际分类进行比较，对模型进行评价

Python实现

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

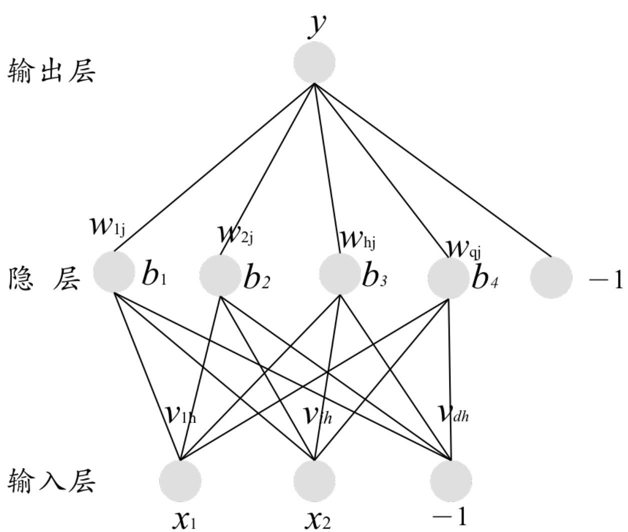
```
def sigmoid(x):  #映射函数
        return 1/(1+math.exp(-x))
import math
import numpy as np
import pandas as pd
from pandas import DataFrame,Series
```

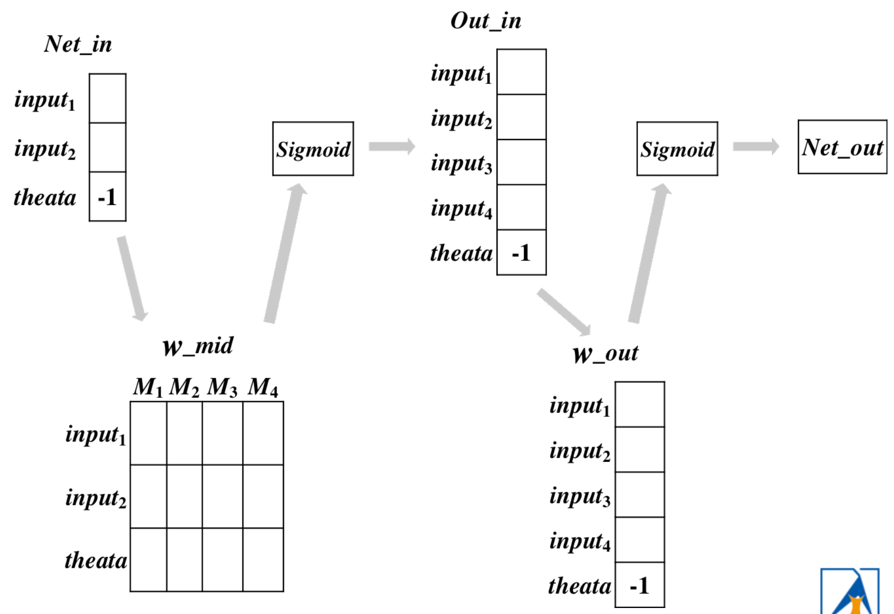


泰迪智能科技
TipDM Intelligent Technology

#中间层神经元输入和输出层神经元输入

```
Net_in = DataFrame(0.6,index=['input1','input2','theata'],columns=['a'])
Out_in = DataFrame(0,index=['input1','input2','input3','input4','theata'],columns=['a'])
Net_in.ix[2,0] = -1
Out_in.ix[4,0] = -1
```

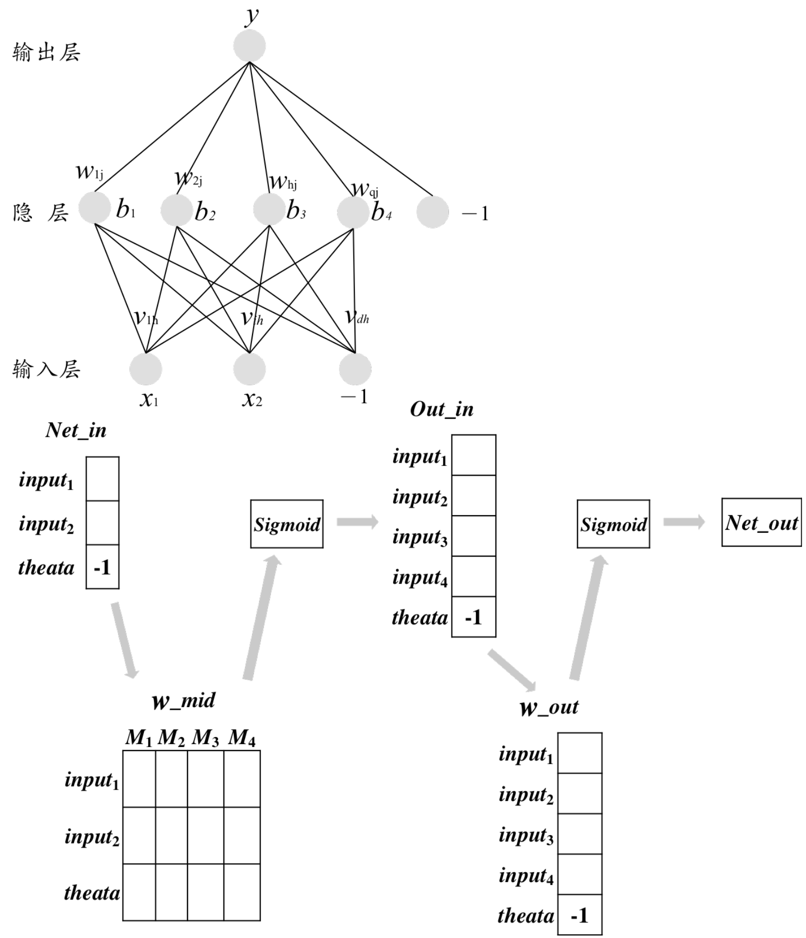




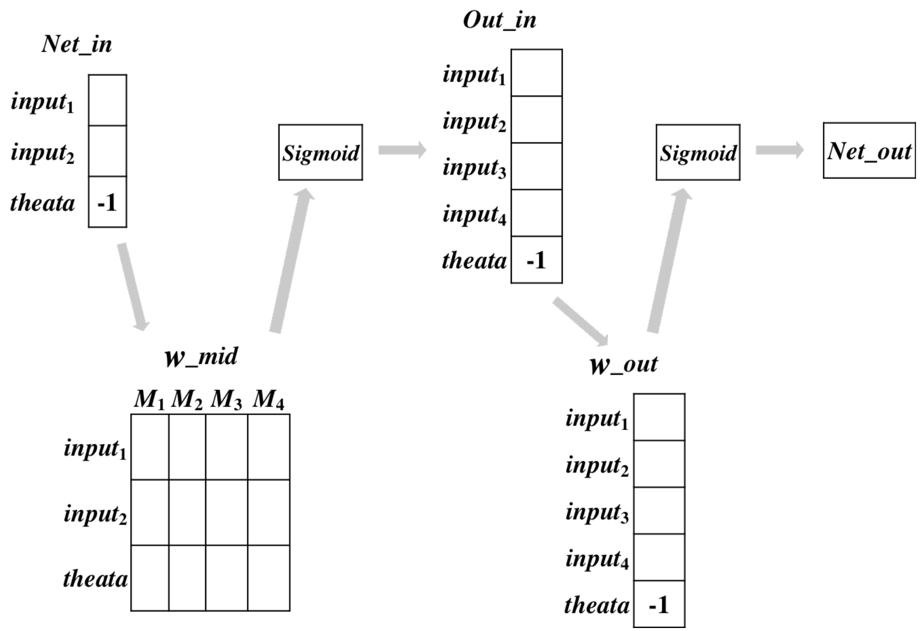
Python实现

#中间层和输出层神经元权值

```
W_mid = DataFrame(0.5,index=['input1','input2','theata'],columns=['mid1','mid2','mid3','mid4'])
W_out = DataFrame(0.5,index=['input1','input2','input3','input4','theata'],columns=['a'])
W_mid_delta = DataFrame(0,index=['input1','input2','theata'],columns=['mid1','mid2','mid3','mid4'])
W_out_delta = DataFrame(0,index=['input1','input2','input3','input4','theata'],columns=['a'])
```



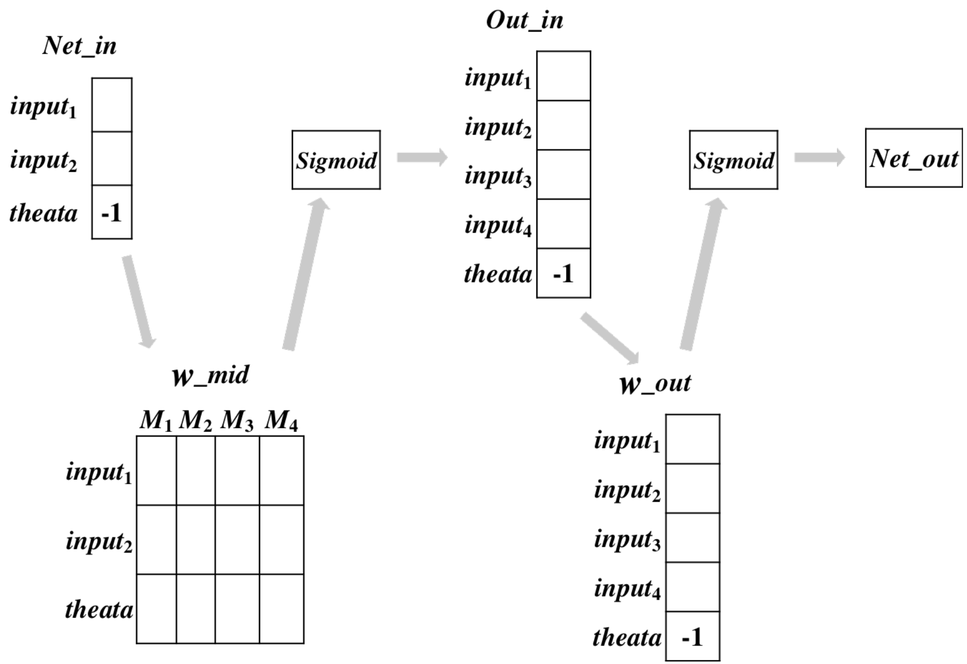
```
#中间层的输出
for i in range(0,4):
    Out_in.ix[i,0] = sigmoid(sum(W_mid.ix[:,i]*Net_in.ix[:,0]))
#输出层的输出/网络输出
res = sigmoid(sum(Out_in.ix[:,0]*W_out.ix[:,0]))
error = abs(res-real)
```



Python实现

$$\Delta w_{hj} = \eta \hat{y}_j (1-\hat{y}_j)(y_j - \hat{y}_j) b_h$$

$$\nabla \theta_j = -\eta \hat{y}_j (1-\hat{y}_j)(y_j - \hat{y}_j)$$

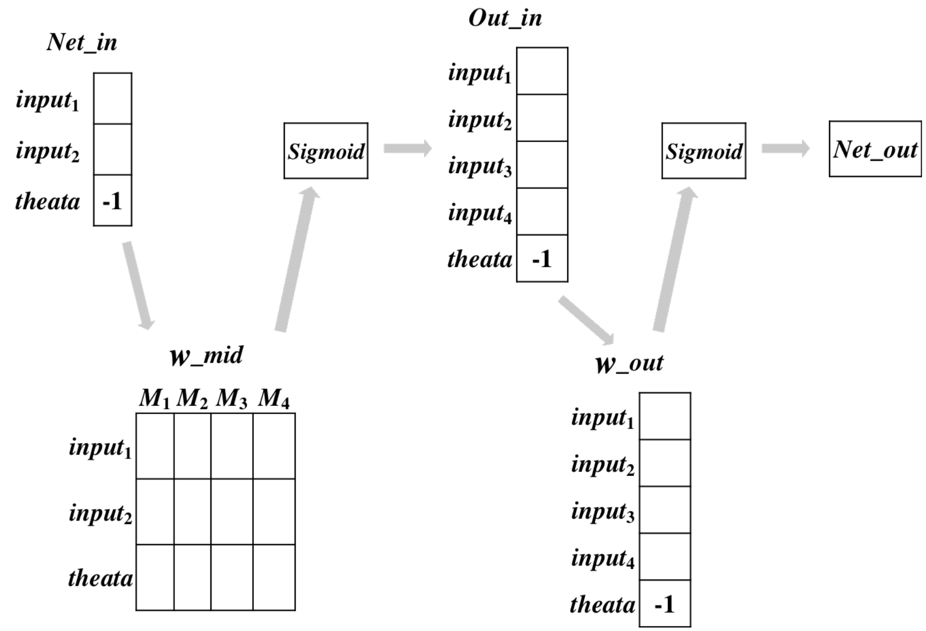



```
#输出层权值变化量
W_out_delta.ix[:,0] = yita*res*(1-res)*(real-res)*Out_in.ix[:,0]
W_out_delta.ix[4,0] = -(yita*res*(1-res)*(real-res))
W_out = W_out + W_out_delta #输出层权值更新
```

Python实现

$$\nabla v_{ih} = \eta b_h (1 - b_h) \sum_{j=1}^{l} w_{hj} g_j x_i$$

$$\nabla \gamma_h = -\eta b_h (1 - b_h) \sum_{j=1}^{l} w_{hj} g_j$$




```
#中间层权值变化量
for i in range(0,4):
    W_mid_delta.ix[:,i] = yita*Out_in.ix[i,0]*(1-Out_in.ix[i,0])*W_out.ix[i,0]*res*(1-res)*(real-res)*Net_in.ix[:,0]
    W_mid_delta.ix[2,i] = -(yita*Out_in.ix[i,0]*(1-Out_in.ix[i,0])*W_out.ix[i,0]*res*(1-res)*(real-res))
W_mid = W_mid + W_mid_delta #中间层权值更新
```

泰迪智能科技 TipDM Intelligent Technology

Thanks

泰迪智能科技
TipDM Intelligent Technology