# BP神经网络

张敏

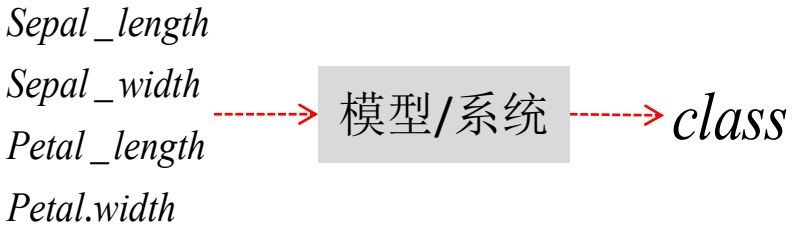
# 神经网络

下列鸢尾花分别属于哪一类：setosa、versicolor、virginica



# 神经网络

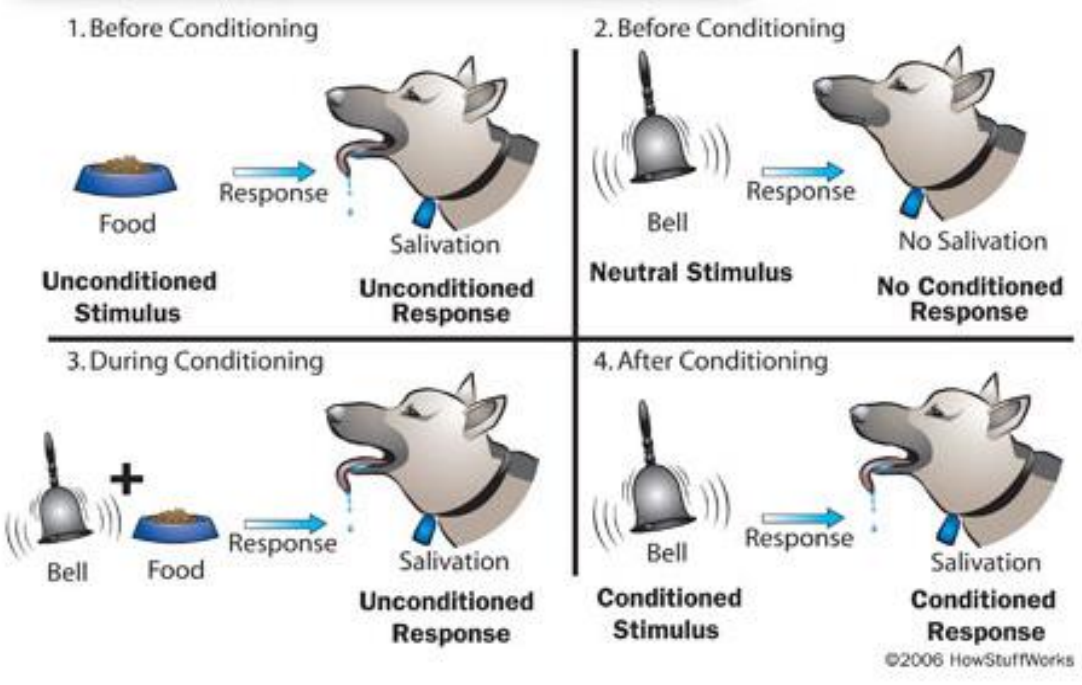
| Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | class |
|---|---|---|---|---|
| 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 4.9 | 3 | 1.4 | 0.2 | setosa |
| 7 | 3.2 | 4.7 | 1.4 | versicolor |
| 6.4 | 3.2 | 4.5 | 1.5 | versicolor |
| 6.3 | 3.3 | 6 | 2.5 | virginica |
| 5.8 | 2.7 | 5.1 | 1.9 | virginica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




$$y = f(x_1, x_2, x_3, x_4, x_5.....)$$

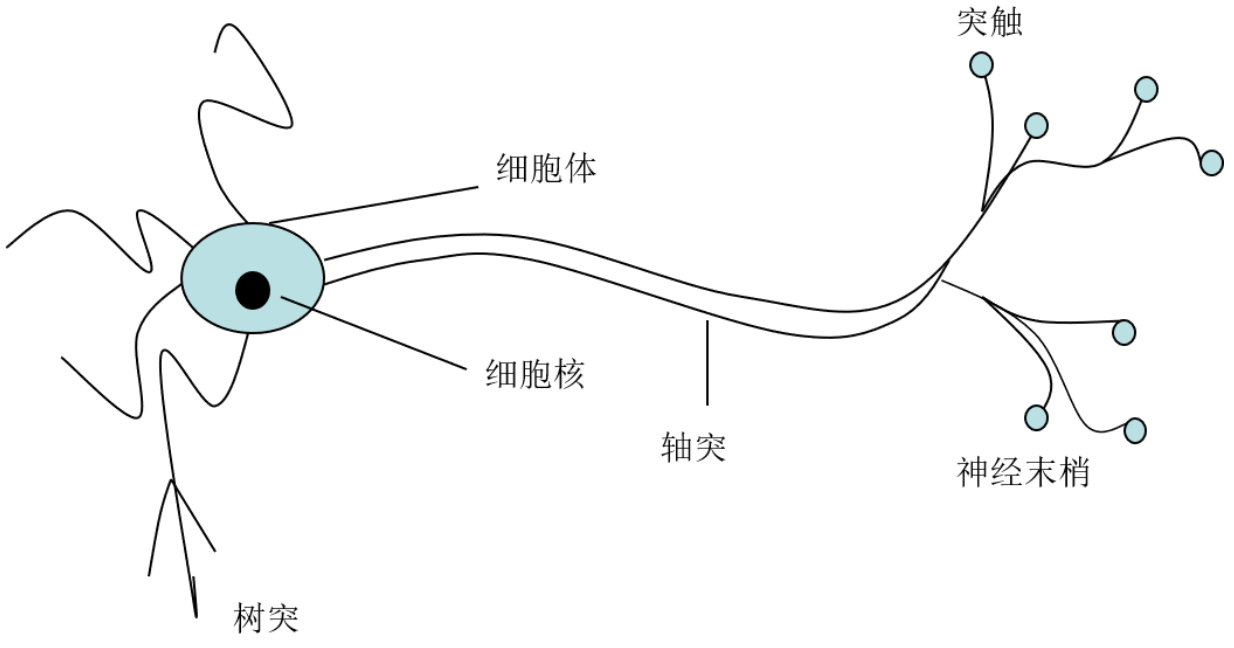
# 神经网络

巴普洛夫关于神经反射的实验



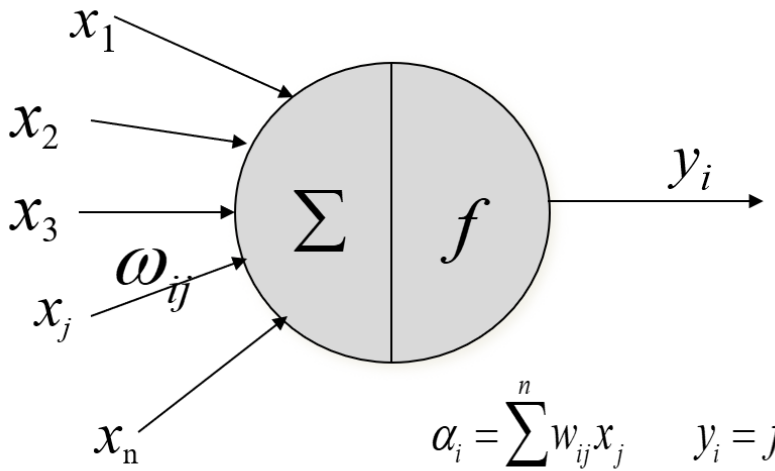
# 神经网络

## 生物神经元结构



# 神经网络

## 数学神经元结构

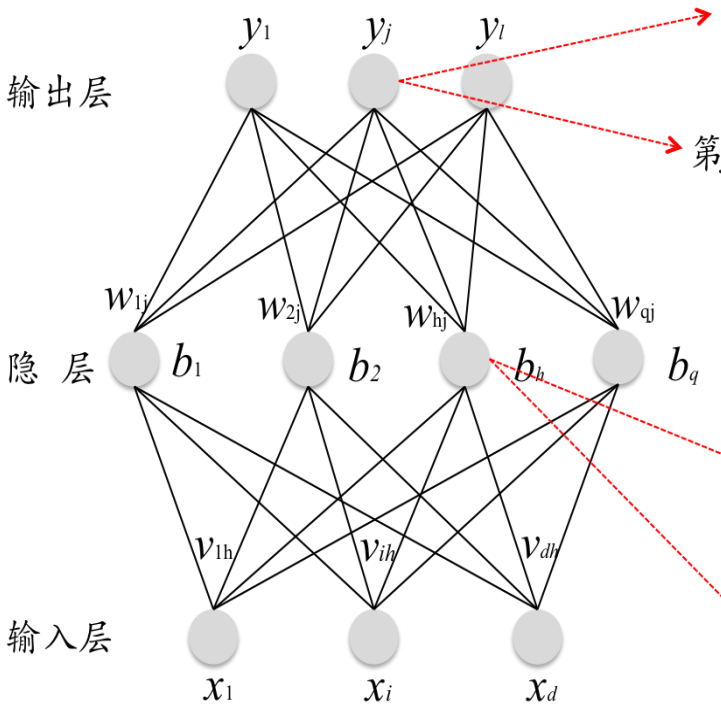



$$\alpha_i = \sum_{j=1}^{n} w_{ij} x_j \qquad y_i = f\left(\sum_{j=1}^{n} w_{ij} x_j - \theta\right)$$

$x_j$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，$q$表示阈值

# 神经网络

## BP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：

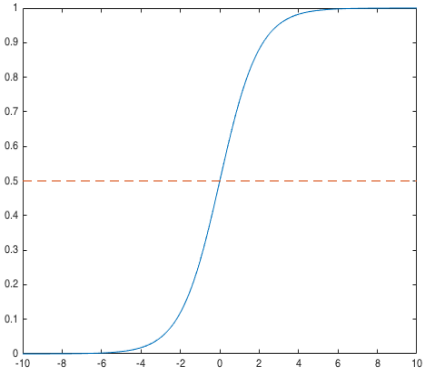
$$\alpha_h = \sum_{i=1}^{d} v_{ih} x_i$$

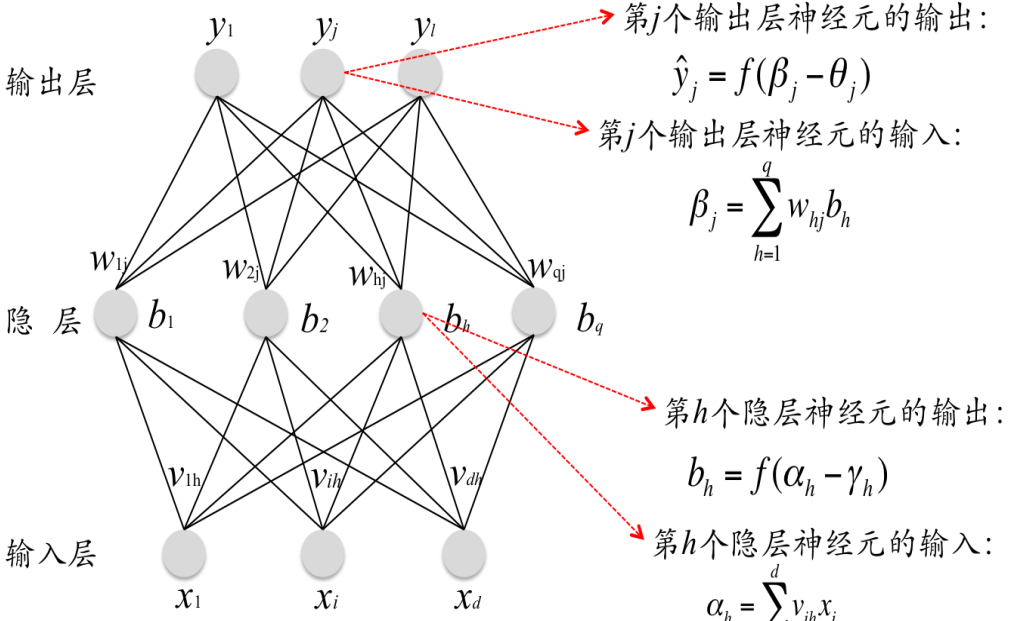
# 神经网络

## BP网络结构

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

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




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：

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

# 神经网络

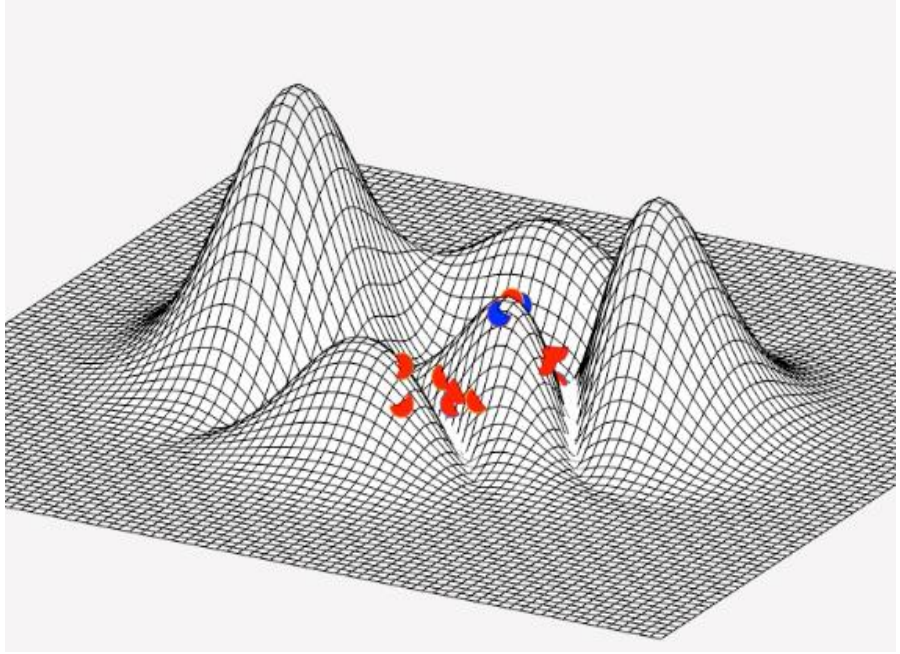
## BP网络结构

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

网络训练目标：

找出合适的权值和阈值，

使得误差 $E$ 最小



# 神经网络

## BP网络结构

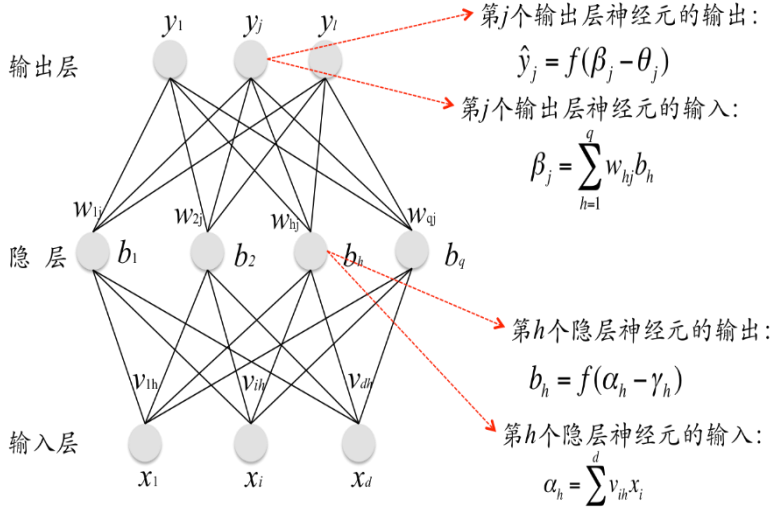
$$f(x)=sigmoid(x)=\frac{1}{1+e^{-x}}$$

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

$$\hat{y}_j=f(b_j-q_j)$$

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

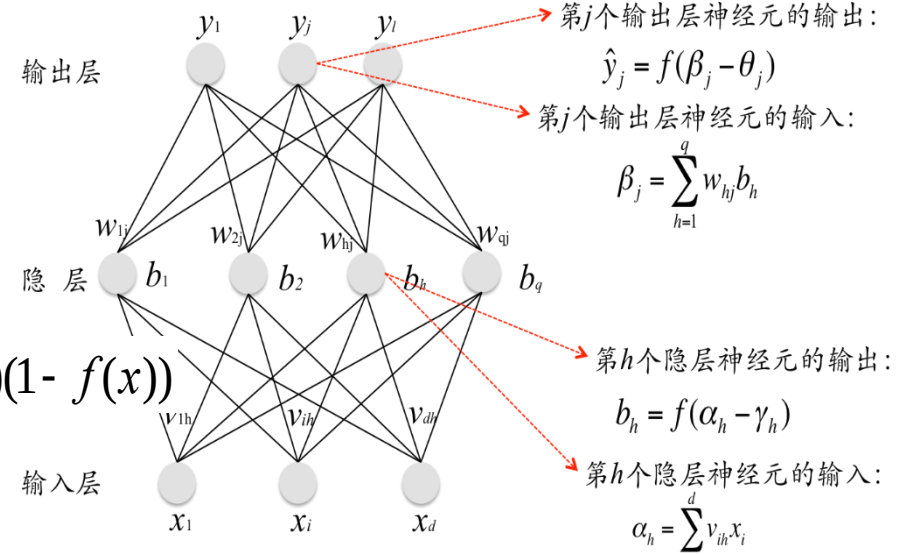
$$\Delta w_{hj}=-h\frac{\partial E}{\partial w_{hj}} \quad \frac{\partial E}{\partial w_{hj}}=\frac{\partial E}{\partial \hat{y}_j}\times\frac{\partial \hat{y}_j}{\partial b_j}\times\frac{\partial b_j}{\partial w_{hj}}$$



# 神经网络

$$\frac{\partial E}{\partial w_{hj}} = \frac{\partial E}{\partial \hat{y}_j} \times \frac{\partial \hat{y}_j}{\partial b_j} \times \frac{\partial b_j}{\partial w_{hj}}$$

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




$$\frac{\partial \hat{y}_j}{\partial b_j} = f'(b_j - q_j) \qquad f'(x) = f(x)(1 - f(x))$$

$$= f(b_j - q_j)(1 - f(b_j - q_j))$$

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

$$g_j = -\frac{\partial E}{\partial \hat{y}_j} \times \frac{\partial \hat{y}_j}{\partial b_j}$$

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

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

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

$$= -h\frac{\partial E}{\partial \hat{y}_j} \cdot \frac{\partial \hat{y}_j}{\partial b_j} \cdot \frac{\partial b_j}{\partial w_{hj}} \longrightarrow \Delta w_{hj} = h\hat{y}_j(1 - \hat{y}_j)(y_j - \hat{y}_j)b_h$$

$$= hg_j b_h$$

# 神经网络

## BP网络结构

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

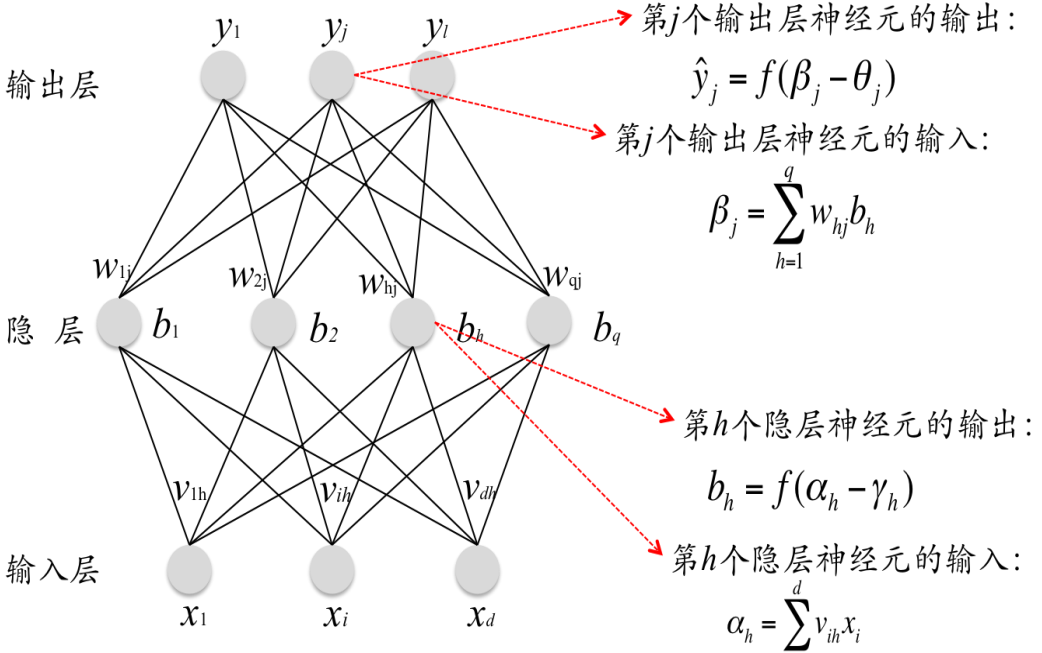
$$= -\eta \frac{\partial E}{\partial \hat{y}_j} \cdot \frac{\partial \hat{y}_j}{\partial b_j} \cdot \frac{\partial b_j}{\partial w_{hj}}$$

$$= \eta g_j b_h$$

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

$$\Delta q_j = -\eta g_j$$

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




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：

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

# 神经网络

## BP网络结构

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

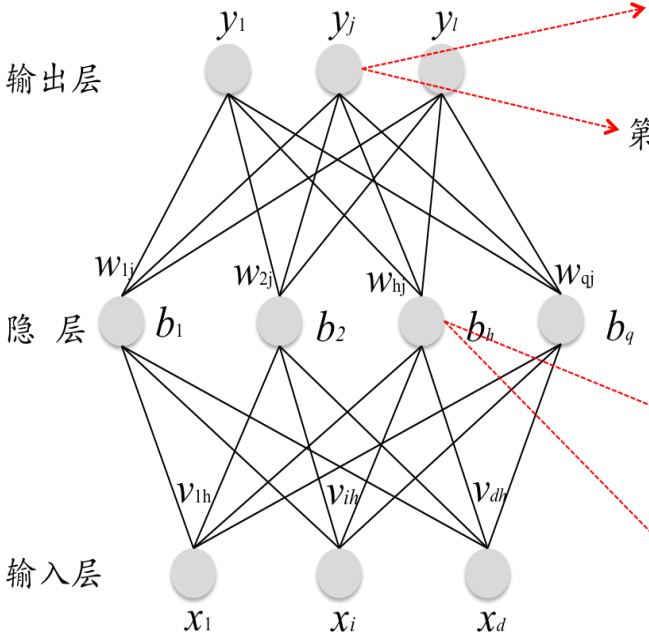
$$= -\eta \frac{\partial E}{\partial b_h} \cdot \frac{\partial b_h}{\partial a_h} x_i$$

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

$$\Delta g_h = -\eta e_h$$

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

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




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：

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

# 神经网络

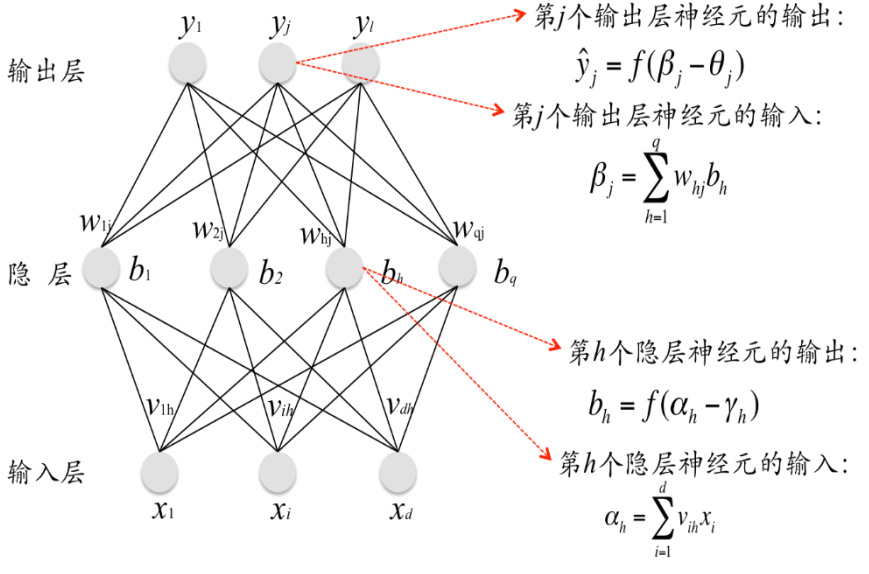
## BP网络结构

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

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

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

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



# 神经网络

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$e_h$
  - 更新连接权值和阈值
- until达到停止条件
- 输出：连接权值和阈值

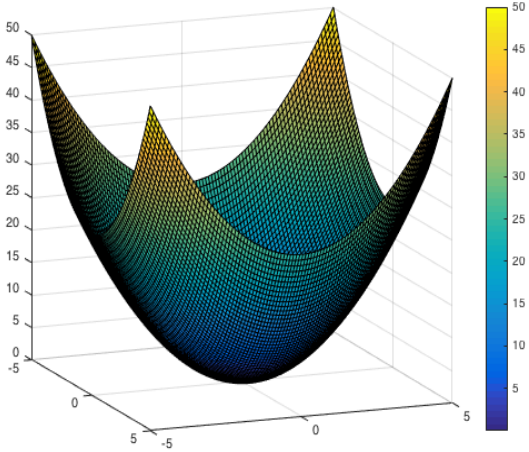
# 神经网络

## BP神经网络实现

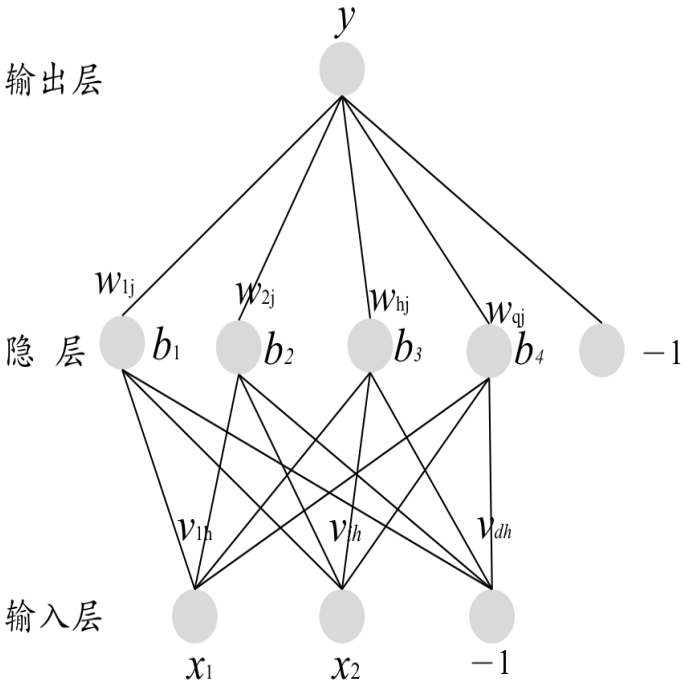
$$y = x_1^2 + x_2^2$$

训练集数据：BPdata_tr.txt

测试集数据：BPdata_te.txt

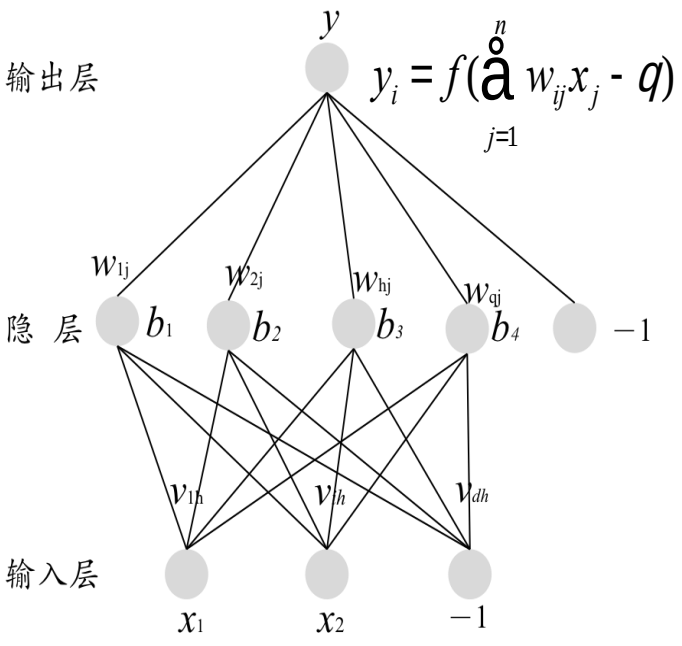


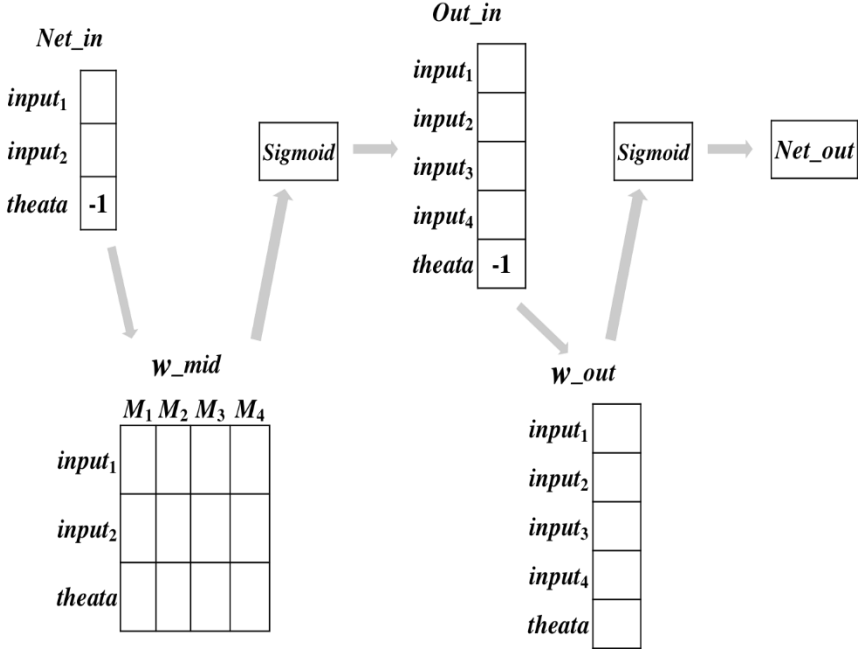
| | $x_1$ | $x_2$ | $y$ |
|---|---|---|---|
| 0 | 0.29 | 0.23 | 0.14 |
| 1 | 0.50 | 0.62 | 0.64 |
| 2 | 0.00 | 0.53 | 0.28 |
| 3 | 0.21 | 0.53 | 0.33 |
| 4 | 0.10 | 0.33 | 0.12 |
| 5 | 0.06 | 0.15 | 0.03 |
| 6 | 0.13 | 0.03 | 0.02 |
| 7 | 0.24 | 0.23 | 0.11 |
| 8 | 0.28 | 0.03 | 0.08 |
| 9 | 0.38 | 0.49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



# 神经网络

## BP神经网络实现

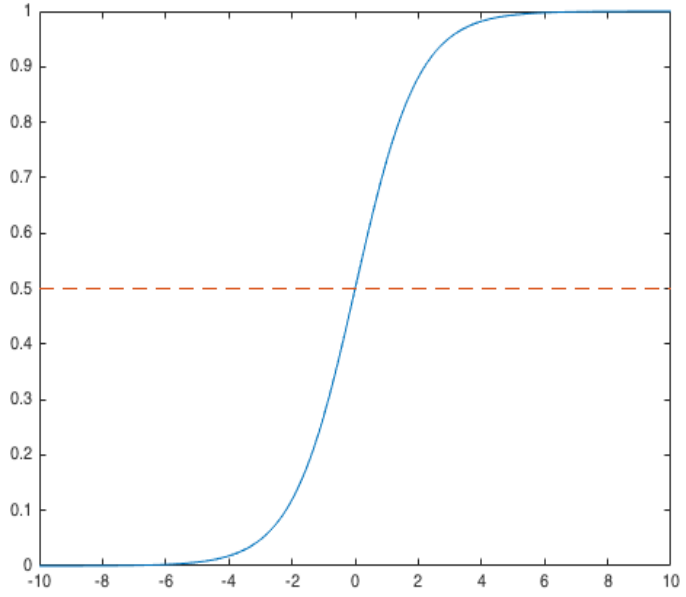


# 神经网络

## 映射函数

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




```
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,Seres
```

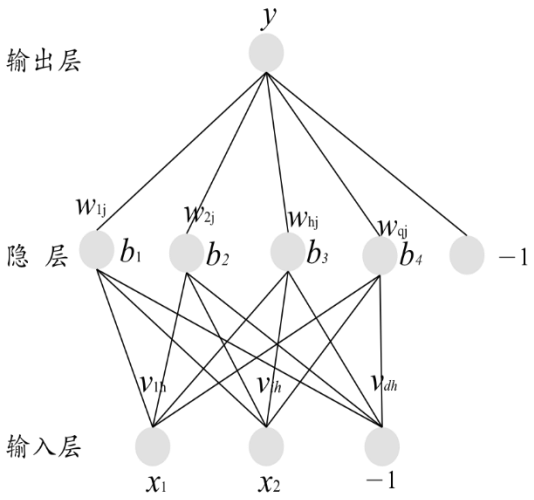
# 神经网络

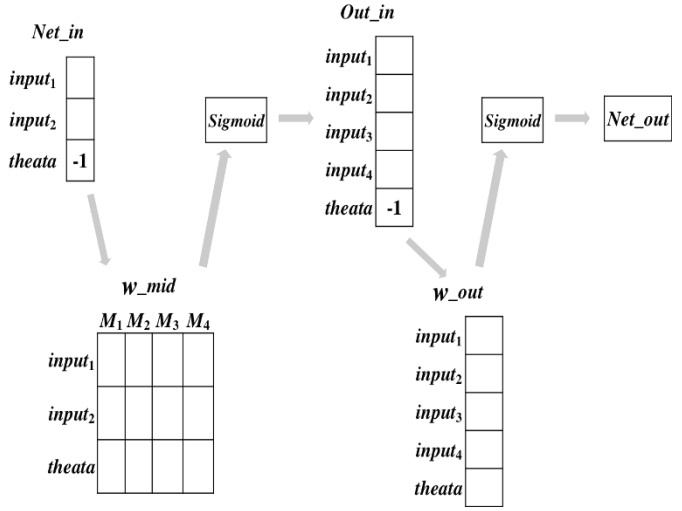
## 中间层神经元输入和输出层神经元输入

#中间层神经元输入和输出层神经元输入

Net_in = np.array([0,0,-1])

Out_in = np.array([0,0,0,0,-1])



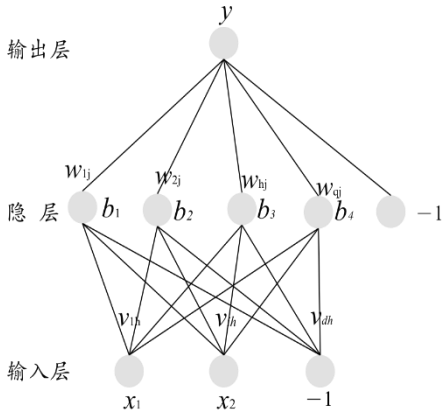



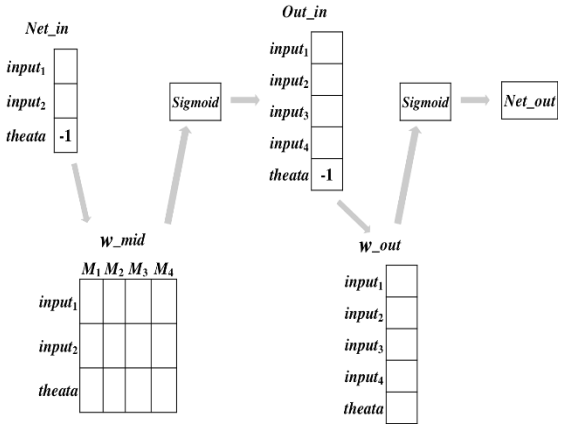
# 神经网络

## 中间层和输出层神经元权值及其变化量

#中间层和输出层神经元权值及其变化量

```
w_mid = np.zeros([3,4])
w_out = np.array([0.3,0.3,0.3,0.3,0.3])
delta_w_mid = np.zeros([3,4])
delta_w_out = np.array([0,0,0,0,0])
```



# 神经网络

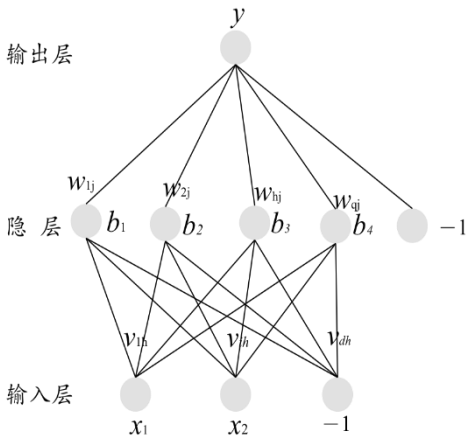
## 中间层的输出

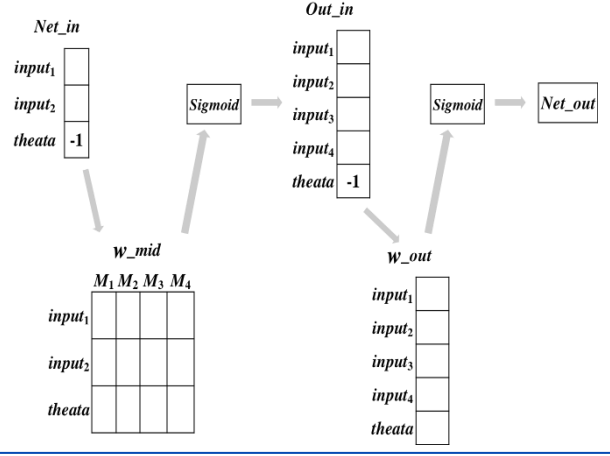
#中间层的输出

```
for i in range(4):
    Out_in[i] = sigmoid(sum(w_mid[:,i]*Net_in))
```

#输出层的输出/网络输出

```
res = sigmoid(sum(Out_in*w_out))
error = abs(res-real)
```



# 神经网络

## 输出层权值变化量

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

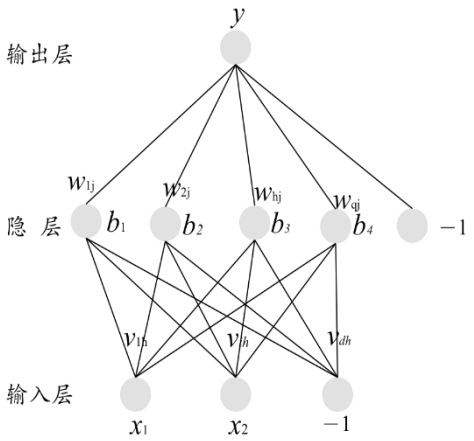
$$\Delta q_j = -h\hat{y}_j(1-\hat{y}_j)(y_j-\hat{y}_j)$$

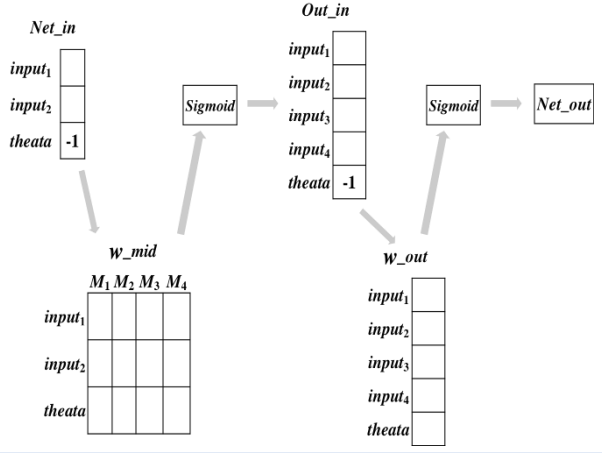
#输出层权值变化量

delta_w_out = yita*res*(1-res)*(real-res)*Out_in

delta_w_out[4] = -(yita*res*(1-res)*(real-res))

w_out = w_out+delta_w_out



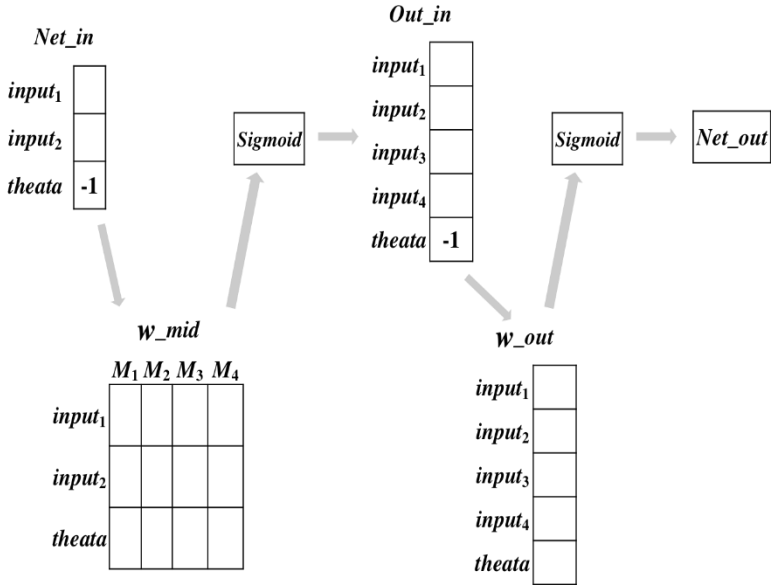



# 神经网络

## 中间层权值变化量

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

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

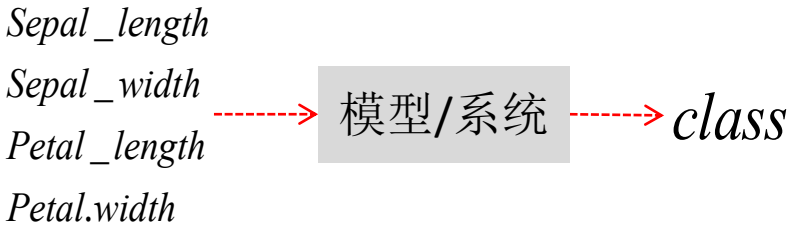



#中间层权值变化量

```
for i in range(4):
    delta_w_mid[:,i] = yita*Out_in[i]*(1-Out_in[i])*w_out[i]*res*(1-res)*(real-res)*Net_in
    delta_w_mid[2,i] = -(yita*Out_in[i]*(1-Out_in[i])*w_out[i]*res*(1-res)*(real-res))
w_mid = w_mid+delta_w_mid
```

# 神经网络

| 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 | ? |




$$y = f(x_1, x_2, x_3, x_4, x_5.....)$$

# 神经网络

代码实现

Python（sklearn）

- Net = MLPClassifier(hidden_layer_sizes=10,max_iter=1000).fit(tr_data.ix[:,0:6],tr_data.ix[:,6])
- res = Net.predict(te_data.ix[:,0:6])

R（nnet）

- nnet(x, y, size, softmax = FALSE, maxit = 100)

# Thank you!