人工神经网络

Artificial Neural Network：ANN

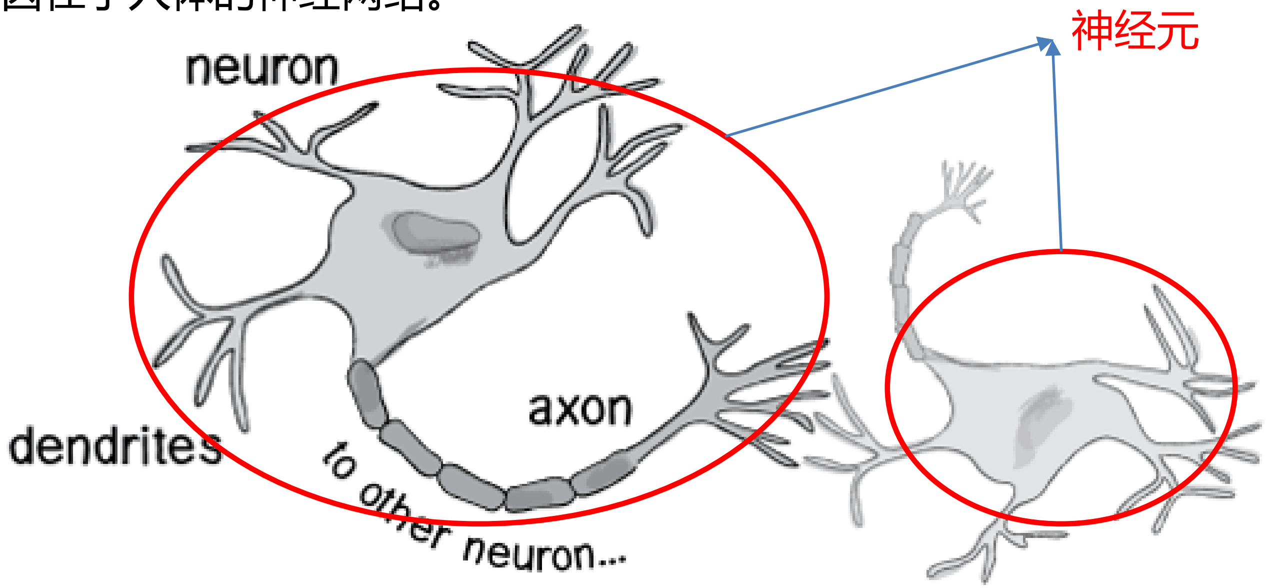
01

PART ONE

ANN的起源和结构

起源

　　历史上，科学家一直希望模拟人的大脑，造出可以思考的机器。人为什么能够思考？科学家发现，原因在于人体的神经网络。



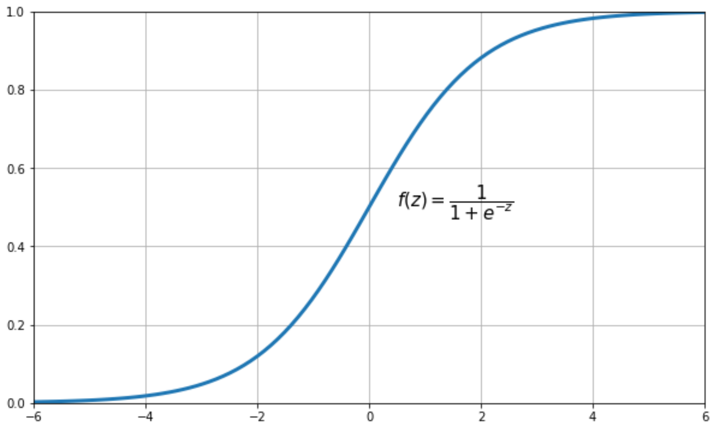
- 外部刺激通过神经末梢，转化为电信号，转导到神经细胞（又叫神经元）。
- 无数神经元构成神经中枢。
- 神经中枢综合各种信号，做出判断。
- 人体根据神经中枢的指令，对外部刺激做出反应。

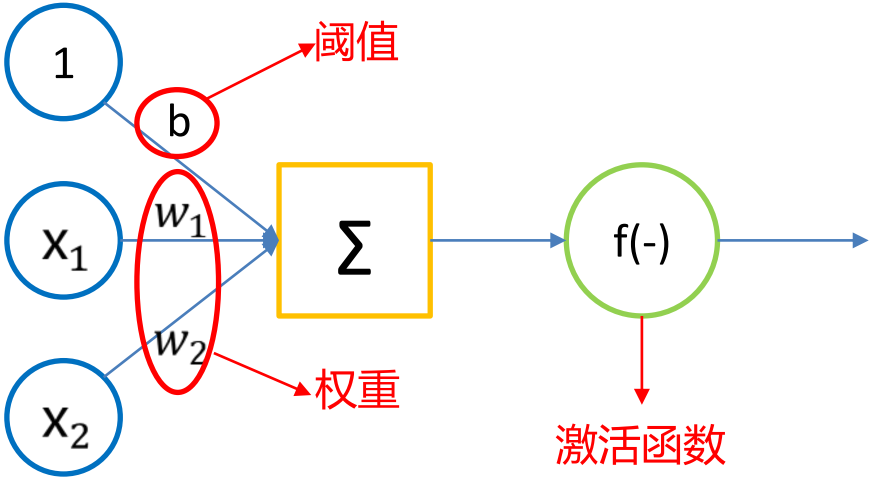
人造神经元-感知器

逻辑回归 → 感知器

$$z = \theta_0 + \theta_1 x_1 + \theta x_2$$

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



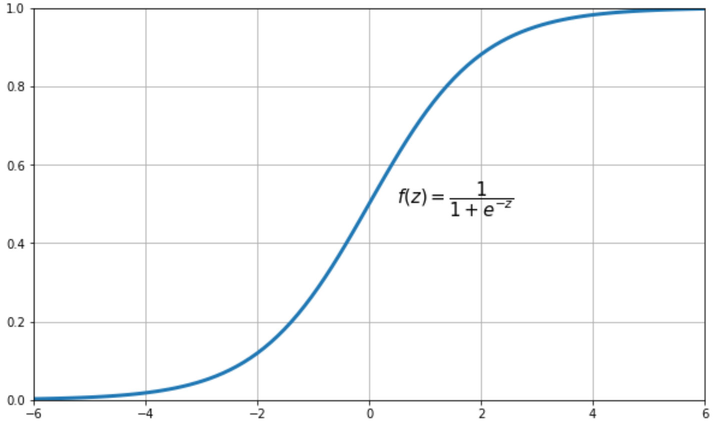


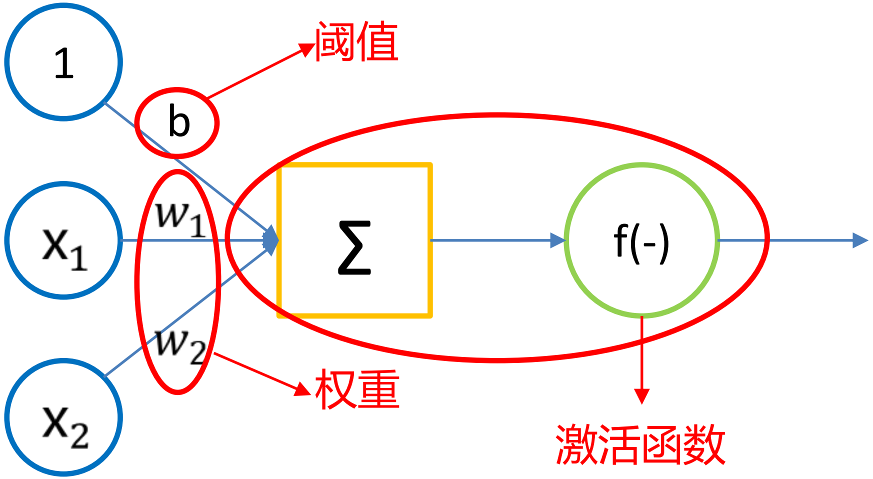
人造神经元-感知器

逻辑回归 → 感知器

$$z = \theta_0 + \theta_1 x_1 + \theta x_2$$

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

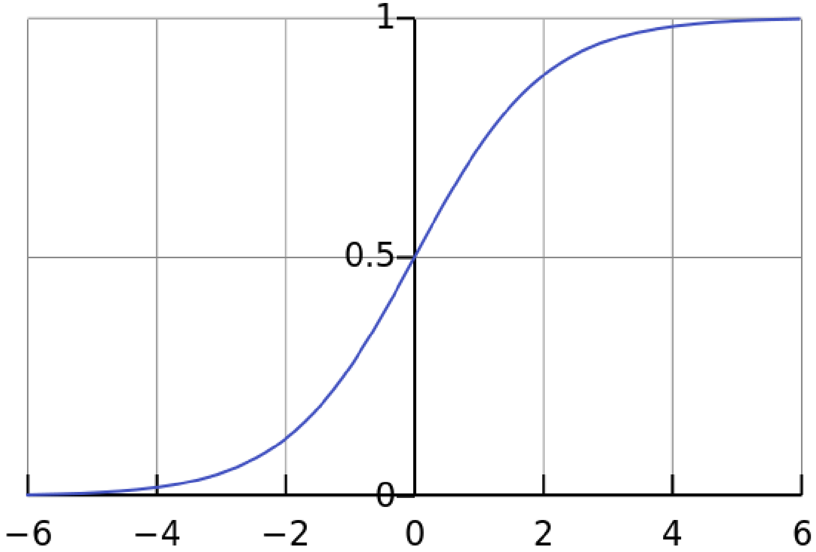




激活函数

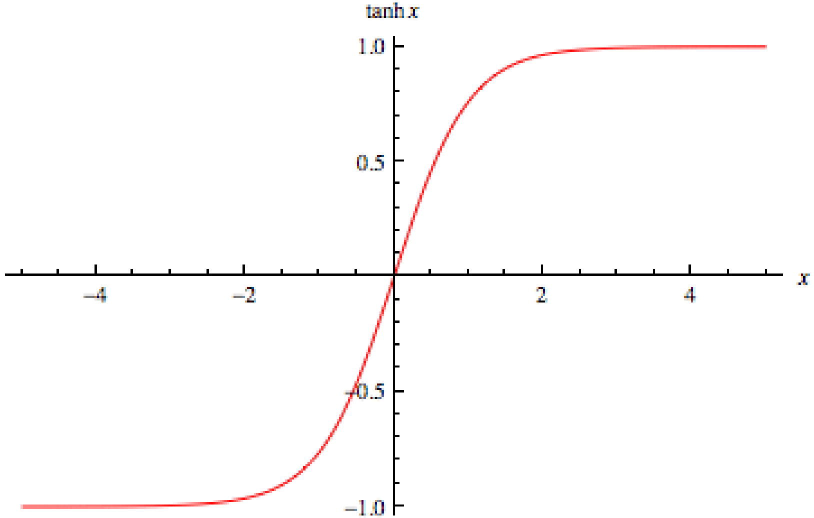
- S函数(Sigmoid)

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

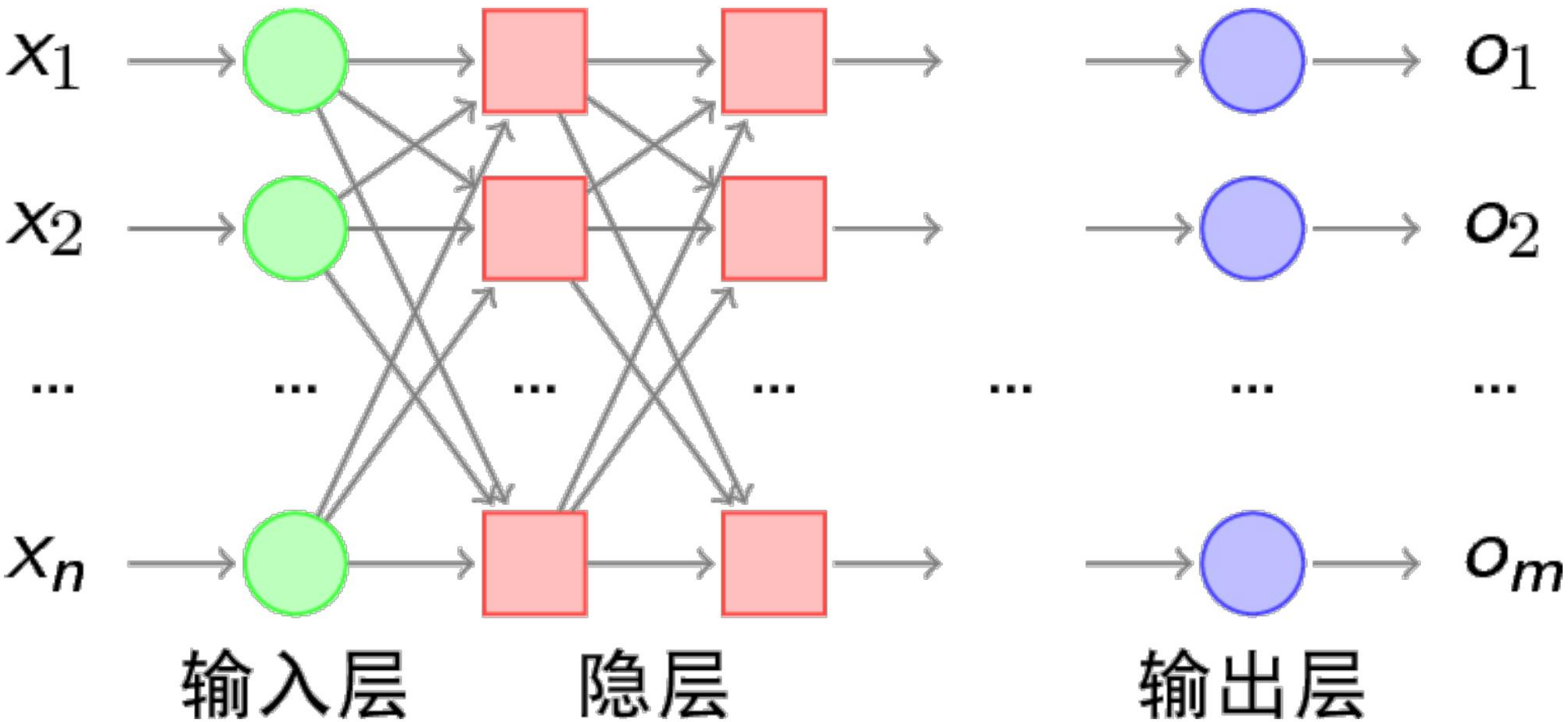


- 双S函数

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

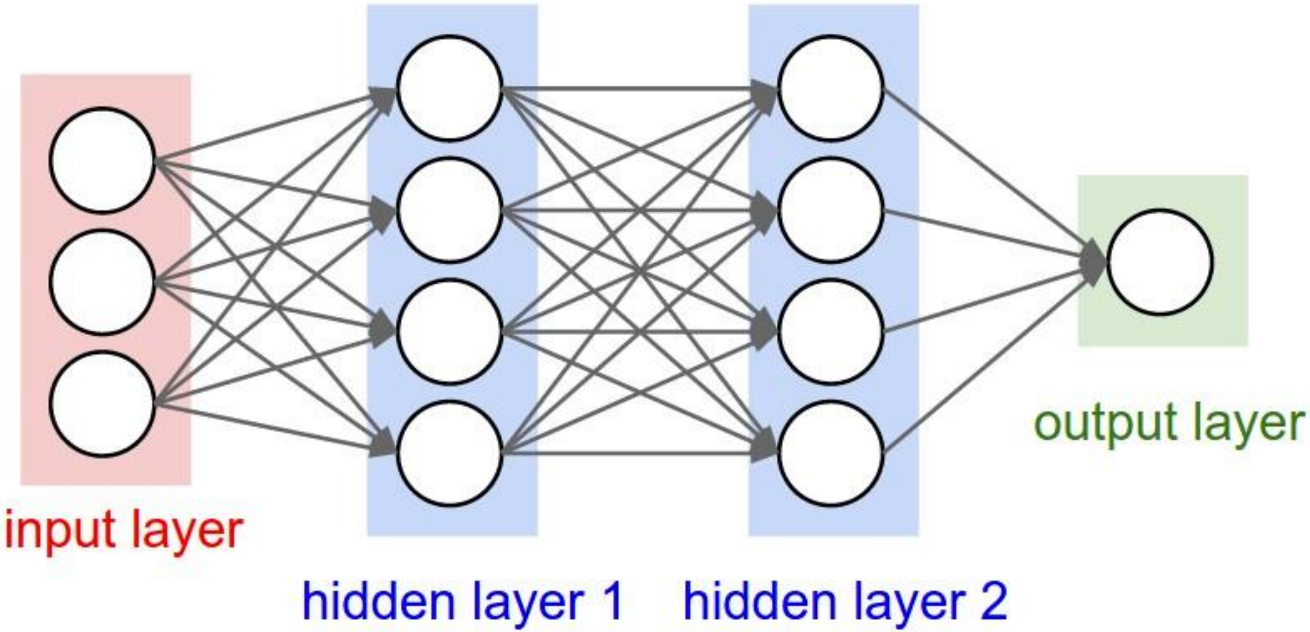


神经网络结构



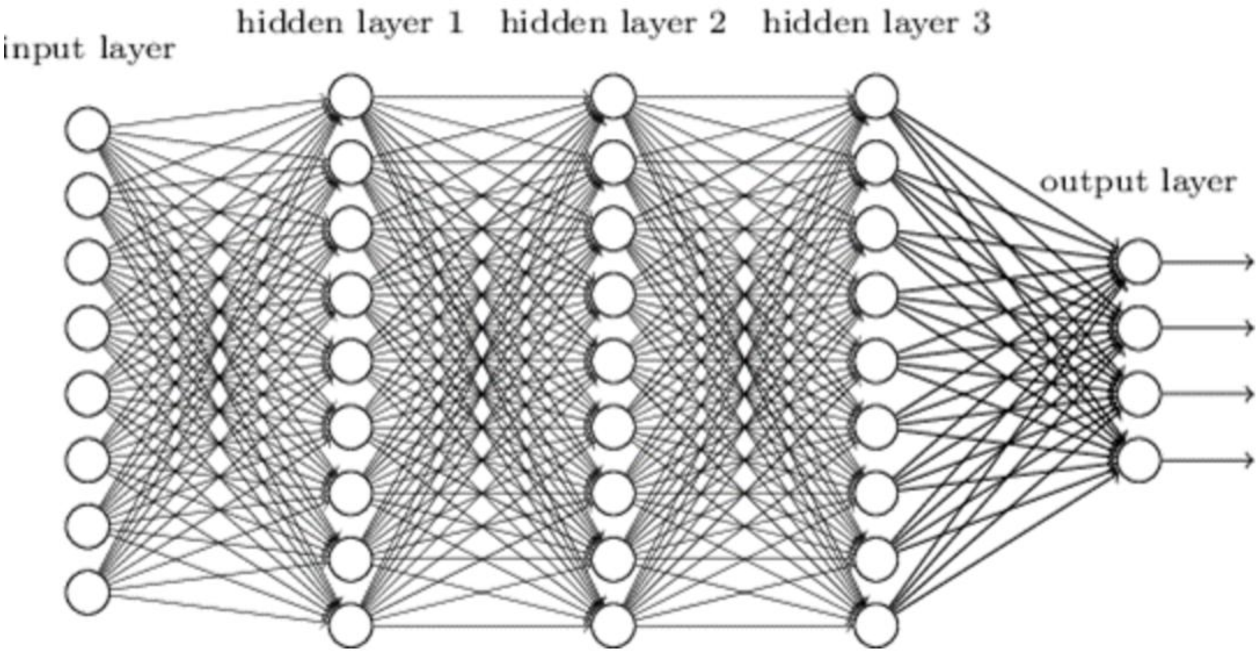
浅层网络

添加少量隐层 ⟶ 浅层神经网络



深度神经网络

增多中间隐层 ⟶ 深度神经网络(DNN)



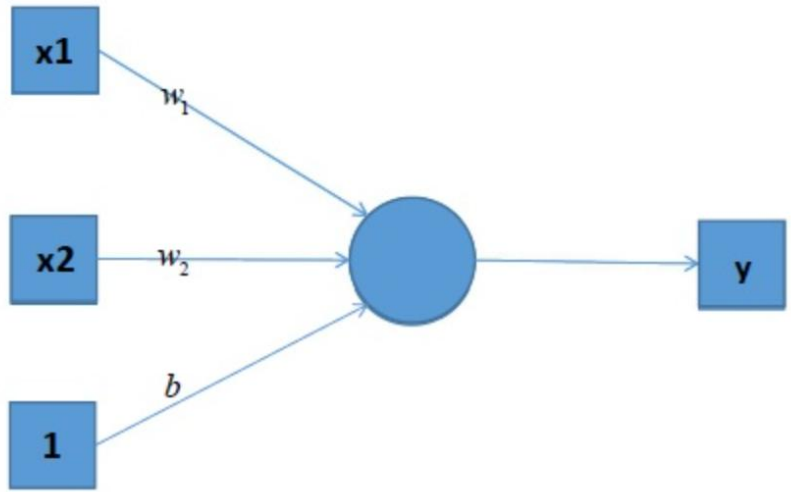
02

PART TWO

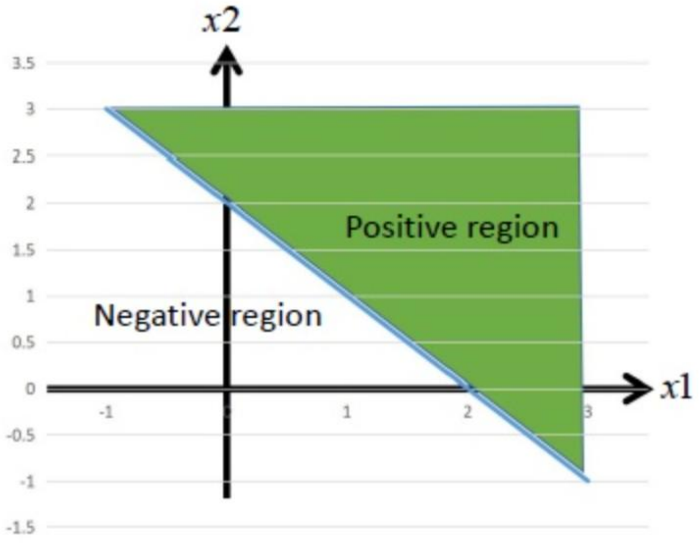
线性逼近

线性分割

单层的感知机，也是我们最常用的神经网络组成单元啦. 用它可以划出一条线，把平面分割开。



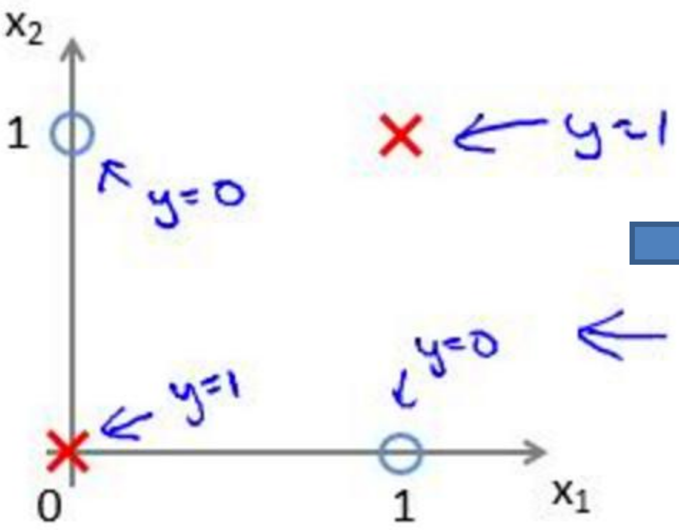
$$y = w_1 x_1 + w_2 x_2 + b$$

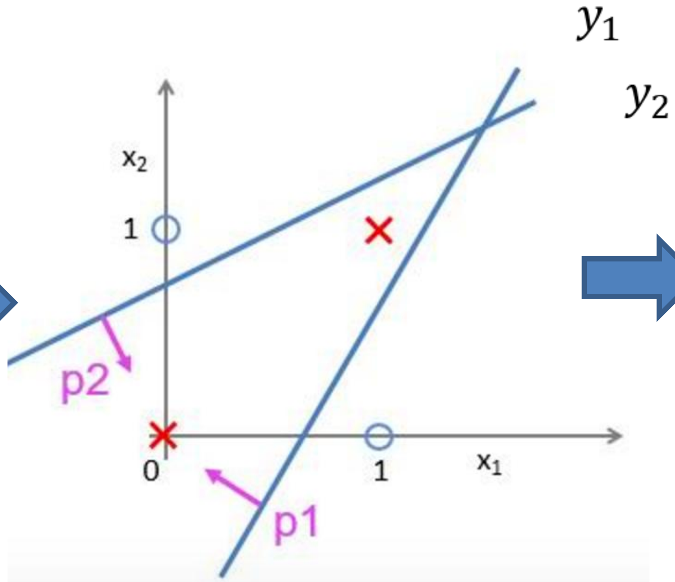


$$w_1 = 1, w_2 = 1, b = -2$$

非线性问题

- 异或问题

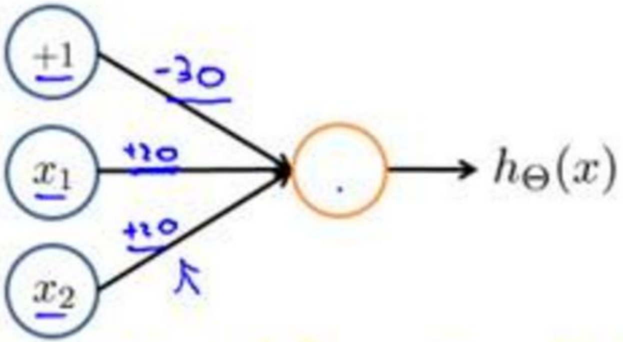




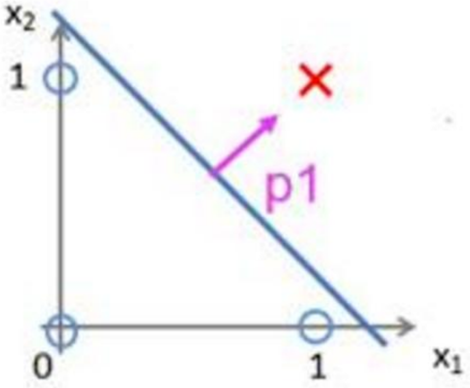
逻辑与

$x_1, x_2 \in \{0, 1\}$

$y = x_1 \text{ AND } x_2$

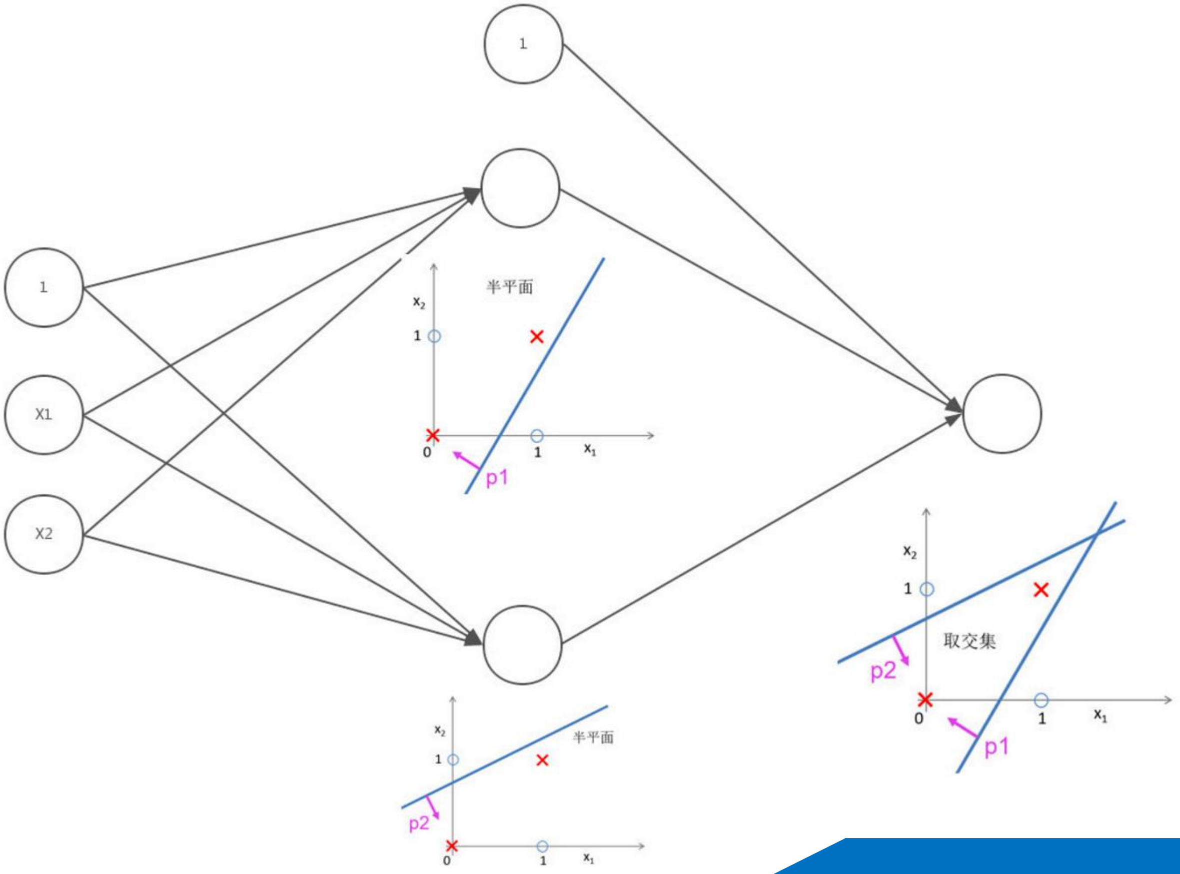


$h_\Theta(x) = g(-30 + 20x_1 + 20x_2)$



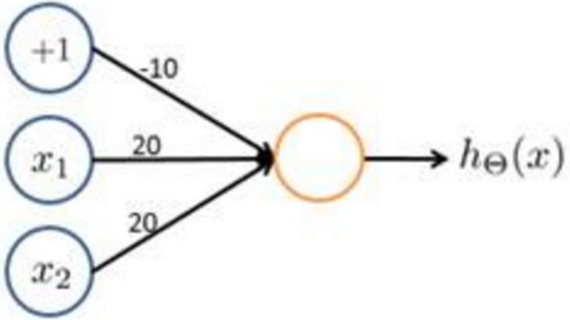
x_1	x_2	$h_\Theta(x)$
0	0	0
0	1	0
1	0	0
1	1	1

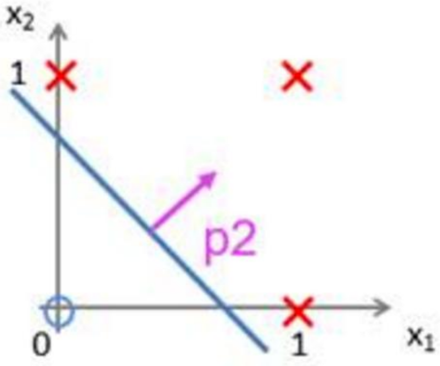
最优化目标



逻辑或

Example: OR function

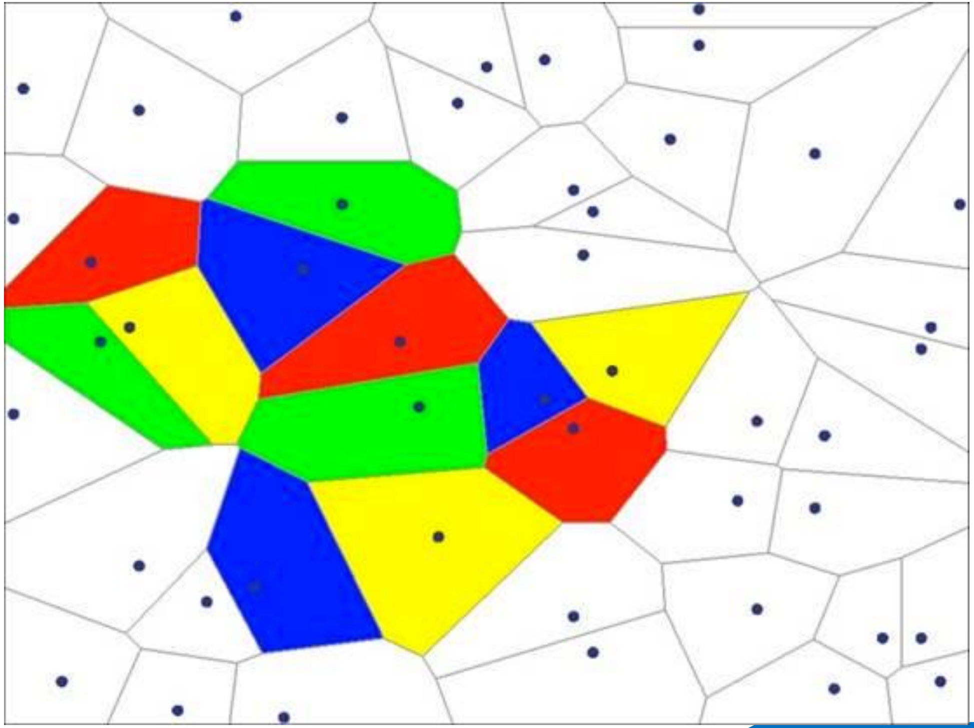




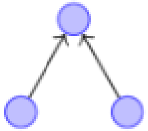
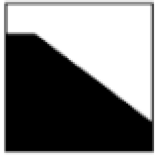
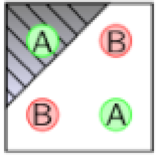
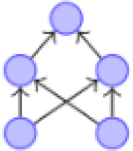
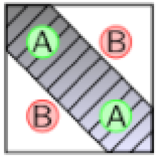
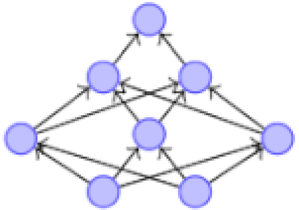
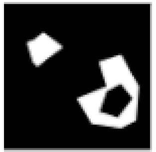
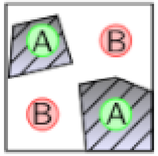
x_1	x_2	$h_\Theta(x)$
0	0	0
0	1	1
1	0	1
1	1	1

线性组合

- 对线性分类器的『与』和『或』的组合
- 完美对平面样本点分布进行分类



最大间隔

结构	决策区域类型	区域形状	异或问题
无隐层 	由一超平面分成两个		
单隐层 	开凸区域或闭凸区域		
双隐层 	任意形状（其复杂度由单元数目确定）		

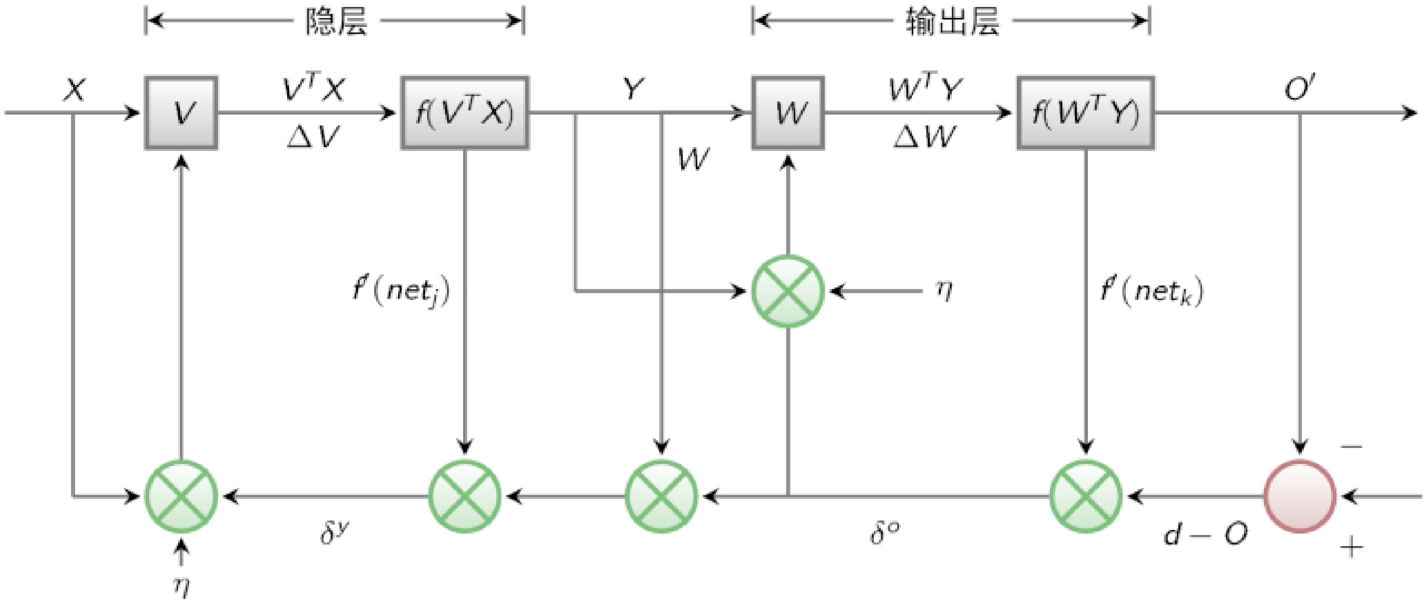
03

PART THREE

训练参数

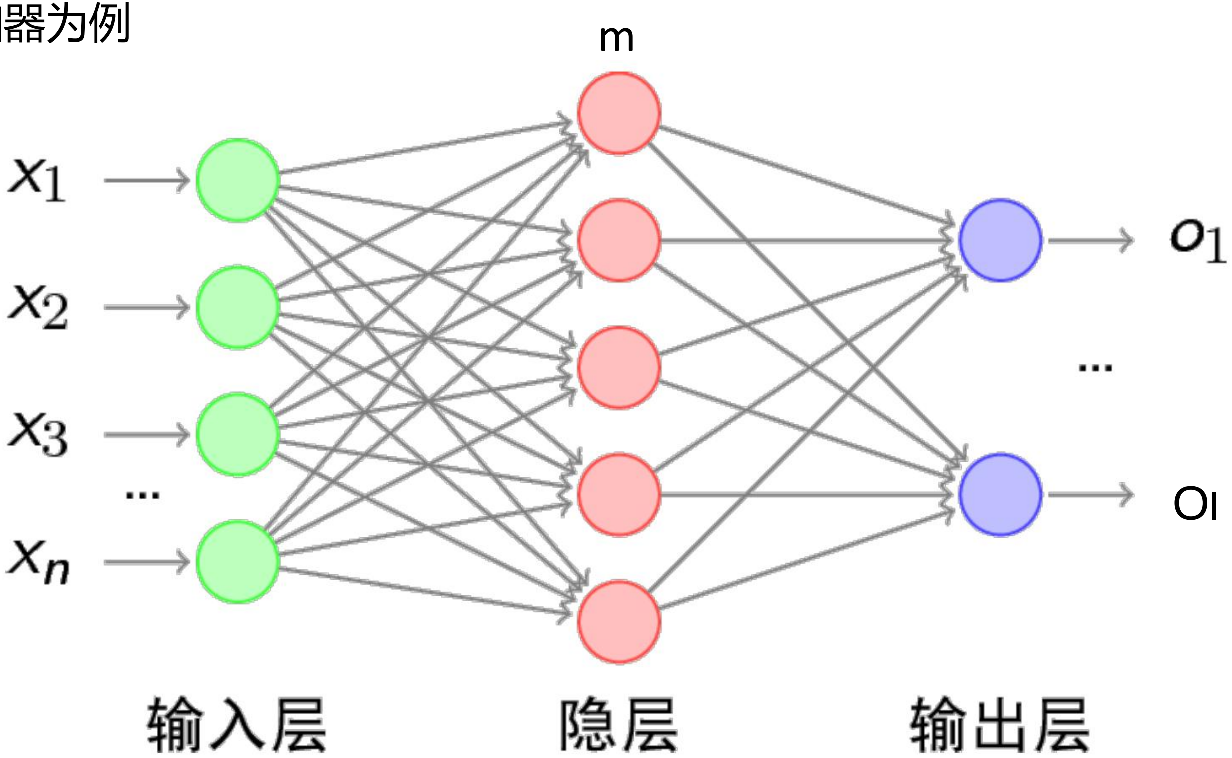
BP算法

- “正向传播”求损失，“反向传播”回传误差
- 根据误差信号修正每层的权重



BP算法

- BP算法，也叫δ算法
- 以3层的感知器为例



BP算法

- **输出层**

$$E = \frac{1}{2}(d - O)^2 \qquad = \frac{1}{2}\sum_{\kappa=1}^{\ell}(d_k - o_k)^2$$

- **误差展开至隐层**

$$E = \frac{1}{2}\sum_{\kappa=1}^{\ell}[d_\kappa - f(net_\kappa)]^2 \qquad = \frac{1}{2}\sum_{\kappa=1}^{\ell}[d_\kappa - f(\sum_{j=0}^{m}\omega_{j\kappa}y_j)]^2$$

- **误差展开至输入层**

$$E = \frac{1}{2}\sum_{\kappa=1}^{\ell} d_\kappa - f[\sum_{j=0}^{m}\omega_{j\kappa}f(net_j)]^{2} \qquad = \frac{1}{2}\sum_{\kappa=1}^{\ell} d_\kappa - f[\sum_{j=0}^{m}\omega_{j\kappa}f(\sum_{j=0}^{n}v_{ij}\chi_i)]^{2}$$

继续求解

- 误差E有了，怎么调整权重让误差不断减小

- (随机)梯度下降，梯度怎么算？

$$\Delta\omega_{j\kappa} = -\eta\frac{\partial E}{\partial\omega_{j\kappa}} j = 0, 1, 2, \ldots, m; \qquad \kappa = 1, 2, \ldots, \ell$$

$$\Delta v_{ij} = -\eta\frac{\partial E}{\partial v_{ij}} i = 0, 1, 2, \ldots, n; \qquad j = 1, 2, \ldots, m$$

- 更新权重

$$w_{jk} = w_{jk} + \Delta w_{jk}$$

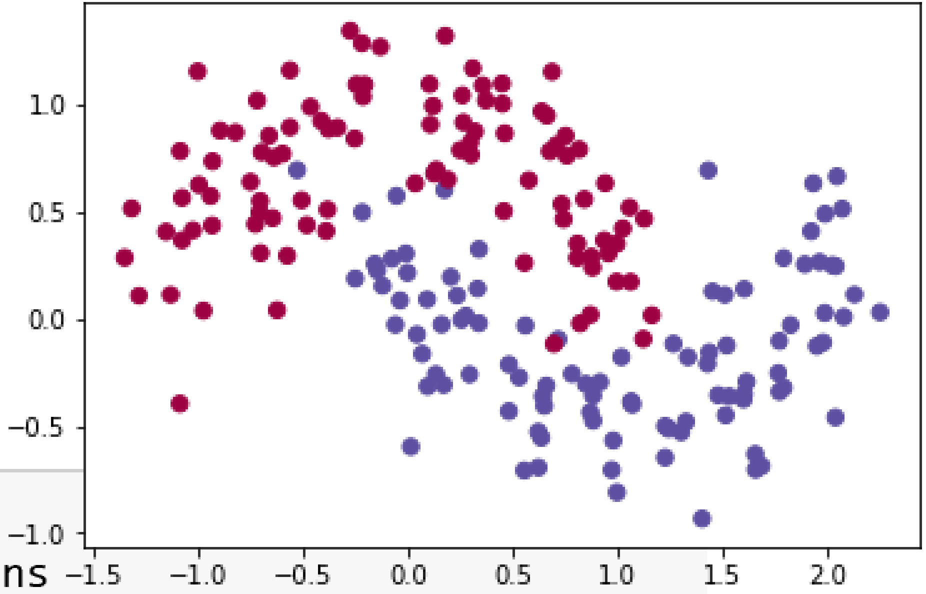
$$v_{ij} = v_{ij} + \Delta v_{ij}$$

04

PART FOUR

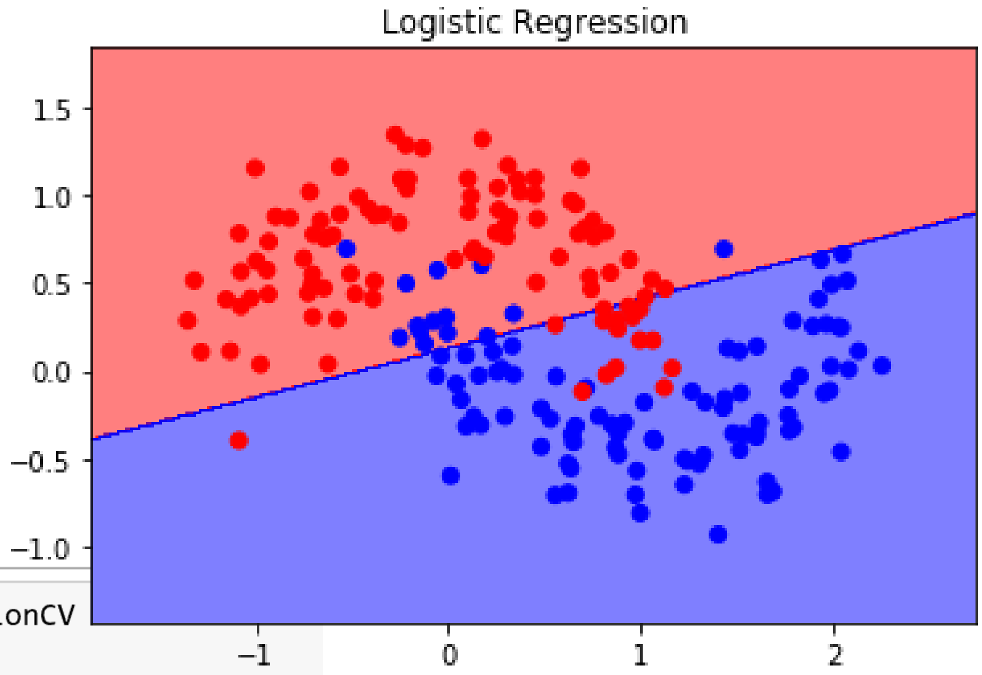
代码实例

生成数据集




```
import numpy as np
from sklearn.datasets import make_moons
import matplotlib.pyplot as plt
# 手动生成一个随机的平面点分布，并画出来
np.random.seed(0)
X, y = make_moons(200, noise=0.20)
plt.scatter(X[:,0], X[:,1], s=40, c=y, cmap=plt.cm.Spectral)
plt.show()
```

逻辑回归

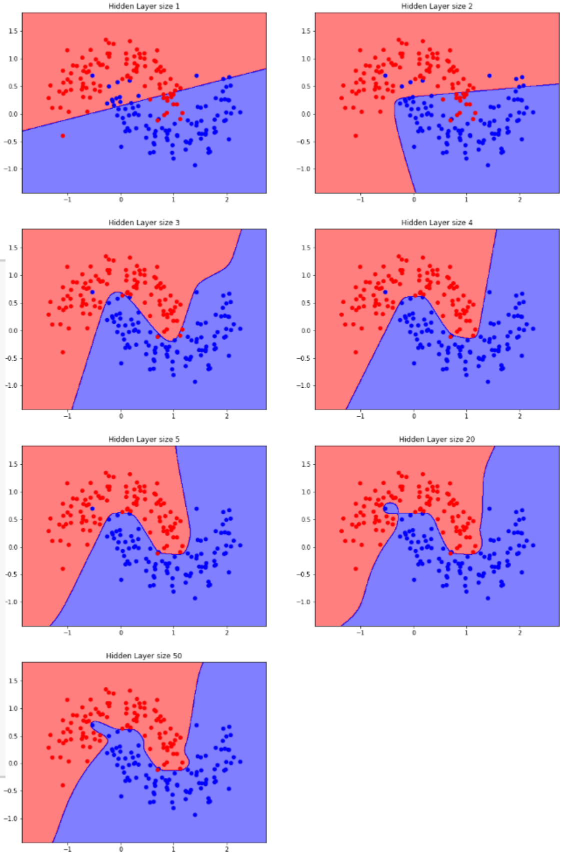



```python
from sklearn.linear_model import LogisticRegressionCV
#咱们先来瞄一眼逻辑斯特回归对于它的分类效果
clf = LogisticRegressionCV()
clf.fit(X, y)

# 画一下决策边界
plot_decision_boundary(lambda x: clf.predict(x))
plt.title("Logistic Regression")
plt.show()
```

ANN：一个隐层，不同节点数

```
# 不同的隐层个数对结果的影响
# 那咱们来一起看看吧
plt.figure(figsize=(16, 32))
hidden_layer_dimensions = [1, 2, 3, 4, 5, 20, 50]
for i, nn_hdim in enumerate(hidden_layer_dimensions):
    plt.subplot(5, 2, i+1)
    plt.title('Hidden Layer size %d' % nn_hdim)
    ann=MLPClassifier(alpha=0.01,epsilon = 0.01,solver='lbfgs',
                      hidden_layer_sizes=(nn_hdim,),activation='tanh')
    ann.fit(X, y)
    # 画一下决策边界
    plot_decision_boundary(lambda x: ann.predict(x))
plt.show()
```



案列分析

MNIST图片数据手写识别

- 图片数据：
  - 包含手写0到9的图片
  - 可以将图片信息转化为向量信息
  - 进行分类