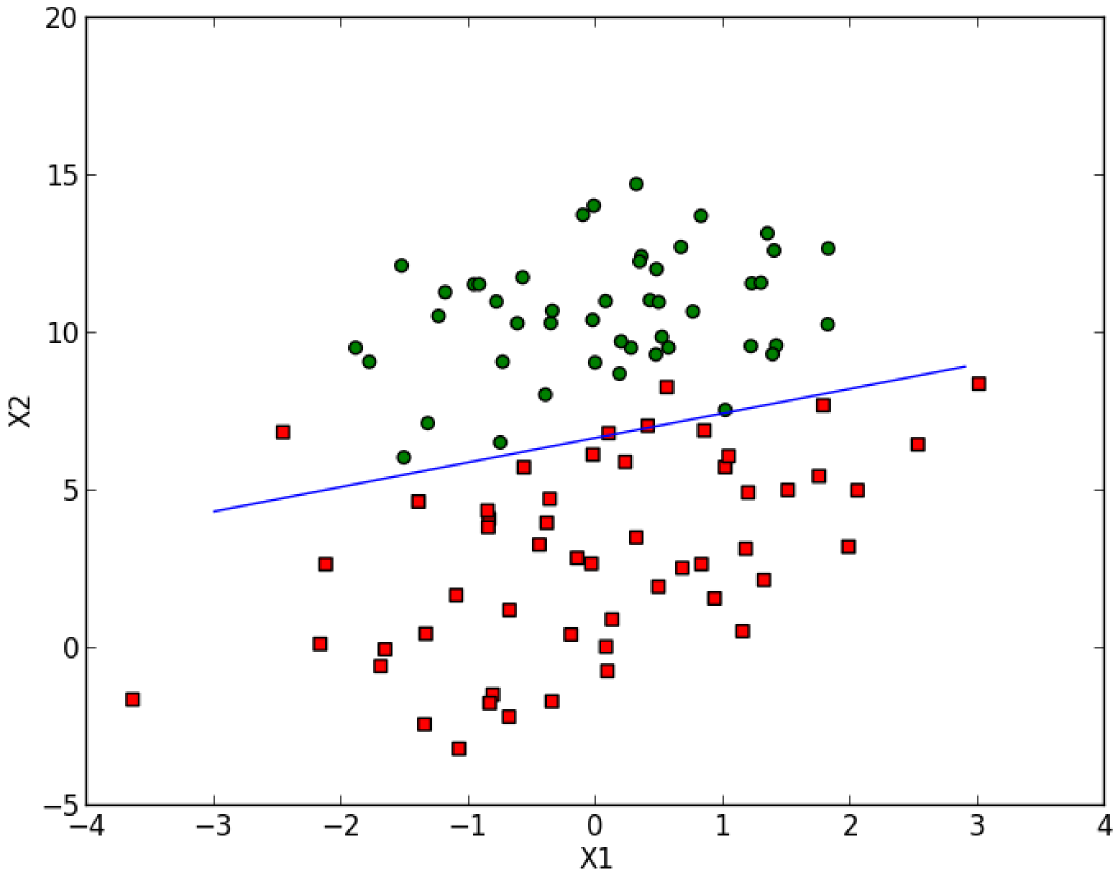
逻辑回归 Logistic Regression

自兴人工智能

陈白帆

什么是逻辑回归

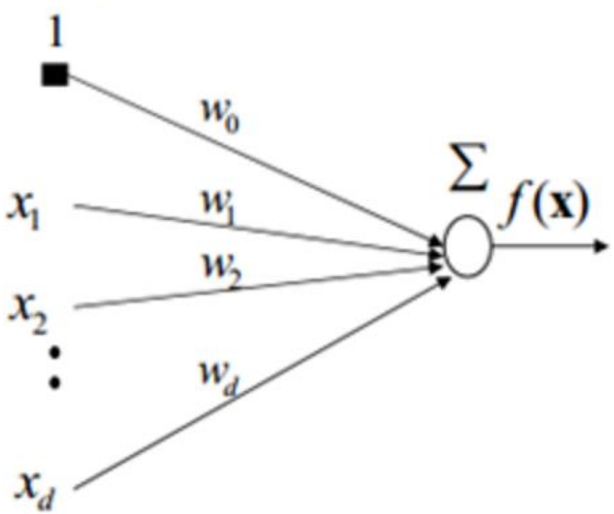
- “回归”仍是指拟合
- 分类方法，用于两分类问题
- 根据现有数据对分类边界线建立回归，以此分类。
- 采用最优化方法，找到分类边界的最佳拟合参数集。



20
15
10
5
0
−5
−4
−3
−2
−1
0
1
2
3
4
X1
X2

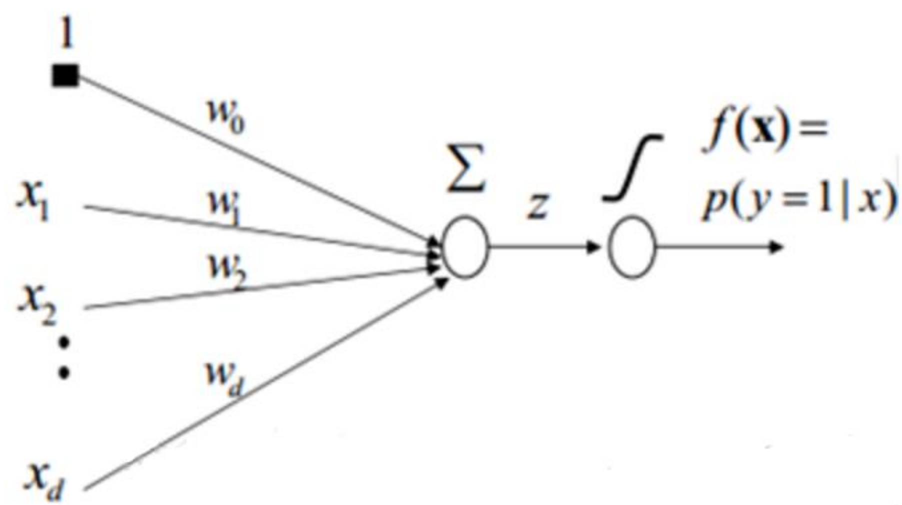
线性回归与逻辑回归

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$$



线性回归

$$f(\mathbf{x}) = p(y=1\,|\,\mathbf{x},\mathbf{w}) = g(\mathbf{w}^T\mathbf{x})$$



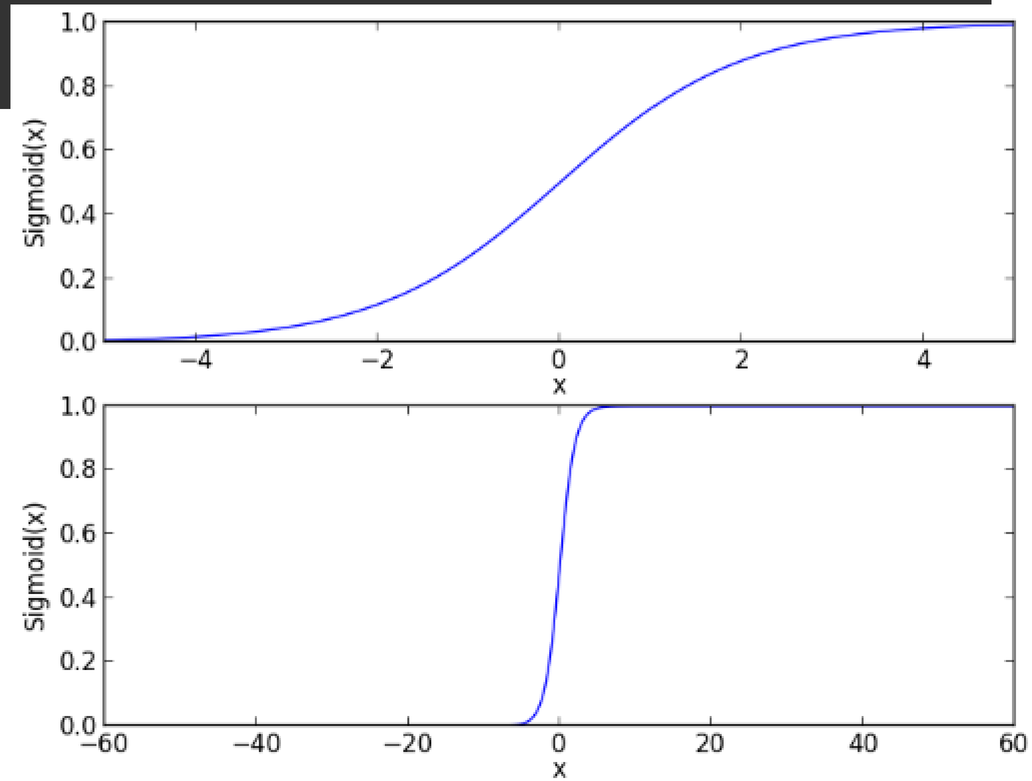
逻辑回归

分类函数

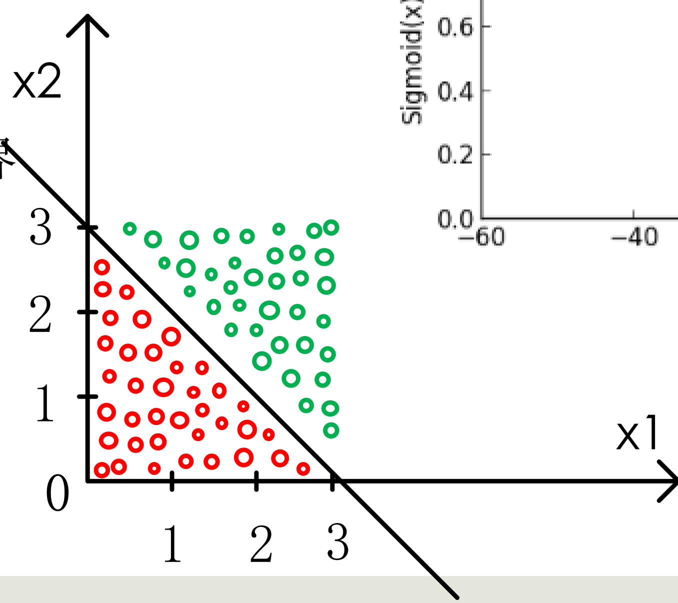
- 分类函数能接受所有的输入，然后预测出类别，如：0或1。
- 单位阶跃函数
- Sigmoid函数

Sigmoid函数

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



输入z，为分类边界



逻辑回归方程

- 边界

$$z = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n = \mathbf{w}^{\mathrm{T}}\mathbf{x} \tag{1}$$

  - w_i为回归系数
- 逻辑回归方程

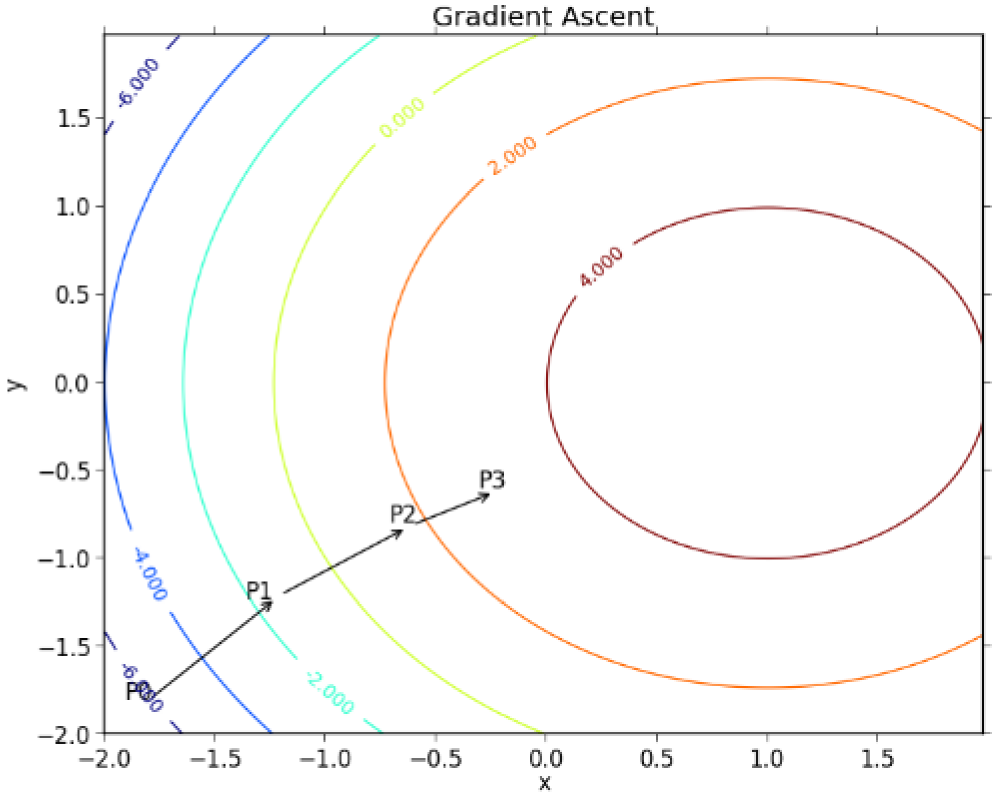
$$h(x) = g(\mathbf{w}^{\mathrm{T}}\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^{\mathrm{T}}\mathbf{x}}} \tag{2}$$

- ***h(x)*函数值的含义：分类为1的概率。**
  - 对于输入*x*分类结果为类别1和类别0的概率分别为：

$$\begin{aligned} p(y=1) &= h(x) \\ p(y=0) &= 1 - h(x) \end{aligned} \tag{3}$$

最佳回归系数

- 最佳回归系数w，可使分类器更加精确。
- 最优化方法
  - 梯度上升法
  - 随机梯度上升法



梯度上升法

- 沿着函数的梯度方向搜索最大值。

$$w := w + \underbrace{a}_{\text{学习步长}} \nabla_w f(w) \tag{4}$$

- 梯度算子

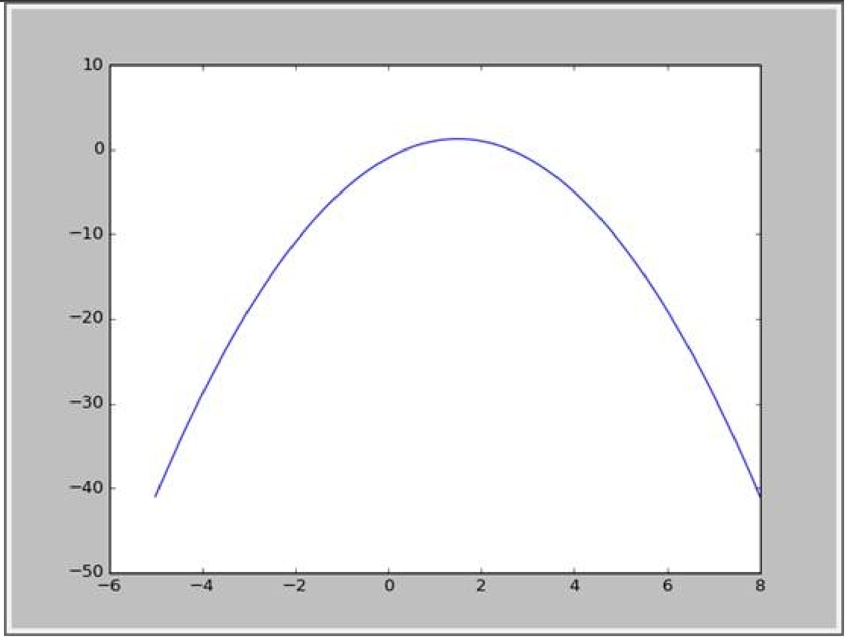
$$\nabla f(x,y) = \begin{pmatrix} \underbrace{\dfrac{\partial f(x,y)}{\partial x}}_{\text{沿}x\text{方向移动}} \\ \underbrace{\dfrac{\partial f(x,y)}{\partial y}}_{\text{沿}y\text{方向移动}} \end{pmatrix}$$

- 函数$f(x, y)$须在(x, y)处有定义且可微。

关于梯度

$$f(x) = -x^2 + 3x - 1$$

$$f'(x) = -2x + 3$$



x = 1.5，取得函数的最大值1.25

关于梯度（推导）

样本x得到输出y的生成概率

$$p(y \mid x;\theta) = (h_\theta(x))^y \left(1-h_\theta(x)\right)^{1-y}$$

假定样本与样本之间相互独立，那么整个样本集生成的概率即似然函数为：

$$\begin{aligned} L(\theta) &= p(\vec{y} \mid X;\theta) \\ &= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)};\theta) \\ &= \prod_{i=1}^{m} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \left(1-h_\theta(x^{(i)})\right)^{1-y^{(i)}} \end{aligned}$$

其中，m为样本的总数，$y^{(i)}$表示第i个样本的类别，$x^{(i)}$表示第i个样本

关于梯度（推导）

对似然函数求对数

$$\begin{aligned}\ell(\theta) &= \log L(\theta)\\ &= \sum_{i=1}^{m} y^{(i)}\log h(x^{(i)}) + (1-y^{(i)})\log(1-h(x^{(i)}))\end{aligned}$$

对似然函数对数求导

$$\begin{aligned}\frac{\partial}{\partial\theta_j}\ell(\theta) &= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)\frac{\partial}{\partial\theta_j}g(\theta^T x)\\ &= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)g(\theta^T x)(1-g(\theta^T x)\frac{\partial}{\partial\theta_j}\theta^T x\\ &= \left(y(1-g(\theta^T x)) - (1-y)g(\theta^T x)\right)x_j\\ &= \left(y-h_\theta(x)\right)x_j\end{aligned}$$

关于梯度（推导）

梯度为：

$$\nabla_{w_j} f(w_j) = (y - h(x))x_j$$

回归系数更新公式为：

$$w_j(t+1) := w_j(t) + a(y(t) - h(x(t))x_j(t)$$

梯度上升算法

- 初始化回归系数为1
- 重复下面步骤直到收敛
  - 计算整个数据集的梯度
  - 使用alpha * gradient更新回归系数
- 返回回归系数值

```
def gradAscent(dataMatIn, classLabels):
    dataMatrix = mat(dataMatIn)             #convert to NumPy matrix
    labelMat = mat(classLabels).transpose() #convert to NumPy matrix
    m,n = shape(dataMatrix)
    alpha = 0.001
    maxCycles = 500
    weights = ones((n,1))
    for k in range(maxCycles):              #heavy on matrix operations
        h = sigmoid(dataMatrix*weights)     #matrix mult
        error = (labelMat - h)              #vector subtraction
        weights = weights + alpha * dataMatrix.transpose()* error
    return weights
```

随机梯度上升算法

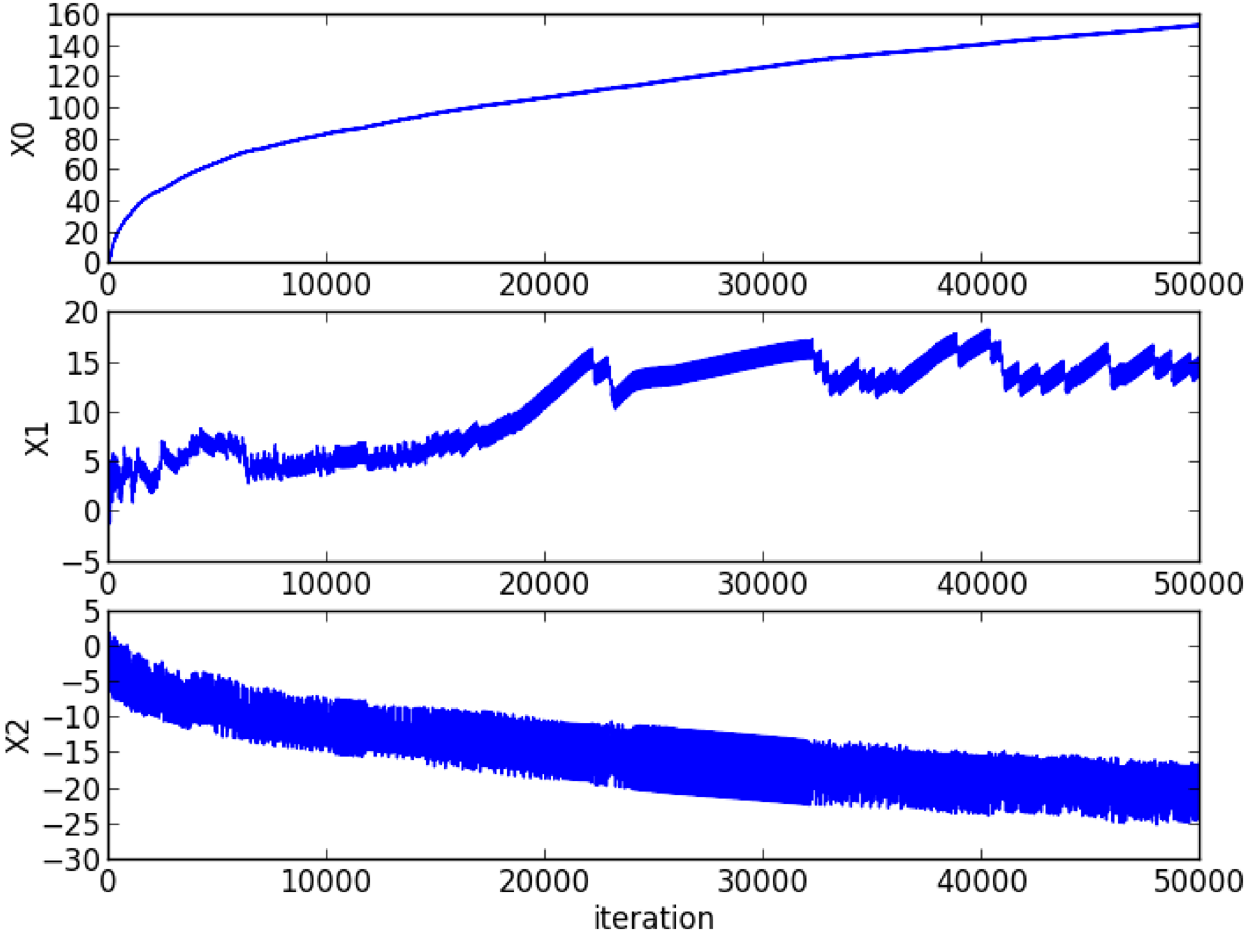
- 思想：一次仅用一个样本点（的回归误差）来更新回归系数
- 在线学习算法

随机梯度上升算法

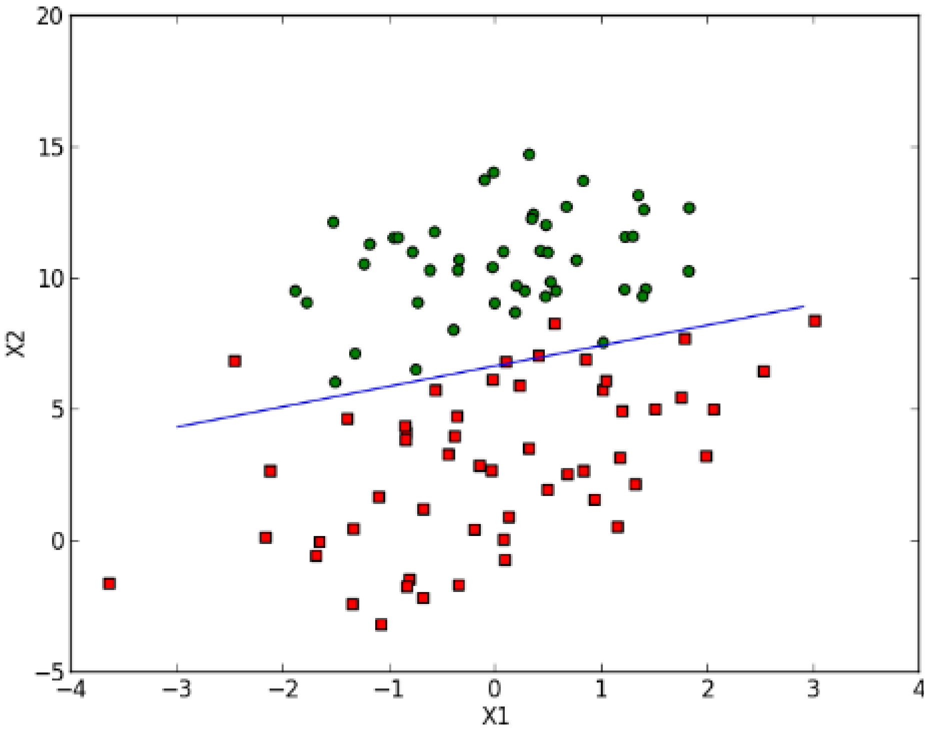
- 初始化回归系数为1
- 对数据集中每个样本
  - 计算该样本的梯度
  - 使用alpha * gradient更新回归系数
- 返回回归系数值

ZIXING AI 自兴人工智能

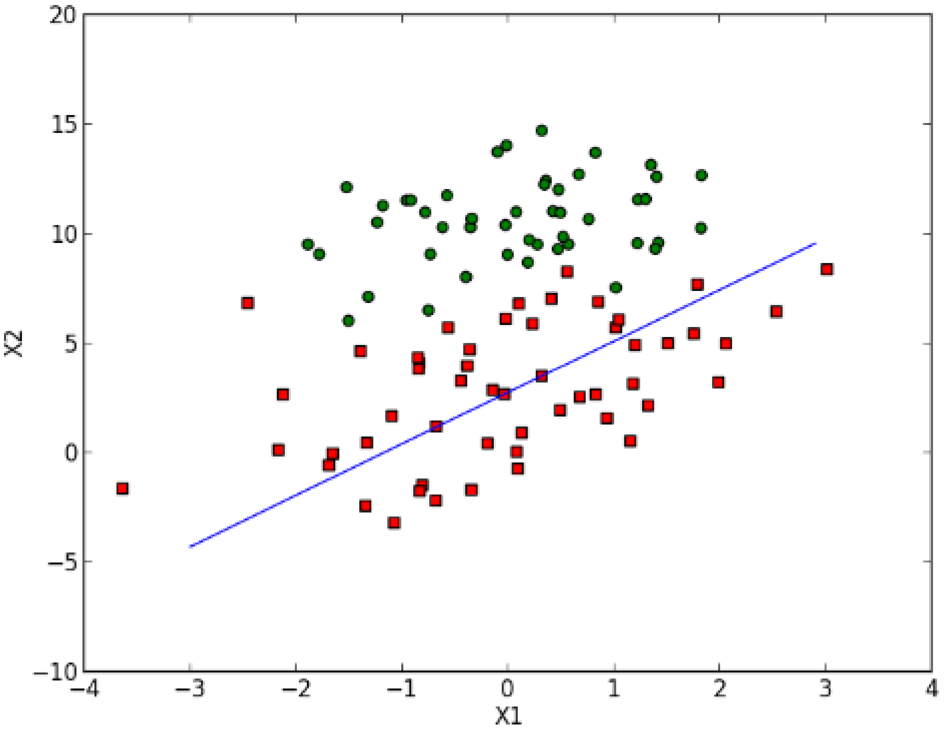
回归系数变化



比较

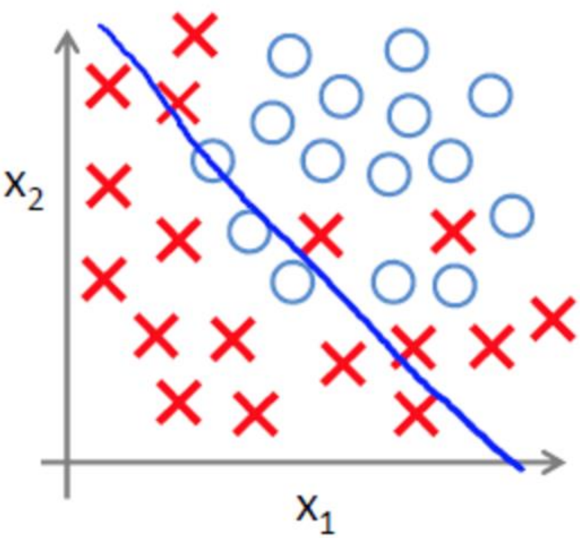


梯度上升

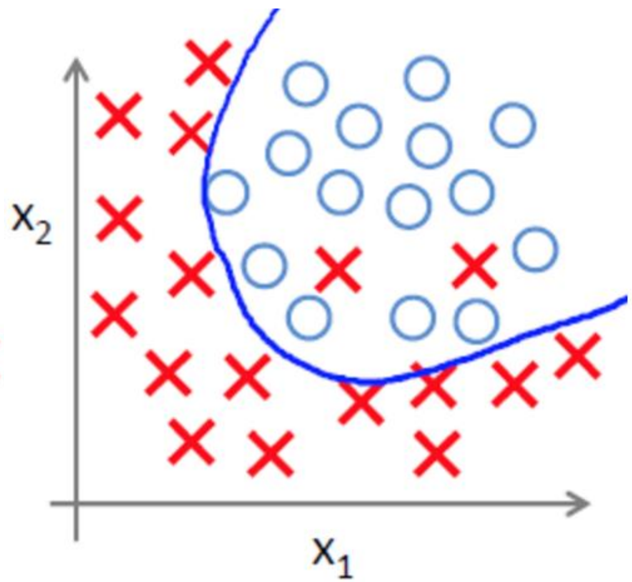


随机梯度上升

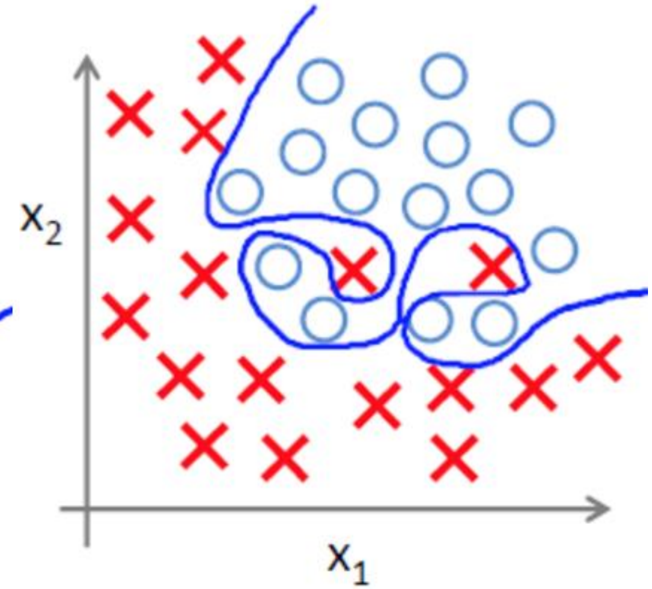
三种拟合



欠拟合



合适拟合



过拟合

总结

- 优点
  - 计算代价不高、易于理解和实现
- 缺点
  - 易欠拟合，分类精度不高
- 适用数据
  - 数值型、标签型

练习：病马死亡率预测

- 数据集：预处理后的Horse+Colic
- 示例数：368
- 特征（21）
- 种类（2）
- 要求
  - 画出数据集带分类的图示和分类线
  - 获得最佳回归系数
  - 计算分类错误率
  - 调整迭代次数、学习步长比较结果，并分析收敛情况