线性回归

- 一个栗子
  - 数据：工资和年龄（2个特征）
  - 目标：预测银行会贷款给我多少钱（标签）
  - 考虑：工资和年龄都会影响最终银行贷款的结果那么它们各自有多大的影响呢？（参数）

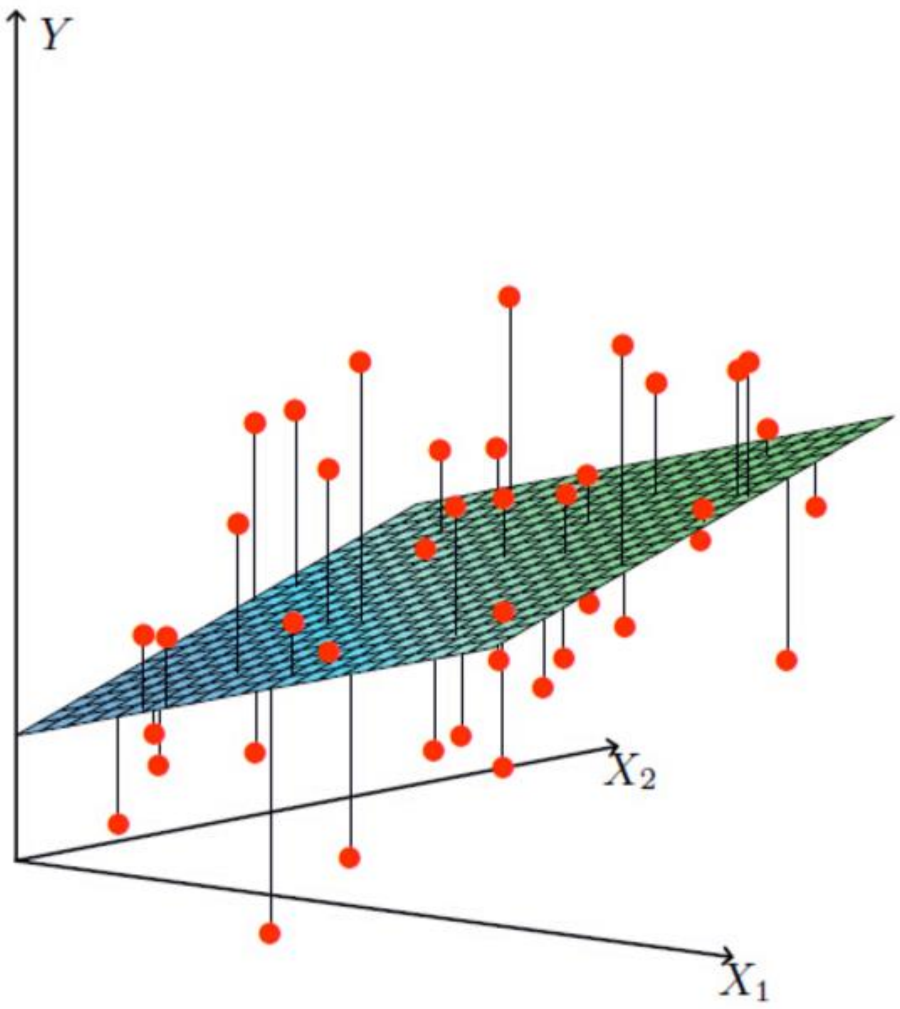
工资	年龄	额度
4000	25	20000
8000	30	70000
5000	28	35000
7500	33	50000
12000	40	85000

线性回归

- 通俗解释

  - X1,X2就是我们的两个特征（年龄，工资）
    Y是银行最终会借给我们多少钱

  - 找到最合适的一条线（想象一个高维）来
    最好的拟合我们的数据点



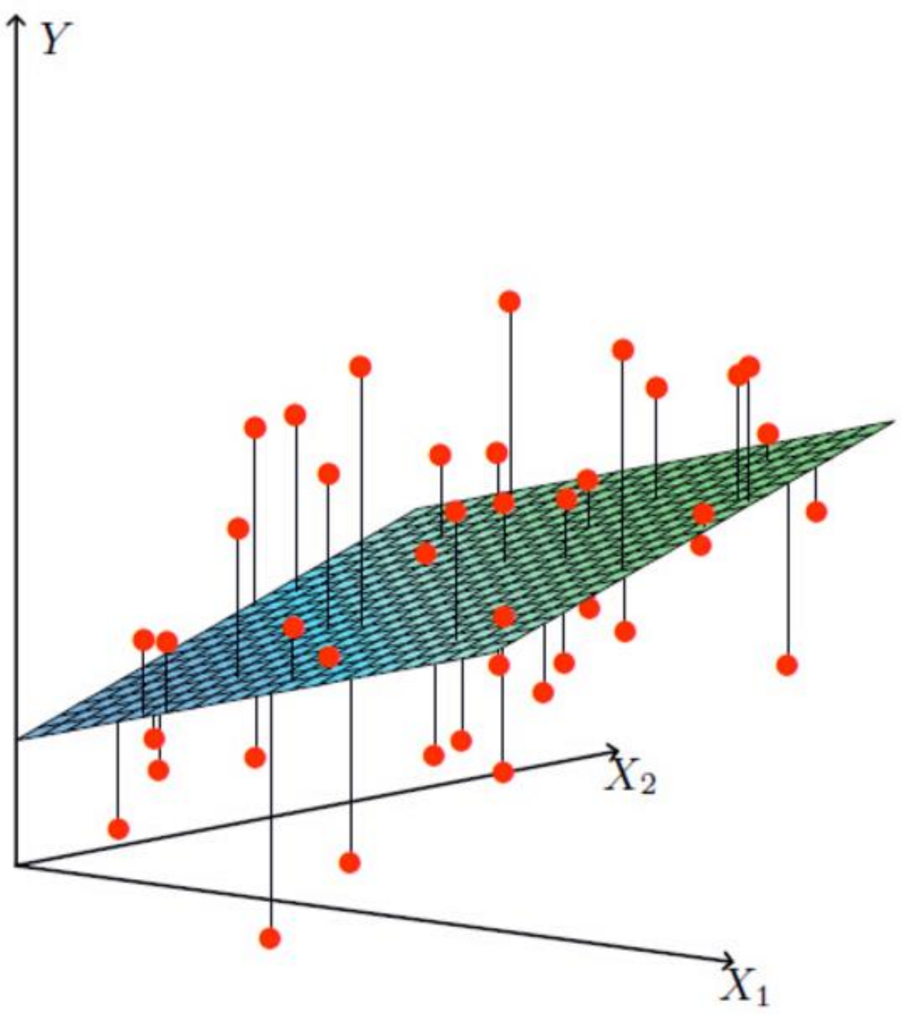
线性回归

- 数学来了

- 假设 θ_1 是年龄的参数，θ_2 是工资的参数

- 拟合的平面：$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$
（θ_0 是偏置项）

- 整合：$h_\theta(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$

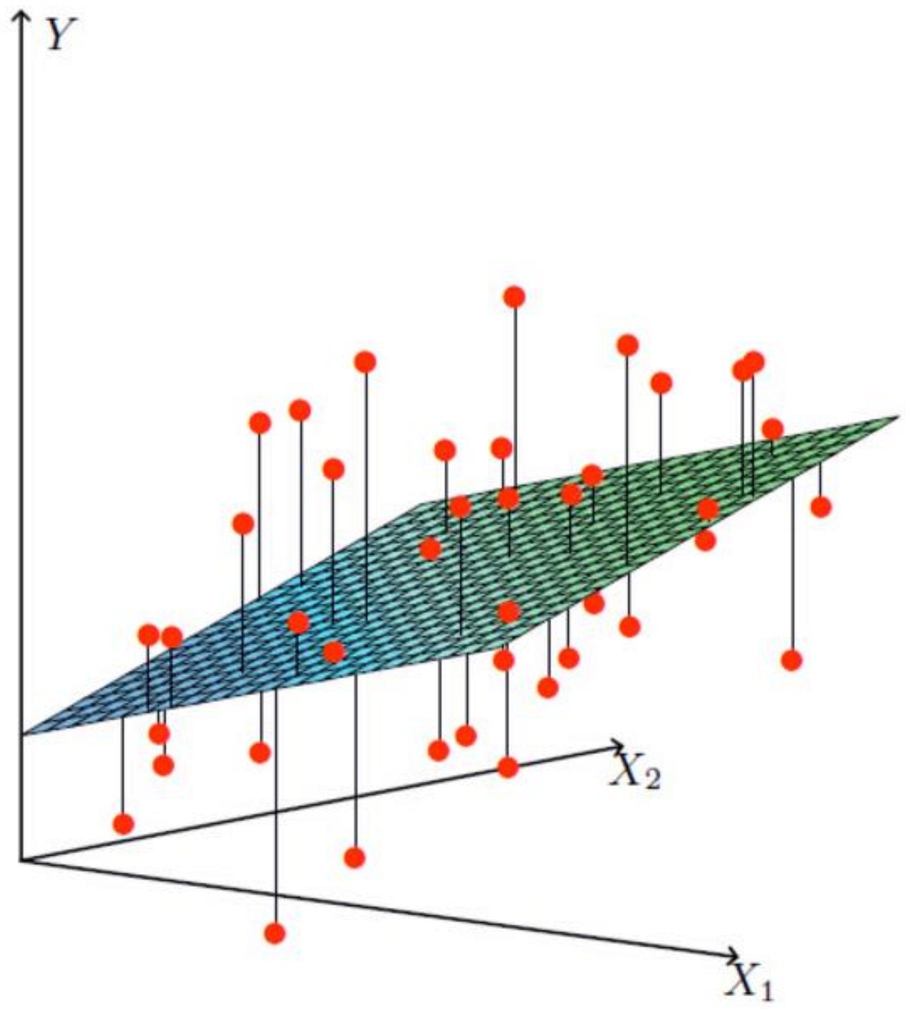


线性回归

误差

- 真实值和预测值之间肯定是要存在差异的（用 ε 来表示该误差）

- 对于每个样本：$y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)}$



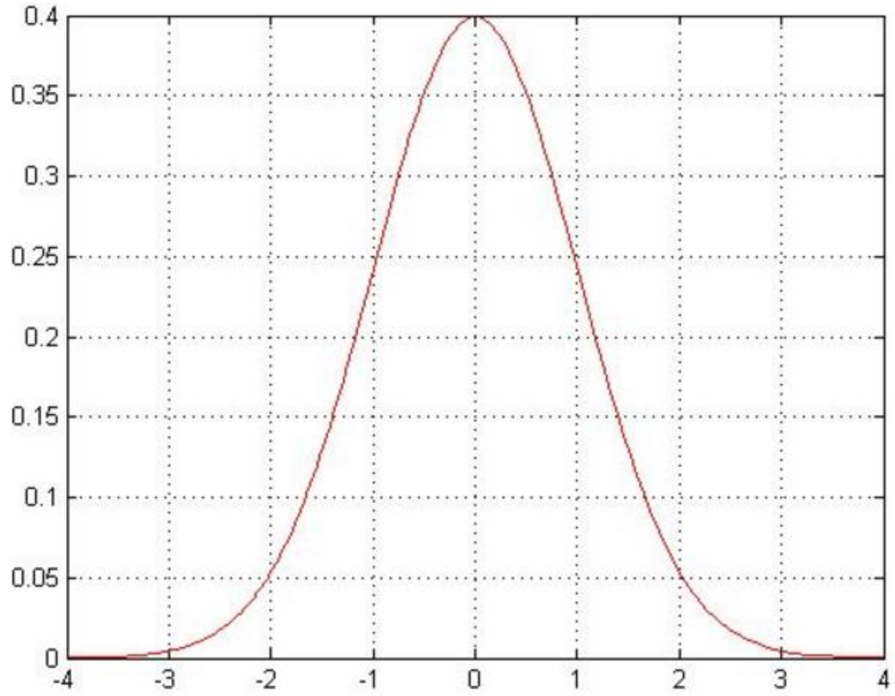
线性回归

误差

- 误差 $\varepsilon^{(i)}$是独立并且具有相同的分布，并且服从均值为0方差为 θ^2 的高斯分布

- 独立：张三和李四一起来贷款，他俩没关系

- 同分布：他俩都来得是我们假定的这家银行



- 高斯分布：银行可能会多给，也可能会少给，但是绝大多数情况下这个浮动不会太大，极小情况下浮动会比较大，符合正常情况

线性回归

误差

预测值与误差：$y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)}$ （1）

由于误差服从高斯分布：$p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\epsilon^{(i)})^2}{2\sigma^2}\right)$ （2）

将（1）式带入（2）式：$p(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$

线性回归

误差

似然函数： $L(\theta) = \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)}; \theta) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$

解释：什么样的参数跟我们的数据组合后恰好是真实值

对数似然： $\log L(\theta) = \log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$

解释：乘法难解，加法就容易了，对数里面乘法可以转换成加法

线性回归

误差

展开化简：$\sum_{i=1}^{m} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$

$= m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2$

目标：让似然函数（对数变换后也一样）越大越好

$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2$ （最小二乘法）

线性回归

误差

- 目标函数：$J(\theta)=\frac{1}{2}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right)^2=\frac{1}{2}(X\theta-y)^T(X\theta-y)$

- 求偏导：

$$\nabla_\theta J(\theta)=\nabla_\theta\left(\frac{1}{2}(X\theta-y)^T(X\theta-y)\right)=\nabla_\theta\left(\frac{1}{2}\left(\theta^TX^T-y^T\right)(X\theta-y)\right)$$

$$=\nabla_\theta\left(\frac{1}{2}\left(\theta^TX^TX\theta-\theta^TX^Ty-y^TX\theta+y^Ty\right)\right)$$

$$=\frac{1}{2}\left(2X^TX\theta-X^Ty-\left(y^TX\right)^T\right)=X^TX\theta-X^Ty$$

- 偏导等于0：$\theta=\left(X^TX\right)^{-1}X^Ty$

线性回归

评估方法

- 最常用的评估项 R^2：$1-\dfrac{\sum_{i=1}^{m}(\hat{y}_i-y_i)^2}{\sum_{i=1}^{m}(y_i-\bar{y})^2}$

  （残差平方和）

  （类似方差项）

- R^2 的取值越接近于1我们认为模型拟合的越好

线性回归

梯度下降

- 引入：当我们得到了一个目标函数后，如何进行求解？
  直接求解？（并不一定可解，线性回归可以当做是一个特例）

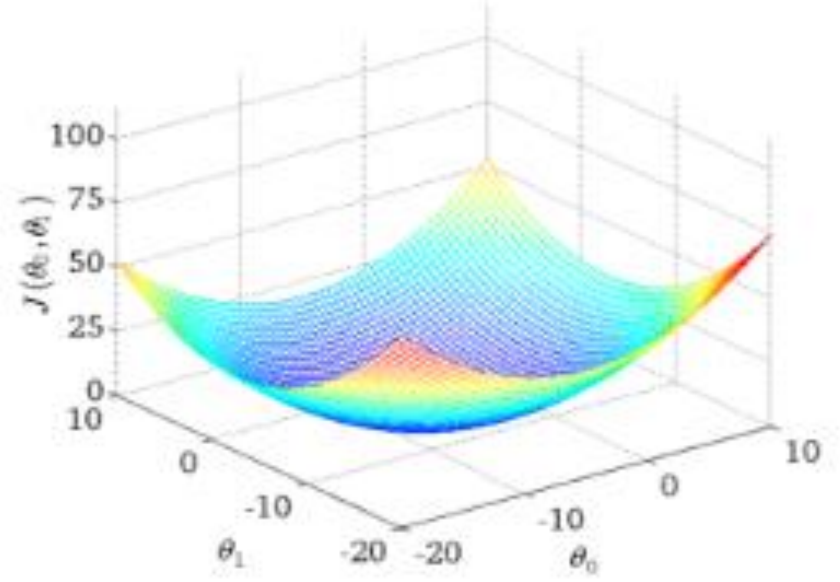
- 常规套路：机器学习的套路就是我交给机器一堆数据，然后告诉它
  什么样的学习方式是对的（目标函数），然后让它朝着这个方向去做

- 如何优化：一口吃不成个胖子，我们要静悄悄的一步步的完成迭代
  （每次优化一点点，累积起来就是个大成绩了）

线性回归

梯度下降

目标函数： $J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)$



寻找山谷的最低点，也就是我们的目标函数终点（什么样的参数能使得目标函数达到极值点）

下山分几步走呢？（更新参数）
（1）：找到当前最合适的方向
（2）：走那么一小步，走快了该”跌倒 ”了
（3）：按照方向与步伐去更新我们的参数

线性回归

theta就是方程中每一个变量的系数组合成一个矩阵

梯度下降，目标函数： $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(y^i - h_\theta(x^i))^2$

批量梯度下降： $\frac{\partial J(\theta)}{\partial \theta_j} = -\frac{1}{m}\sum_{i=1}^{m}(y^i - h_\theta(x^i))x_j^i \quad \theta_j' = \theta_j + \frac{1}{m}\sum_{i=1}^{m}(y^i - h_\theta(x^i))x_j^i$

（容易得到最优解，但是由于每次考虑所有样本，速度很慢）

随机梯度下降： $\theta_j' = \theta_j + (y^i - h_\theta(x^i))x_j^i$

选择的样本数量

（每次找一个样本，迭代速度快，但不一定每次都朝着收敛的方向）

小批量梯度下降法： $\underbrace{\theta_j}_{\text{theta}} := \theta_j - \underbrace{\alpha}_{\text{alpha}} \boxed{\frac{1}{10}}\sum_{k=i}^{i+9}\boxed{(h_\theta(x^{(k)}) - y^{(k)})}x_j^{(k)}$

delta = predict − label

（每次更新选择一小部分数据来算，实用！）

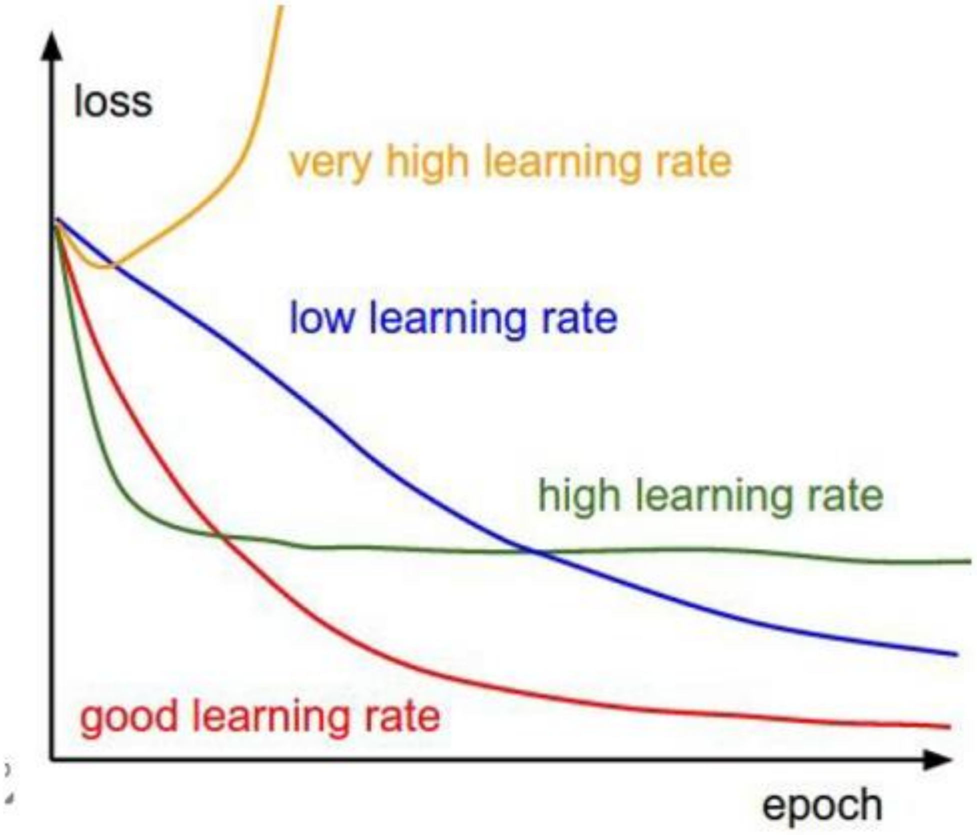
线性回归

- 梯度下降

- 学习率（步长）：对结果会产生巨大的影响，一般小一些

- 如何选择：从小的时候，不行再小

- 批处理数量：32，64，128都可以，很多时候还得考虑内存和效率



逻辑回归

- Logistic regression
  - 目的：分类还是回归？经典的二分类算法！
  - 机器学习算法选择：先逻辑回归再用复杂的，能简单还是用简单的
  - 逻辑回归的决策边界：可以是非线性的

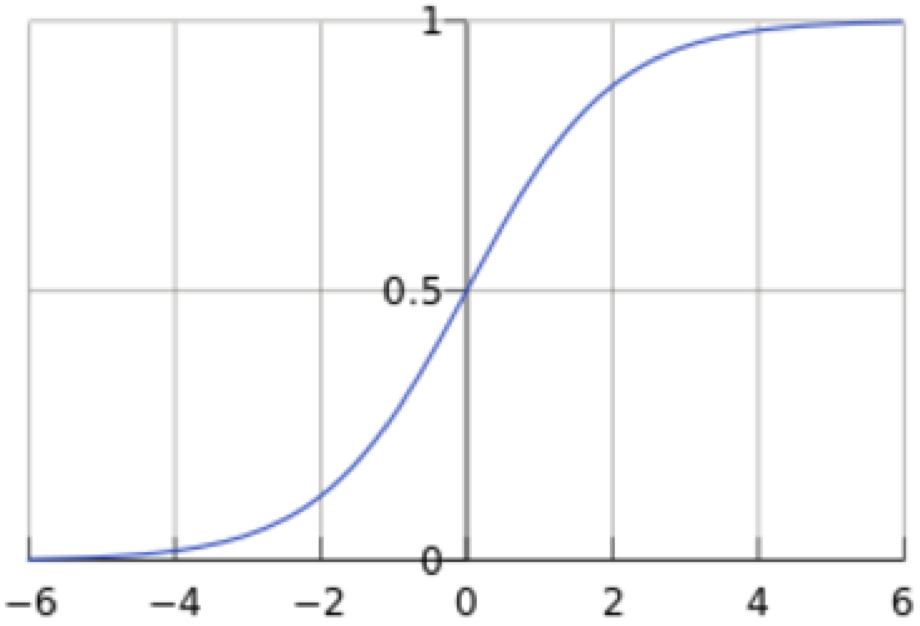
逻辑回归

- Sigmoid 函数

  - 公式： $g(z)=\dfrac{1}{1+e^{-z}}$

  - 自变量取值为任意实数，值域[0,1]

  - 解释：将任意的输入映射到了[0,1]区间
    我们在线性回归中可以得到一个预测值，再将该值映射到Sigmoid 函数
    中这样就完成了由值到概率的转换，也就是分类任务

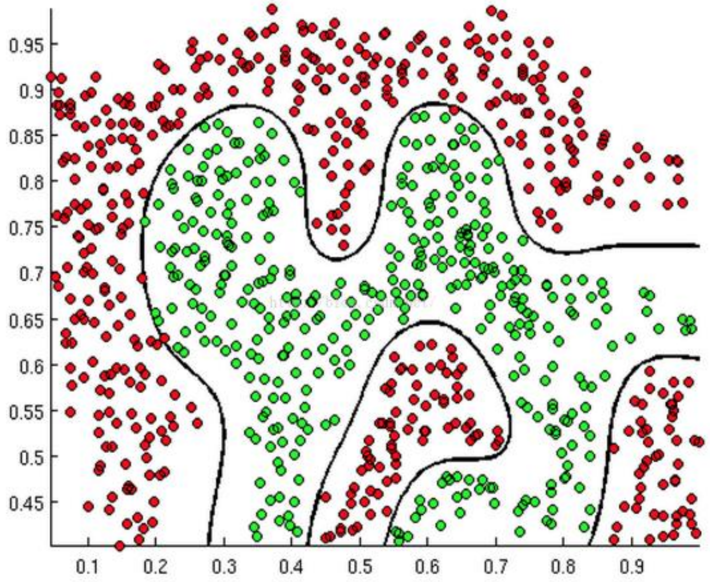


逻辑回归

- Sigmoid 函数

预测函数： $h_\theta(x) = g(\theta^T x) = \dfrac{1}{1+e^{-\theta^T x}}$

其中 $\theta_0 + \theta_1 x_1 + ,\ldots, + \theta_n x_n = \sum_{i=1}^{n} \theta_i x_i = \theta^T x$



分类任务： $P(y=1|x;\theta) = h_\theta(x)$
$P(y=0|x;\theta) = 1 - h_\theta(x)$

整合： $P(y|x;\theta) = (h_\theta(x))^y (1-h_\theta(x))^{1-y}$

解释：对于二分类任务（0，1），整合后y取0只保留 $(1-h_\theta(x))^{1-y}$
y取1只保留 $(h_\theta(x))^y$

逻辑回归

- Logistic regression

- 似然函数：$L(\theta)=\prod_{i=1}^{m}P(y_i\mid x_i;\theta)=\prod_{i=1}^{m}(h_\theta(x_i))^{y_i}(1-h_\theta(x_i))^{1-y_i}$

- 对数似然：$l(\theta)=\log L(\theta)=\sum_{i=1}^{m}(y_i\log h_\theta(x_i)+(1-y_i)\log(1-h_\theta(x_i)))$

- 此时应用梯度上升求最大值，引入 $J(\theta)=-\frac{1}{m}l(\theta)$ 转换为梯度下降任务

逻辑回归

- 求导过程：$l(\theta)=\log L(\theta)=\sum_{i=1}^{m}\left(y_i\log h_\theta(x_i)+(1-y_i)\log(1-h_\theta(x_i))\right)$

$$\frac{\delta}{\delta\theta_j}J(\theta)=-\frac{1}{m}\sum_{i=1}^{m}\left(y_i\frac{1}{h_\theta(x_i)}\frac{\delta}{\delta\theta_j}h_\theta(x_i)-(1-\mathrm{y}_i)\frac{1}{1-h_\theta(x_i)}\frac{\delta}{\delta\theta_j}h_\theta(x_i)\right)$$

$$=-\frac{1}{m}\sum_{i=1}^{m}\left(y_i\frac{1}{g(\theta^{\mathrm{T}}x_i)}-(1-\mathrm{y}_i)\frac{1}{1-g(\theta^{\mathrm{T}}x_i)}\right)\frac{\delta}{\delta\theta_j}g(\theta^{\mathrm{T}}x_i)$$

$$=-\frac{1}{m}\sum_{i=1}^{m}\left(y_i\frac{1}{g(\theta^{\mathrm{T}}x_i)}-(1-\mathrm{y}_i)\frac{1}{1-g(\theta^{\mathrm{T}}x_i)}\right)g(\theta^{\mathrm{T}}x_i)(1-g(\theta^{\mathrm{T}}x_i))\frac{\delta}{\delta\theta_j}\theta^{\mathrm{T}}x_i$$

$$=-\frac{1}{m}\sum_{i=1}^{m}\left(y_i(1-g(\theta^{\mathrm{T}}x_i))-(1-\mathrm{y}_i)g(\theta^{\mathrm{T}}x_i)\right)x_i^j$$

$$=-\frac{1}{m}\sum_{i=1}^{m}\left(y_i-g(\theta^{\mathrm{T}}x_i)\right)x_i^j$$

$$=\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x_i)-y_i\right)x_i^j$$

逻辑回归

- Logistic regression

参数更新：$\theta_j := \theta_j - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)x_i^j$

多分类的softmax: $h_\theta(x^{(i)}) = \begin{bmatrix} p(y^{(i)} = 1|x^{(i)};\theta) \\ p(y^{(i)} = 2|x^{(i)};\theta) \\ \vdots \\ p(y^{(i)} = k|x^{(i)};\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_2^T x^{(i)}} \\ \vdots \\ e^{\theta_k^T x^{(i)}} \end{bmatrix}$

总结：逻辑回归真的真的很好很好用！