线性回归

2022年3月29日, 星期二　　11:28

1、监督学习包括:

- 分类
- 线性回归（找一个具体的值）

2、例子:

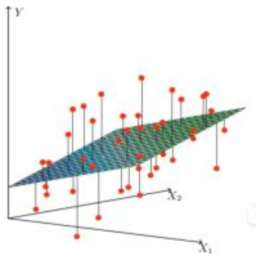
数据：工资(X1)、年龄（X2)

目标：预测银行会贷多少款

考虑：工资和年龄都会有影响，

　　　那么他们各自的影响多大呢（参数）

工资	年龄	额度
4000	25	2000
8000	30	70000
5000	28	35000
7500	33	50000
12000	40	85000



三、误差

- 误差的概率论意义:

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

$$h_{(\theta)}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$
$$= [1, x_1, x_2]\begin{bmatrix}\theta_0 \\ \theta_1 \\ \theta_2\end{bmatrix} = \sum\nolimits_0^2 \theta_i x_i = \theta^T x$$

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

误差服从高斯分布：$$P\left(\varepsilon^{(i)}\right) = \frac{1}{\sqrt{2\pi}\sigma} e^{\left(-\frac{\left(\varepsilon^{(i)}-0\right)^2}{2\sigma^2}\right)}$$ 2

将1式代入2式（消去$\varepsilon^{(i)}$），得:

$$p\left(y^{(i)} | x^{(i)}; \theta\right) = \frac{1}{\sqrt{2\pi}\sigma} e^{\left[-\frac{\left(y^{(i)}-\theta^T x^{(i)}-0\right)^2}{2\sigma^2}\right]}$$ 3

似然函数：$$L(\theta) = \prod_{i=1}^{m} p(y^{(i)} | x^{(i)}; \theta) = \prod_{i=1}^{m} \frac{1}{\theta\sqrt{2\pi}} e^{-\frac{(y^{(i)}-\theta^T x^{(i)})^2}{2\theta^2}}$$ 4

解释：什么样的参数跟我们的数据组合后恰好是真实值

对数函数：$$logL(\theta) = log\prod_{i=1}^{m} \frac{1}{\theta\sqrt{2\pi}} e^{-\frac{(y^{(i)}-\theta^T x^{(i)})^2}{2\theta^2}}$$ 5

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

- **展开化简5式**：$$log\prod_{i=1}^{m}\frac{1}{\theta\sqrt{2\pi}}e^{-\frac{(y^{(i)}-\theta^T x^{(i)})^2}{2\theta^2}}$$
$$=\dot{m}\,log\frac{1}{\sqrt{2\pi}\sigma}-\frac{1}{\sigma^2}\cdot\frac{1}{2}\sum_{i=1}^{m}\left(y^{(i)}-\theta^T\chi^{(i)}\right)^2 \qquad 6$$

- **目标**：让似然函数越大越好，也就是让蓝色部分越小越好（概率论原理：已经出现的概率越大越好）
  - 目标函数（损失函数、lossfunction）：

$$J(\theta)=\frac{1}{2}\sum_{i=1}^{m}\left(y^{(i)}-\theta^T\chi^{(i)}\right)^2 \qquad \text{（最小二乘法）} \qquad 7$$

预测值 真实值

$$7式=\frac{1}{2}\sum_{i=1}^{m}\left(y^{(i)}-\theta^T\chi^{(i)}\right)^2=\frac{1}{2}\sum_{i=1}^{m}\left(h_\theta(x^{(i)})-y^{(i)}\right)^2=\frac{1}{2}(X\theta-y)^T(X\theta-y)$$

化成矩阵的形式，等于矩阵的转置乘以自身

- 对目标函数求偏导：
$$\nabla_\theta J(\theta)=\nabla_\theta\left(\frac{1}{2}(X\theta-y)^T(X\theta-y)\right)=\nabla_\theta\left(\frac{1}{2}(\theta^TX^T-y^T)(X\theta-y)\right)$$
$$=\nabla_\theta\left(\frac{1}{2}(\theta^TX^TX\theta-\theta^TX^Ty-y^TX\theta+y^Ty)\right)$$
$$=\frac{1}{2}\left(2X^TX\theta-X^Ty-(y^TX)^T\right)$$
$$=X^TX\theta-X^Ty$$

标注$\theta^TX^TX\theta$对θ求偏导得：$2X^TX\theta$

- 偏导等于0：$\theta=\left(X^TX\right)^{-1}X^Ty$ 　　偏导为0得极值点，由实际意义得：J(θ)函数取得最大值 　　8

问题来了：根据公式8代入数据可直接求得θ,没有学习的过程，而且不一定存在可逆矩阵 ?

四：真正机器学习的求解方法：梯度下降

1、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，但是其他问题不可以，线性回归是个特例）

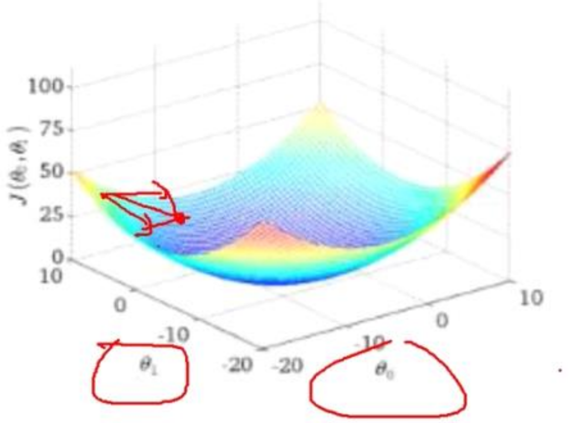
常规套路：机器学习的套路就是我交给机器一堆数据，然后发告诉他什么样的学习方式是对的（目标函数），然后让它

常规套路：机器学习的套路就是我交给机器一堆数据，然后发告诉他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：一口吃不成一个胖子，我们要一步步的完成迭代（每次优化一点点，累计起来就是个大成绩了）

目标函数：J(θ_0,θ_1)=$\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})$

J是分别对两个参数求导更新，$\frac{\partial J}{\partial\theta0}$，$\frac{\partial J}{\partial\theta1}$，根据两个导数值（各自的方向）产生新的位置（新的方向）



寻找山谷最低点，也就是我们的目标函数终点（什么样的参数能使的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，走快了容易”跌倒“

3、按照**方向与步伐**更新我们的参数

2、三种梯度下降方法（工程上的）

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

=$\frac{1}{2m}$

求导

- 批量梯度下降（选取M个样本）：$\frac{\partial J(\theta)}{\partial\theta_j}=-\frac{1}{m}\sum_{i=1}^{\mathrm{m}}(y^{(i)}-h_\theta\left(x^{(i)}\right))\,x_j^i$　　算出M个样本点的平均梯度（对一个参数θ_j*进行求导*）

  更新θ_j：　$\theta_j=\theta_j+\frac{1}{m}\sum_{i=1}^{\mathrm{m}}(y^{(i)}-h_\theta\left(x^{(i)}\right))\,x_j^i$

  （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：　$\theta_j=\theta_j+(y^i-h_\theta(x^i))x_j^i$　　(算一个样本)

  (每次找一个样本迭代，速度快，但不一定每次都朝着收敛方向)

- 小批量梯度下降法(miniBatch)：$\theta_j = \theta_j - \alpha \frac{1}{10} \sum_{k=i}^{i+9} \left(y^{(k)} - h_\theta\left(x^{(k)}\right)\right) x_j^k$ 选择10个样本，一般选64

(每次更新选择一小部分数据来算，实用！！！)

3、学习率（步长）

学习率对结果会产生巨大影响，一般小一些。就是上面的α

如何选择：从小的时候，不行再小

批量处理：32,64,128都可以，很多时候还得考虑内存和效率

正则化

2022年4月30日, 星期六　　17:07

1、8式(X^TX)不一定可逆，表现为可能会出现过拟合

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

2、加入正则化项

$$J\ (\theta)\ = \sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 \quad + \quad \lambda\theta^T\theta$$

正则项

J(θ):

$$\begin{aligned} &= \left(\theta^TX^T - Y^T\right)(X\theta - Y) + \lambda\theta^T\theta \\ &=\theta^TX^TX\theta - 2\theta^TX^TY + Y^T + \lambda\theta^T\theta \\ &=\theta^T\left(\theta^T\theta + \lambda I\right)\theta - 2\theta^TX^TY + Y^T \end{aligned}$$

对J(θ)求导：

$$\frac{\partial J(\theta)}{\partial\theta} = 2\left(\theta^T\theta + \lambda I\right)\theta - 2\theta^TY = 0$$

$$\theta = (X^TX + \lambda I)^{-1}X^TY$$

这样的话$(X^TX + \lambda I)^{-1}$就是可逆的了。就可以防止过拟合

加入正则项相当于什么呢?

$$J\ (\theta)\ = \sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 \quad + \quad \lambda\theta^T\theta$$

相当于加了一个约束条件，就是**拉格朗日乘数法**，在约束条件下求最优解。

约束条件：$\theta_1^2 + \theta_2^2 + \theta_3^2 + \cdots\cdots \le W$

$$J(\theta) = \sum_{i=1}^{m}(\quad)^2$$

$$L(\theta) = J(\theta) + \lambda(\theta_1^2 + \theta_2^2 + \cdots - W)$$

$$L(\theta) = J(\theta) + \lambda(\theta_1^2 + \theta_2^2 + \cdots - W)$$

$$\begin{cases} \dfrac{\partial L}{\partial \theta_1} = 0 \\ \dfrac{\partial L}{\partial \theta_2} = 0 \\ \vdots \\ \dfrac{\partial L}{\partial \lambda} = 0 \end{cases}$$

在梯度下降中的表现

L2正则化是怎么避免过拟合的呢？先推导一下看看：

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{\partial C0}{\partial \theta} + \frac{\lambda}{n}\theta^2$$

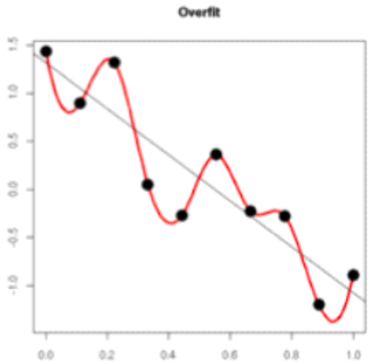
$$\frac{\partial J(\theta)}{\partial b} = \frac{\partial C0}{\partial b}$$

可以发现L2正则化项对b的更新没有影响，但是对θ的更新有影响：

$$\theta \xrightarrow{更新} \theta - \frac{\partial C0}{\partial \theta} - \frac{\lambda}{n}\theta^2$$
$$= \left(1 - \frac{\lambda}{n}\right)\theta - \frac{\partial C0}{\partial \theta}$$

在不使用L2正则化时，求导结果中w前系数为1，现在w前面系数为 1−ηλ/n ，因为η、λ、n都是正的，所以 1−η λ/n小于1，它的效果是减小w，这也就是权重衰减（weight decay）的由来。当然考虑到后面的导数项，w最终的值可能增大也可能减小。

到目前为止，我们只是解释了L2正则化项有让w“变小”的效果，但是还没解释为什么w“变小”可以防止overfitting？一个所谓“显而易见”的解释就是：更小的权值w，从某种意义上说，表示网络的复杂度更低，对数据的拟合刚刚好（这个法则也叫做奥卡姆剃刀），而在实际应用中，也验证了这一点，L2正则化的效果往往好于未经正则化的效果。当然，对于很多人（包括我）来说，这个解释似乎不那么显而易见，所以这里添加一个稍微数学一点的解释（引自知乎）：



过拟合的时候，拟合函数的系数往往非常大，为什么？如下图所示，过拟合，就是拟合函数需要顾忌每一个点，最终形成的拟合函数波动很大。在某些很小的区间里，函数值的变化很剧烈。这就意味着函数在某些小区间里的导数值（绝对值）非常大，由于自变量值可大可小，所以只有系数足够大，才能保证导数值很大。