线性回归与最小二乘法

杨航锋

本文先以一元线性回归为例推导出一元线性回归方程，然后再推导出更一般化的线性回归方程，在推导的过程中采取两种不同的方式：一是从样本数据出发，二是从统计理论着手。最后我们会发现，采用两种不同的方式最后推导出的线性回归模型的损失函数都会殊途同归。线性回归简单点讲就是对已知的样本数据进行最优拟合，然后通过拟合出的线性回归方程进行预测。更多文章见GitHub地址

一元线性回归

其实我们初中时就接触过一元线性回归方程，那时只需要记住两个参数α 和β 直接套公式即可进行预测，现在完整的推导出一元线性回归方程。设一元线性回归方程为$\hat{y}=\alpha x+\beta$ ，数据样本点为(x_1,y_1)，(x_2,y_2)，$\ldots$，(x_n,y_n)，要使得这n个样本点落在在一元线性回归方程附近，不妨假设误差为ε，使得每个样本点都落在一元线性回归方程上。因此有$\hat{y_i}=y_i+\varepsilon_i$恒成立，所以，回归直线应满足的条件是：全部观测值与对应的回归估计值的误差平方和最小，即：

$$\begin{aligned}\arg\min_{\alpha,\beta}\sum_{i=1}^{n}\varepsilon_i^2&=\arg\min_{\alpha,\beta}\sum_{i=1}^{n}(y_i-\hat{y_i})^2\\&=\arg\min_{\alpha,\beta}\sum_{i=1}^{n}(y_i-\alpha x_i-\beta)^2\end{aligned}$$

令$J(\alpha,\beta)=\sum_{i=1}^{n}(y_i-\alpha x_i-\beta)^2$原问题就转化为求二元函数的极小值，由微积分相关知识可知

$$\begin{aligned}\nabla_\alpha J(\alpha,\beta)&=-2\sum_{i=1}^{n}(y_i-\alpha x_i-\beta)x_i\\&=-2\sum_{i=1}^{n}x_iy_i+2\alpha\sum_{i=1}^{n}x_i^2+2\beta\sum_{i=1}^{n}x_i\\\nabla_\beta J(\alpha,\beta)&=-2\sum_{i=1}^{n}(y_i-\alpha x_i-\beta)\\&=-2\sum_{i=1}^{n}y_i+2\alpha\sum_{i=1}^{n}x_i+2n\beta\end{aligned}$$

然后令$\nabla_\alpha J(\alpha,\beta)=0$和$\nabla_\beta J(\alpha,\beta)=0$即可求出$\alpha$、$\beta$的值

$$
\begin{aligned}
&\nabla_\beta J(\alpha,\beta)=0\\
&\Rightarrow \sum_{i=1}^{n} y_i=\alpha\sum_{i=1}^{n} x_i+n\beta\\
&\Rightarrow \bar{y}=\alpha\bar{x}+\beta
\end{aligned}
$$

$$
\begin{aligned}
&\nabla_\alpha J(\alpha,\beta)=0\\
&\Rightarrow \sum_{i=1}^{n} x_i y_i=\alpha\sum_{i=1}^{n} x_i^2+\beta\sum_{i=1}^{n} x_i\\
&\Rightarrow \alpha=\frac{\sum\limits_{i=1}^{n}(x_i y_i-\bar{y}x_i)}{\sum\limits_{i=1}^{n}(x_i^2-\bar{x}x_i)}=\frac{\sum\limits_{i=1}^{n} x_i y_i-\frac{1}{n}(\sum\limits_{i=1}^{n} x_i)(\sum\limits_{i=1}^{n} y_i)}{\sum\limits_{i=1}^{n} x_i^2-\frac{1}{n}(\sum\limits_{i=1}^{n} x_i)(\sum\limits_{i=1}^{n} x_i)}\\
&=\frac{\sum\limits_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sum\limits_{i=1}^{n}(x_i-\bar{x})^2}
\end{aligned}
$$

根据：

$$
\begin{aligned}
\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})&=\sum_{i=1}^{n} x_i y_i-\frac{1}{n}(\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)\\
\sum_{i=1}^{n}(x_i-\bar{x})^2&=\sum_{i=1}^{n} x_i^2-\frac{1}{n}(\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} x_i)
\end{aligned}
$$

至此，一元线性回归方程就拟合出来了，上面的这些公式推导看起来很复杂其实如果引入矩阵表示，最后的结果将会很简洁。

一般化的线性回归算法推导

对于给定了m个样本点($x^{(1)}$,$y^{(1)}$)，($x^{(2)}$,$y^{(2)}$)，…，($x^{(m)}$,$y^{(m)}$)，其中$x^{(i)}\in\mathbb{R}^n$。定义假设函数为$h_\theta(x)$，即$h_\theta(x)$为最终的拟合函数，θ为待拟合参数也称作权重。

$$
\begin{aligned}
h_\theta(x)&=\theta_0+\theta_1x_1+\cdots+\theta_nx_n\\
&=\begin{bmatrix}\theta_0, & \theta_1, & \cdots, & \theta_n\end{bmatrix}\begin{bmatrix}1\\x_1\\\vdots\\x_n\end{bmatrix}\\
&=\theta^Tx
\end{aligned}
$$

从样本数据出发推导损失函数

在样本数据中$y^{(i)}$是实际存在值而$h_\theta(x^{(i)})$对应的是模型预测值，显然如果想要模型预测的效果好，那么对应的误差就要小，假设函数在任意样本点的误差为$|h_\theta(x^{(i)})-y^{(i)}|$则$m$个样本点的误差和为 $\sum\limits_{i=1}^{m}|h_\theta(x^{(i)})-y^{(i)}|$，因此问题就转化为求解 $\arg\min\limits_\theta\sum\limits_{i=1}^{m}|h_\theta(x^{(i)}-y^{(i)}|$ 为了后续求解最优值(绝对值函数不好求导)，所以损失函数采用了误差平方和的形式 $\arg\min\limits_\theta\sum\limits_{i=1}^{m}(h_\theta(x^{(i)}-y^{(i)})^2$ 。

从统计学理论出发推导损失函数

为什么线性回归问题的损失函数会是误差平方和的形式？这里从统计理论上进行解释，对于给定的$y^{(i)}$总能找到$\varepsilon^{(i)}$使得这个等式成立$y^{(i)}=h_\theta(x^{(i)})+\varepsilon^{(i)}$， $\varepsilon^{(i)}$代表真实值和预测值之间的误差且$\varepsilon^{(i)}\sim\mathcal{N}(0,\sigma^2)$,简单解释下为什么误差$\varepsilon^{(i)}$会服从均值为零的正态分布，误差的产生有很多种因素的影响，误差可以看作是这些因素(随机变量)之和共同作用而产生的，由中心极限定理可知随机变量和的分布近似的服从正态分布；更通俗易懂的解释是，当你在选择$h_\theta(x)$时主观的会认定这个$h_\theta(x)$是比较符合样本数据的，比如对一些样本数据可视化后，发现样本数据明显是趋近于一条直线，而你在对$h_\theta(x)$的选择上肯定会选择直线方程作为$h_\theta(x)$而不会选择多项式函数作为$h_\theta(x)$。而这种$h_\theta(x)$一旦选定，可以认为大部分数据都在$h_\theta(x)$的附近，因此误差大部分集中在零值附近所以$\mathcal{N}(0,\sigma^2)$作为$\varepsilon^{(i)}$的先验分布是比较合理的。

随机变量$\varepsilon^{(i)}$的概率密度函数为：

$$p(\varepsilon^{(i)})=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(\varepsilon^{(i)})^2}{2\sigma^2}}$$

代入$h_\theta(x^{(i)})$，$y^{(i)}$则

$$p(y^{(i)}|x^{(i)};\theta)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(\theta^Tx^{(i)}-y^{(i)})^2}{2\sigma^2}}$$

这里的$p(y^{(i)}|x^{(i)};\theta)$并不代表条件概率，只是一个记号它表示给定$x^{(i)}$，$y^{(i)}$ 和一组θ后的概率密度函数。由最大似然估计可知：

$$\begin{aligned}L(\theta)&=\prod_{i=1}^{m}p(y^{(i)}|x^{(i)};\theta)\\&=\prod_{i=1}^{m}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(\theta^Tx^{(i)}-y^{(i)})^2}{2\sigma^2}}\end{aligned}$$

对$L(\theta)$取对数从而得到对数化最大似然估计函数

$$\begin{aligned}\mathcal{L}(\theta)&=log(L(\theta))\\&=\log\prod_{i=1}^{m}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(\theta^Tx^{(i)}-y^{(i)})^2}{2\sigma^2}}\\&=\sum_{i=1}^{m}\log\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(\theta^Tx^{(i)}-y^{(i)})^2}{2\sigma^2}}\\&=m\log\frac{1}{\sqrt{2\pi}\sigma}-\frac{1}{2\sigma^2}\sum_{i=1}^{m}(\theta^Tx^{(i)}-y^{(i)})^2\end{aligned}$$

求解最大化对数似然函数可得：

$$
\begin{aligned}
\arg\max_{\theta} L(\theta) &\Leftrightarrow \arg\max_{\theta} \mathcal{L}(\theta) \\
&\Leftrightarrow \arg\max_{\theta} \left\{ m\log\frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2}\sum_{i=1}^{m}(\theta^T x^{(i)} - y^{(i)})^2 \right\} \\
&\Leftrightarrow \arg\max_{\theta} -\frac{1}{2\sigma^2}\sum_{i=1}^{m}(\theta^T x^{(i)} - y^{(i)})^2 \\
&\Leftrightarrow \arg\min_{\theta} \frac{1}{2\sigma^2}\sum_{i=1}^{m}(\theta^T x^{(i)} - y^{(i)})^2 \\
&\Leftrightarrow \arg\min_{\theta} \sum_{i=1}^{m}(\theta^T x^{(i)} - y^{(i)})^2 \\
&\Leftrightarrow \arg\min_{\theta} \sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2
\end{aligned}
$$

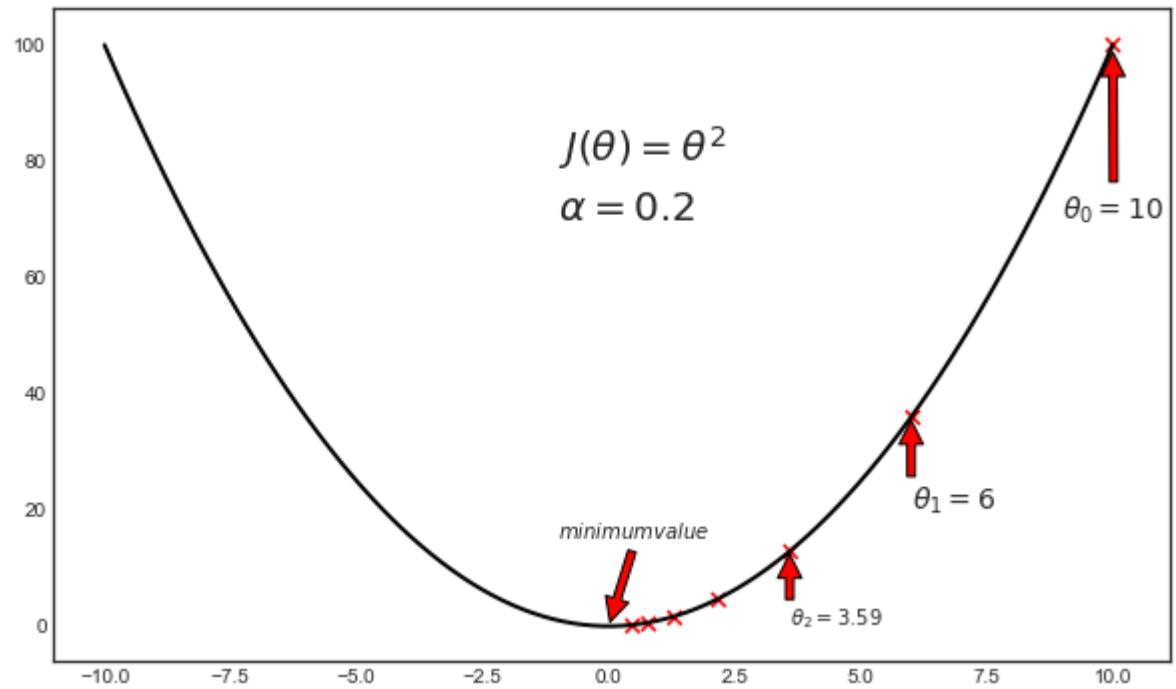
可以发现两种方法推导出的损失函数都是一样的，下面定义线性回归模型的损失函数$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$，其实只是在先前的最优化模型前面乘了一个常系数$\frac{1}{2}$对最优化的结果并不会产生影响。

梯度下降算法求解凸优化问题

在机器学习算法中，求解最小化损失函数时，可以通过梯度下降法来一步步进行迭代求解，从而得到最小化损失函数的模型参数值，梯度下降算法不一定能够找到全局的最优解，有可能是一个局部最优解。然而，如果损失函数是凸函数，那么梯度下降法得到的解就一定是全局最优解。 在这里不加证明的给出梯度下降法的迭代格式:

$$\theta_j := \theta_j - \alpha\nabla_{\theta_j} J(\theta)$$

其中θ_j为假设函数$h_\theta(x)$的参数值，$\alpha \in (0, 1]$且为常数代表模型学习速率，$\nabla_{\theta_j} J(\theta)$为损失函数$J(\theta)$对参数$\theta_j$的梯度。举一个简单的例子直观的解释一下梯度下降算法的流程和有效性，令$J(\theta) = \theta^2$使用梯度下降算法求$J(\theta)$的最小值的过程如下图所示。



可以看出对于二次函数梯度下降算法最终会收敛到全局最小值。

下面使用梯度下降算法求解上面推导出的线性回归模型的损失函数 $J(\theta)=\frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2$，为了实现该算法首先要求出损失函数$J(\theta)$对参数$\theta_j$的梯度：

$$\begin{aligned}\nabla_{\theta_j}J(\theta)&=\nabla_{\theta_j}\frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2\\&=2\cdot\frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})\nabla_{\theta_j}(h_\theta(x^{(i)})-y^{(i)})\\&=\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})\nabla_{\theta_j}\theta^Tx^{(i)}\\&=\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})x_j^{(i)}\end{aligned}$$

因此在线性回归模型中利用所有的样本数据，训练梯度下降算法的完整迭代格式为

$$\theta_j:=\theta_j-\alpha\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})x_j^{(i)}\quad(j\ \ for\ \ 0\ \sim\ n)$$

上述迭代过程每次迭代都会使用所有的样本数据，数学上已经证明线性回归模型的损失函数通过梯度下降算法求解一定会全局收敛，所以如果要编程实现该算法只需要控制迭代次数即可，不过对于线性回归模型的求解一般不用梯度下降算法，还有更容易实现且更快捷的形式—正规方程。

正规方程

对损失函数$J(\theta)=\frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2$改写成矩阵乘法的形式，在此之前需要先定义一些矩阵，不妨令：

$$Y=\begin{bmatrix}(y^{(1)})\\(y^{(2)})\\\vdots\\(y^{(m)})\end{bmatrix}$$

$$X=\begin{bmatrix}(x^{(1)})^T\\(x^{(2)})^T\\\vdots\\(x^{(m)})^T\end{bmatrix}=\begin{bmatrix}1&x_1^{(1)}&x_2^{(1)}&\cdots&x_n^{(1)}\\1&x_1^{(2)}&x_2^{(2)}&\cdots&x_n^{(2)}\\\vdots&\vdots&\vdots&\ddots&\vdots\\1&x_1^{(m)}&x_2^{(m)}&\cdots&x_n^{(m)}\end{bmatrix}$$

由于$h_\theta(x^{(i)})=\theta^Tx^{(i)}=(x^{(i)})^T\theta$，因此可以得出

$$X\theta-Y=\begin{bmatrix}(x^{(1)})^T\theta\\(x^{(2)})^T\theta\\\vdots\\(x^{(m)})^T\theta\end{bmatrix}-\begin{bmatrix}(y^{(1)})\\(y^{(2)})\\\vdots\\(y^{(m)})\end{bmatrix}=\begin{bmatrix}(x^{(1)})^T\theta-(y^{(1)})\\(x^{(2)})^T\theta-(y^{(2)})\\\vdots\\(x^{(m)})^T\theta-(y^{(m)})\end{bmatrix}$$

然后根据$\sum_{i=1}^{n}\phi_i^2=\phi^T\phi$,综上可以得出$J(\theta)$的矩阵形式

$$J(\theta)=\frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2=\frac{1}{2}(X\theta-Y)^T(X\theta-Y)$$

最后求解 $\arg\min\limits_{\theta}\frac{1}{2}(X\theta-Y)^T(X\theta-Y)$ 即可。

在求解上述优化问题之前先简单的介绍下矩阵求导法则

$$
\begin{aligned}
\nabla_x x^T b &= \nabla_x \begin{bmatrix} x_1, & x_2, & \cdots, & x_n \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \\
&= \nabla_{x_i} \sum_{i=1}^{n} x_i b_i \\
&= b \\
\nabla_x Ax &= \nabla_x \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \\
&= \begin{bmatrix} \nabla_{x_i} \sum\limits_{i=1}^{n} a_{1i} x_i \\ \nabla_{x_i} \sum\limits_{i=1}^{n} a_{2i} x_i \\ \vdots \\ \nabla_{x_i} \sum\limits_{i=1}^{n} a_{ni} x_i \end{bmatrix} \\
&= A^T
\end{aligned}
$$

所以

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \ \frac{1}{2}(X\theta-Y)^T(X\theta-Y) \\
&= \frac{1}{2}\nabla_\theta \ (\theta^T X^T - Y^T)(X\theta-Y) \\
&= \frac{1}{2}\nabla_\theta \ (\theta^T X^T X\theta - \theta^T X^T Y - Y^T X\theta + Y^T Y) \\
&= \frac{1}{2}(2X^T X\theta - 2X^T Y) \\
&= X^T X\theta - X^T Y
\end{aligned}
$$

称 $X^T X\theta = X^T Y$ 为正规方程，因此$\theta = (X^T X)^{-1} X^T Y$，实际上$X^T X$不可逆的情况非常少就算$X^T X$真的是不可逆也无妨，可以先对原始数据进行特征筛选或者正则化即可。

总结

一般情况下对于求解线性回归模型通常采用正规方程的方式，回过头来对于文章开头的求解一元线性回归方程系数的问题可以大大化简，具体参数形式为

$$
\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \left(\begin{bmatrix} 1 & x_1^T \\ 1 & x_2^T \\ \vdots & \vdots \\ 1 & x_n^T \end{bmatrix}^T \begin{bmatrix} 1 & x_1^T \\ 1 & x_2^T \\ \vdots & \vdots \\ 1 & x_n^T \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & x_1^T \\ 1 & x_2^T \\ \vdots & \vdots \\ 1 & x_n^T \end{bmatrix}^T \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}
$$

可以看出通过正规方程求解线性回归模型就转化为如何构造X矩阵，只要X矩阵构造出来后剩下的就是交给计算机做矩阵乘法运算就可以了。