Table of Contents

1 任务1：线性回归

【学习任务】
1.线性回归损失函数的极大似然推导：西瓜书公式3.4除了用最小二乘法以外，怎么用极大似然推得？
2.一元线性回归的参数求解公式推导：西瓜书公式3.7和3.8怎么推来的？
3.多元线性回归的参数求解公式推导：西瓜书公式3.10和3.11怎么推来的？
4.线性回归损失函数的最优化算法：什么是批量梯度下降、随机梯度下降、小批量梯度下降？

1.1 线性回归损失函数的极大似然推导

线性回归假设函数为：$y = \theta^T X$
拟合数据，就是把误差减到最小。
误差：$\epsilon = y - \theta^T$

假设误差服从正态分布，误差最小也就是期望为0。$\epsilon \sim N(0, \sigma^2)$
极大似然估计就是使所有样本最接近参数,也就是似然函数最大。

求似然函数：

$$L(\theta) = \prod_{i=1}^{m} p(\epsilon) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{\epsilon^2}{2\sigma^2}}$$

$$L(\theta) = \frac{1}{\sqrt{2\pi}^m} \frac{1}{\sigma^m} e^{-\sum_{i=1}^{m} \frac{\epsilon^2}{2\sigma^2}}$$

$$L(\theta) = \sqrt{2\pi}^{-m} \sigma^{-m} e^{-\sum_{i=1}^{m} \frac{(y-\theta^T X)^2}{2\sigma^2}}$$

两边取ln:

$$\ln L(\theta) = -m \ln(\sqrt{2\pi}) - mln(\sigma) - \frac{1}{2\sigma^2} \sum_{i=1}^{m} (y - \theta^T X)^2$$

要使似然函数最大，$\sum_{i=1}^{m} (y - \theta^T X)^2$就要最小。也就是误差平法和最小。

1.2 一元线性回归的参数求解公式推导

对于样本集$D=(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)$线性回归模型就是要找到一个模型$f(x)=wx+b$使得$\forall i\in[1,m]$都有$f(x_i)$尽可能地接近于$y_i$所以我们使每一个样本的预测值与真实值的差的平方和最小，即：

$$\begin{aligned}(w^*,b^*)&=\arg_{w,b}\min\sum_{i=1}^{m}(f(x_i)-y_i)^2\\&=\arg_{w,b}\min\sum_{i=1}^{m}(wx_i+b-y_i)^2\end{aligned}\tag{1}$$

这可以看做是对多元函数$h(w,b)=\sum_{i=1}^{m}(wx_i+b-y_i)^2$求最小值，则按照微积分的知识，函数$h$对$w$ 和b分别求偏导，并令二者的偏导数为零，则此时对应的w^*和 b^* 便是使得h取最小的值。故，求偏导后得到：

$$\begin{aligned}\frac{\partial h(w,b)}{\partial w}&=\sum_{i=1}^{m}2(wx_i+b-y_i)*x_i\\&=2\left(w\sum_{i=1}^{m}x_i^2-\sum_{i=1}^{m}(y_i-b)x_i\right)\end{aligned}\tag{2}$$

$$\begin{aligned}\frac{\partial h(w,b)}{\partial b}&=\sum_{i=1}^{m}2(wx_i+b-y_i)*1\\&=2\left(mb-\sum_{i=1}^{m}(y_i-wx_i)\right)\end{aligned}\tag{3}$$

分别令二者等于0便可以得到w 和b的最优解：

$$w=\frac{\sum_{i=1}^{m}y_i(x_i-\bar{x})}{\sum_{i=1}^{m}x_i^2-\frac{1}{m}\left(\sum_{i=1}^{m}x_i\right)^2}\tag{4}$$

$$b=\frac{1}{m}\sum_{i=1}^{m}(y_i-wx_i)\tag{5}$$

这其中，式(5)是很容易从式(3)得到的，此处不赘述，但是式(4)的得来并没有这么直观，我们来详细推导一下:

令式(2)等于 0 则有：

$$w\sum_{i=1}^{m}x_i^2=\sum_{i=1}^{m}x_iy_i-b\sum_{i=1}^{m}x_i\tag{6}$$

将式(5)代入式(6)得：

$$\begin{aligned}w\sum_{i=1}^{m}x_i^2&=\sum_{i=1}^{m}x_iy_i-\frac{1}{m}\sum_{i=1}^{m}x_i\left(\sum_{i=1}^{m}(y_i-wx_i)\right)\\&=\sum_{i=1}^{m}x_iy_i-\bar{x}\left(\sum_{i=1}^{m}y_i\right)+\frac{1}{m}\sum_{i=1}^{m}x_i*\sum_{i=1}^{m}wx_i\\&=\sum_{i=1}^{m}x_iy_i-\bar{x}\sum_{i=1}^{m}y_i+\frac{w}{m}\left(\sum_{i=1}^{m}x_i\right)^2\end{aligned}\tag{7}$$

式(7)最右的平方项移项到左边可得：

$$w\sum_{i=1}^{m}x_i^2-\frac{w}{m}\left(\sum_{i=1}^{m}x_i\right)^2=\sum_{i=1}^{m}x_iy_i-\bar{x}\sum_{i=1}^{m}y_i \tag{8}$$

$$w\left[\sum_{i=1}^{m}x_i^2-\frac{1}{m}\left(\sum_{i=1}^{m}x_i\right)^2\right]=\sum_{i=1}^{m}x_iy_i-\bar{x}\sum_{i=1}^{m}y_i \tag{9}$$

式(9)变形后即可得式(4)，至此，w 和b的表达式都只与观测样本有关，所以可以通过样本观测值来估计w 和b 。

1.3 多元线性回归的参数求解公式推导

对于样本集$D=(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)$其中$\boldsymbol{x}_i=[x_i^{(1)},x_i^{(2)}\cdots x_i^{(d)}]^T$表示一条样本数据有 d 个属性，我们的目标是寻找 d 维列向量 w 和常数 b，使得模型

$$\$f(\boldsymbol{x}_i)=\boldsymbol{w}^T\boldsymbol{x}_i+b \tag{1}$$

所得的预测值与真实值 y_i 尽可能接近。

把常数b放入权值向量w得到一个 $(d+1)$ 维的权值向量$\hat{\boldsymbol{w}}=(\boldsymbol{w};b)$，同时在每个样本实例中添加第 (d+1)(d+1) 个属性，置为 11，xi^=(xi;1)xi^=(xi;1)。将样本所有属性排列为矩阵可以得到：

$$\boldsymbol{X}=\begin{bmatrix}\hat{\boldsymbol{x_1}}\\ \hat{\boldsymbol{x_2}}\\ \vdots\\ \hat{\boldsymbol{x_m}}\end{bmatrix}$$

令$\boldsymbol{y}=(y_1,y_2\cdots y_m)^T$，同一元线性回归中最小化预测值与真实值误差平方和一样，在多元回归中我们要最小化

$$||\boldsymbol{y}-\boldsymbol{X}\hat{\boldsymbol{w}}||^2$$

即：

$$\boldsymbol{w}^*=\arg_{\hat{\boldsymbol{w}}}\min(\boldsymbol{y}-\boldsymbol{X}\hat{\boldsymbol{w}})^T(\boldsymbol{y}-\boldsymbol{X}\hat{\boldsymbol{w}})$$

此处将最小化的目标函数视为 $\hat{\boldsymbol{w}}$ 的“单变量”函数，令$h(\hat{\boldsymbol{w}})=(\boldsymbol{y}-\boldsymbol{X}\hat{\boldsymbol{w}})^T(\boldsymbol{y}-\boldsymbol{X}\hat{\boldsymbol{w}})$，求它的最小值只需其对 $\hat{\boldsymbol{w}}$求导，导数值为 0 时 $\hat{\boldsymbol{w}}$ 的取值即为所求。

$$\begin{aligned}\frac{\partial h(\hat{\boldsymbol{w}})}{\partial\hat{\boldsymbol{w}}}&=\frac{\partial[(\boldsymbol{y}-\boldsymbol{X}\hat{\boldsymbol{w}})^T(\boldsymbol{y}-\boldsymbol{X}\hat{\boldsymbol{w}})]}{\partial\hat{\boldsymbol{w}}}\\ &=2\frac{\partial(\boldsymbol{y}-\boldsymbol{X}\hat{\boldsymbol{w}})^T}{\partial\hat{\boldsymbol{w}}}(\boldsymbol{y}-\boldsymbol{X}\hat{\boldsymbol{w}}) &(2)\\ &=2\frac{\partial\boldsymbol{y}^T}{\partial\hat{\boldsymbol{w}}}(\boldsymbol{y}-\boldsymbol{X}\hat{\boldsymbol{w}})-2\frac{\partial(\boldsymbol{X}\hat{\boldsymbol{w}})^T}{\partial\hat{\boldsymbol{w}}}(\boldsymbol{y}-\boldsymbol{X}\hat{\boldsymbol{w}}) &(3)\\ &=0-2\boldsymbol{X}^T(\boldsymbol{y}-\boldsymbol{X}\hat{\boldsymbol{w}}) &(4)\\ &=2\boldsymbol{X}^T(\boldsymbol{X}\hat{\boldsymbol{w}}-\boldsymbol{y}) &(5)\end{aligned}$$

上述步骤(2)运用了:

$$
\begin{aligned}
\frac{d(\boldsymbol{x}^T\boldsymbol{x})}{d\boldsymbol{y}} &= \frac{d(\boldsymbol{x}^T)}{d\boldsymbol{y}}\boldsymbol{x} + \frac{d(\boldsymbol{x}^T)}{d\boldsymbol{y}}\boldsymbol{x} \\
&= 2\frac{d(\boldsymbol{x}^T)}{d\boldsymbol{y}}\boldsymbol{x}
\end{aligned} \tag{6}
$$

步骤(3)简单求导的拆分

步骤(4)第一项中$\boldsymbol{y}^T$ 与$\hat{\boldsymbol{w}}$无关，所以求导为0；第二项根据：

$$
\begin{aligned}
\frac{d(\boldsymbol{A}\boldsymbol{x})^T}{d\boldsymbol{x}} &= \frac{d(\sum_{i=1}^n a_{1i}x_i, \sum_{i=1}^n a_{2i}x_i, \cdots, \sum_{i=1}^n a_{mi}x_i,)}{d\boldsymbol{x}} \\
&= \begin{bmatrix} \frac{\partial\sum_{i=1}^n a1ixi}{\partial x1} & \frac{\partial\sum_{i=1}^n a2ixi}{\partial x1} & \cdots & \frac{\partial\sum_{i=1}^n amixi}{\partial x1} \\ \frac{\partial\sum_{i=1}^n a1ixi}{\partial x2} & \frac{\partial\sum_{i=1}^n a2ixi}{\partial x2} & \cdots & \frac{\partial\sum_{i=1}^n amixi}{\partial x2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\sum_{i=1}^n a1ixi}{\partial xn} & \frac{\partial\sum_{i=1}^n a2ixi}{\partial xn} & \cdots & \frac{\partial\sum_{i=1}^n amixi}{\partial xn} \end{bmatrix} \\
&= \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix} = \boldsymbol{A}^T
\end{aligned} \tag{7}
$$

令式(5)为0，此时的$\hat{\boldsymbol{w}}$即为所求$\boldsymbol{w}^*$

$$
\begin{aligned}
&\because 2\boldsymbol{X}^T(\boldsymbol{X}\hat{\boldsymbol{w}} - \boldsymbol{y}) = 2\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{w}} - 2\boldsymbol{X}^T\boldsymbol{y} = 0 \\
&\therefore \boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{w}} = \boldsymbol{X}^T\boldsymbol{y} \\
&\therefore \hat{\boldsymbol{w}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \\
&\therefore \boldsymbol{w}^* = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}
\end{aligned}
$$

1.4 线性回归损失函数的最优化算法

为了便于理解，这里我们将使用只含有一个特征的线性回归来展开。此时线性回归的假设函数为：

其中
表示样本数。

对应的目标函数（代价函数）即为：

1.4.1 批量梯度下降（Batch Gradient Descent，BGD）

批量梯度下降法是最原始的形式，它是指在每一次迭代时使用所有样本来进行梯度的更新。从数学上理解如下：

（1）对目标函数求偏导：

$$
\frac{\Delta J(\theta_0, \theta_1)}{\Delta\theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}
$$

其中 $i = 1, 2, \ldots, m$表示样本数，$j = 0, 1$表示特征数，这里我们使用了偏置项$x_0^{(i)} = 1$。

（2）每次迭代对参数进行更新：

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

伪代码形式为：

repeat{

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$(forj = 0, 1)$$

}

优点：

（1）一次迭代是对所有样本进行计算，此时利用矩阵进行操作，实现了并行。

（2）由全数据集确定的方向能够更好地代表样本总体，从而更准确地朝向极值所在的方向。当目标函数为凸函数时，BGD一定能够得到全局最优。

缺点：

（1）当样本数目 m 很大时，每迭代一步都需要对所有样本计算，训练过程会很慢。

从迭代的次数上来看，BGD迭代的次数相对较少。

1.4.2 随机梯度下降（Stochastic Gradient Descent，SGD）

随机梯度下降法不同于批量梯度下降，随机梯度下降是每次迭代使用一个样本来对参数进行更新。使得训练速度加快。 对于一个样本的目标函数为：

$$J^{(i)}(\theta_0, \theta_1) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$$

（1）对目标函数求偏导：

$$\frac{\Delta J^{(i)}(\theta_0, \theta_1)}{\theta_j} = (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

（2）参数更新：

$$\theta_j := \theta_j - \alpha (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

注意，这里不再有求和符号

优点：

（1）由于不是在全部训练数据上的损失函数，而是在每轮迭代中，随机优化某一条训练数据上的损失函数，这样每一轮参数的更新速度大大加快。

缺点：

（1）准确度下降。由于即使在目标函数为强凸函数的情况下，SGD仍旧无法做到线性收敛。

（2）可能会收敛到局部最优，由于单个样本并不能代表全体样本的趋势。

（3）不易于并行实现。

1.4.3 小批量梯度下降（Mini-Batch Gradient Descent, MBGD）

小批量梯度下降，是对批量梯度下降以及随机梯度下降的一个折中办法。其思想是：每次迭代 使用 batch_size个样本来对参数进行更新。

这里我们假设 batchsize=10 ， 样本数 m=1000 。

伪代码形式为：   repeat{
    for i=1,11,21,31,...,991{

$$\theta_j := \theta_j - \alpha \frac{1}{10} \sum_{k=i}^{(i+9)} (h_\theta(x^{(k)}) - y^{(k)}) x_j^{(k)}$$

(for j =0,1)   }   }

优点：

（1）通过矩阵运算，每次在一个batch上优化神经网络参数并不会比单个数据慢太多。 （2）每次使用一个batch可以大大减小收敛所需要的迭代次数，同时可以使收敛到的结果更加接近梯度下降的效果。(比如上例中的30W，设置batch_size=100时，需要迭代3000次，远小于SGD的30W次)

（3）可实现并行化。

缺点：

（1）batch_size的不当选择可能会带来一些问题。

batcha_size的选择带来的影响：

（1）在合理地范围内，增大batch_size的好处：

a. 内存利用率提高了，大矩阵乘法的并行化效率提高。

b. 跑完一次 epoch（全数据集）所需的迭代次数减少，对于相同数据量的处理速度进一步加快。

c. 在一定范围内，一般来说 Batch_Size 越大，其确定的下降方向越准，引起训练震荡越小。

（2）盲目增大batch_size的坏处：

a. 内存利用率提高了，但是内存容量可能撑不住了。

b. 跑完一次 epoch（全数据集）所需的迭代次数减少，要想达到相同的精度，其所花费的时间大大增加了，从而对参数的修正也就显得更加缓慢。

c. Batch_Size 增大到一定程度，其确定的下降方向已经基本不再变化。

```
In [ ]:
```