机器学习白板推导系列笔记 (3.1∼3.4)

Dexing Huang
dxhuang@bupt.edu.cn
Beijing University of Posts and Telecommunications

日期：2021 年 10 月 28 日

1 线性回归 (Linear Regression)

线性回归是机器学习中最简单的模型, 本节主要讲述求解参数最常用的最小二乘法 (OLS) 矩阵表达及其在几何以及概率上的含义, 最后介绍正则化方法, 分别是 L_1 正则以及 L_2 正则 (岭回归).

先做一些符号上的说明, 假设有数据集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)\}$, 其中 $\mathbf{x}_i \in \mathbb{R}^p, y_i \in \mathbb{R}\,(i = 1, 2, \cdots, N)$, 向量均为列向量, 那么有

$$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N)^T = \begin{pmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Np} \end{pmatrix}_{N\times p}$$

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}_{N\times 1}$$

1.1 最小二乘法及其几何意义

1.1.1 最小二乘法 (OLS)

首先介绍一下线性回归, 以简单的一维为例 (即 $p = 1$), 在初中就学过其表达式为

$$y = f(x) = w \cdot x + b \tag{1}$$

使用最小二乘法就是要找到最优的参数 $\hat{w}$ 和 $\hat{b}$. 下面将问题扩展为 p 维, 式 (1) 变为

$$y = f(\mathbf{x}) = w_1 x_1 + w_2 x_2 + \cdots + w_p x_p + b \tag{2}$$

为了后续的矩阵化表达统一, 通常令$b = w_0$上式化为

$$y = f(\mathbf{x}) = w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_p x_p = \sum_{i=1}^{p} w_i x_i \tag{3}$$

注. $x_0 = 1$

注意式 (3) 是针对一个样本 $(\mathbf{x}, y)$ 的, $\mathbf{w}$ 为列向量, 所以式 (3) 的矩阵形式为

$$y_i = \Big(b, w_1, w_2, \cdots, w_p\Big)_{1\times(p+1)} \begin{pmatrix} 1 \\ x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{pmatrix}_{(p+1)\times 1} = \mathbf{w}^T\mathbf{x}_i \tag{4}$$

使用最小二乘法对参数进行优化, 首先定义损失函数 (loss function)

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{N} ||\mathbf{w}^T\mathbf{x}_i - y_i||^2 \tag{5}$$

将上式转化为矩阵表示

$$\begin{aligned}
\mathcal{L}(\mathbf{w}) &= \sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}_i - y_i)^2 \\
&= \Big(\mathbf{w}^T\mathbf{x}_1 - y_1 \quad \mathbf{w}^T\mathbf{x}_2 - y_2 \quad \cdots \quad \mathbf{w}^T\mathbf{x}_N - y_N\Big) \begin{pmatrix} (\mathbf{w}^T\mathbf{x}_1 - y_1)^T \\ (\mathbf{w}^T\mathbf{x}_2 - y_2)^T \\ \vdots \\ (\mathbf{w}^T\mathbf{x}_N - y_N)^T \end{pmatrix} \\
&= [\mathbf{w}^T \Big(\mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \quad \mathbf{x}_N\Big) - \Big(y_1 \quad y_2 \quad \cdots \quad y_N\Big)] \begin{pmatrix} \mathbf{x}_1^T\mathbf{w} - y_1^T \\ \mathbf{x}_2^T\mathbf{w} - y_2^T \\ \vdots \\ \mathbf{x}_N^T\mathbf{w} - y_N^T \end{pmatrix} \\
&= (\mathbf{w}^T\mathbf{X}^T - \mathbf{Y}^T)(\mathbf{X}\mathbf{w} - \mathbf{Y}) \\
&= \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{w}^T\mathbf{X}^T\mathbf{Y} + \mathbf{Y}^T\mathbf{Y}
\end{aligned}$$

最终得到损失函数的矩阵表示

$$\mathcal{L}(\mathbf{w}) = \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{w}^T\mathbf{X}^T\mathbf{Y} + \mathbf{Y}^T\mathbf{Y} \tag{6}$$

接下来求最优 $\mathbf{w}$ 使得 $\mathcal{L}(\mathbf{w})$ 最小, 即

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) \tag{7}$$

对 $\mathcal{L}(\mathbf{w})$ 求导得

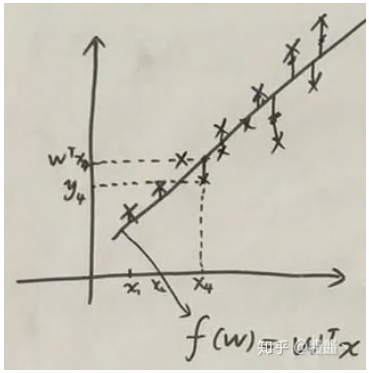
$$\frac{\partial}{\partial \mathbf{w}}\mathcal{L}(\mathbf{w}) = 2\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{X}^T\mathbf{Y} \tag{8}$$

令 $\frac{\partial}{\partial \mathbf{w}}\mathcal{L}(\mathbf{w}) = 0$ 得到 $\mathbf{w}$ 的解析解

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \tag{9}$$

1.1.2 几何意义

OLS 的几何意义可以从两方面来理解, 一是从每一个数据点的误差来看, 每一个数据点对应的真实值 y_i 与估计值 $\mathbf{w}^T\mathbf{x}_i$ 之间的差值便是误差, 因此直观上将所有点的误差求和, 并使得其最小, 便可求得最优的回归函数, 如下图所示.

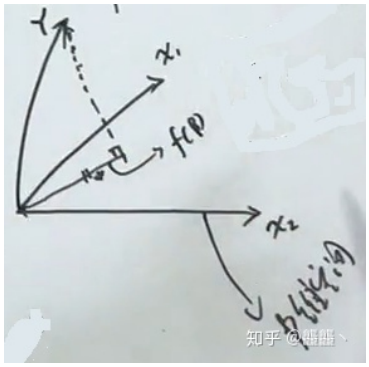


第二种理解方式是在投影的角度上

$$
\begin{aligned}
\mathbf{Xw} &= \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Np} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ x_p \end{pmatrix} \\
&= \begin{pmatrix} x_{11}w_1 + x_{12}w_2 + \cdots + x_{1p}w_p \\ x_{21}w_1 + x_{22}w_2 + \cdots + x_{2p}w_p \\ \vdots \\ x_{N1}w_1 + x_{N2}w_2 + \cdots + x_{Np}w_p \end{pmatrix} \\
&= \begin{pmatrix} w_1 \begin{pmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{N1} \end{pmatrix} + w_2 \begin{pmatrix} x_{12} \\ x_{22} \\ \vdots \\ x_{N2} \end{pmatrix} + \cdots w_p \begin{pmatrix} x_{1p} \\ x_{2p} \\ \vdots \\ x_{Np} \end{pmatrix} \end{pmatrix}
\end{aligned} \tag{10}
$$

注. 省略了偏置项

可以将 **X** 中每一列看作一个向量, **Xw** 便是 **w** 对 **X** 列向量的线性组合. 这里其实可以看作 **X** 的所有列向量组成了一个 p 维空间, 从二维角度来看, 每一个数据点并不都在同一条直线上, 引入到高维空间中, 便是 **Y** 无法由 **X** 的列向量线性表示，即 **Y** 不属于 **X** 的列空间. 因此需要找到 **Y** 在 **X** 列空间上的投影 (即 **X** 列向量的一个线性组合), 将 **Y** 投影在空间中, 便可保留 **Y** 在这一空间上的所有信息. 假设 **Xw** 就是该投影, 如下所示



虚线部分为 $\mathbf{Y}-\mathbf{X}\mathbf{w}$, 其垂直于 $\mathbf{X}$ 列空间 (即与 $\mathbf{X}$ 的每一个列向量都垂直), 因此

$$\mathbf{X}^T(\mathbf{Y}-\mathbf{X}\mathbf{w})=\mathbf{0} \tag{11}$$

同样可得 $\hat{\mathbf{w}}=(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$

2 概率角度看最小二乘法

考虑这么一个情况, 当数据都在一条直线上时是最完美的情况，式 (5) 为 0, 但现实中不可能出现这种情况, 因为数据都带有一定的噪声, 假设噪声 $\epsilon\sim\mathcal{N}(0,\sigma^2)$, 那么

$$y_i=\mathbf{w}^T\mathbf{x}_i+\epsilon \tag{12}$$

因此 $y_i|\mathbf{x}_i,\mathbf{w}\sim\mathcal{N}(\mathbf{w}^T\mathbf{x}_i,\sigma^2)$, 即

$$p(y_i|\mathbf{x}_i,\mathbf{w})=\frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{y_i-\mathbf{w}^T\mathbf{x}_i}{2\sigma^2}) \tag{13}$$

接下来使用极大似然估计求解最优 $\mathbf{w}$

$$\begin{aligned}\mathcal{L}(\mathbf{w})&=\log\prod_{i=1}^{N}p(y_i|\mathbf{x}_i,\mathbf{w})\\&=\sum_{i=1}^{N}\log p(y_i|\mathbf{x}_i,\mathbf{w})\\&=\sum_{i=1}^{N}(\log\frac{1}{\sqrt{2\pi}\sigma}-\frac{(y_i-\mathbf{w}^T\mathbf{x}_i)^2}{2\sigma^2})\end{aligned}$$

$$\begin{aligned}\hat{\mathbf{w}}&=\arg\max_{\mathbf{w}}\mathcal{L}(\mathbf{w})\\&\Rightarrow\arg\min_{\mathbf{w}}\sum_{i=1}^{N}(y_i-\mathbf{w}^T\mathbf{x}_i)^2\end{aligned}$$

与式 (7) 等价, 因此可以得出, 最小二乘估计隐含了一个噪声服从正态分布的假设.

3 正则化 (岭回归)

由式 (9) 结果 $\hat{\mathbf{w}}=(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$ 可知, 计算最优参数需要对 $\mathbf{X}^T\mathbf{X}$ 求逆, 但是若 $N\gg p$ 时, 在数学上矩阵不可逆, 无法计算最优参数的解析解. 在机器学习中, 称为过拟合 (overfitting).

防止过拟合的方法主要有三种 1) 增加数据; 2) 降维 (特征选择/特征提取) 3) 正则化, 下面主要讨论线性回归中的正则化方法.

正则化的主要框架如下, 在原有的损失函数的基础上加上一个惩罚项 (penalty) 进行约束

$$\mathcal{L}(\mathbf{w})+\lambda P(\mathbf{w}) \tag{14}$$

从而将式 (7) 变为

$$\arg\min_{\mathbf{w}}\mathcal{L}(\mathbf{w})+\lambda P(\mathbf{w}) \tag{15}$$

线性回归中通常采用 L_1 正则 (lasso) 和 L_2 正则 (Ridge), 也称为岭回归, 主要讨论岭回归, 其

损失函数的形式如下, 它也有个特殊的名字权重衰减

$$\mathcal{J}(\mathbf{w}) = \sum_{i=1}^{N} ||\mathbf{w}^T\mathbf{x}_i - y_i||^2 + \lambda\mathbf{w}^T\mathbf{w} \tag{16}$$

为什么加入正则项可以防止过拟合, 下面从解析和概率的角度进行解释.

3.1 解析解角度分析

将式 (16) 写为矩阵形式

$$\begin{aligned}\mathcal{J}(\mathbf{w}) &= \sum_{i=1}^{N} ||\mathbf{w}^T\mathbf{x}_i - y_i||^2 + \lambda\mathbf{w}^T\mathbf{w} \\ &= \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{w}^T\mathbf{X}^T\mathbf{Y} + \mathbf{Y}^T\mathbf{Y} + \lambda\mathbf{w}^T\mathbf{w} \\ &= \mathbf{w}^T(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\mathbf{w} - 2\mathbf{w}^T\mathbf{X}^T\mathbf{Y} + \mathbf{Y}^T\mathbf{Y}\end{aligned}$$

对上式进行求导

$$\frac{\partial}{\partial\mathbf{w}}\mathcal{J}(\mathbf{w}) = 2(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\mathbf{w} - 2\mathbf{X}^T\mathbf{Y}$$

令上式为 0 得

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y} \tag{17}$$

从数学上看 $\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$ 一定可逆 (因为 $\mathbf{X}^T\mathbf{X}$ 是半正定矩阵, 加上一个对角矩阵后一定是正定的, 也就可逆), 因此可以求出最优解.

3.2 概率角度分析

本节从贝叶斯学派的角度来看岭回归, 使用 MAP(最大后验概率估计) 来计算最优参数.

注. MAP 为贝叶斯学派常用的参数估计方法, 认为模型参数服从某种潜在分布. 其首先对参数有一个预先估计, 然后根据所给数据对预估计进行不断调整, 因此同一事件, 先验不同则事件状态不同. 先验假设较为靠谱时有显著的效果, 当数据较少时, 先验对模型的参数有主导作用, 随着数据的增加，真实数据样例将占据主导地位

预先得到的数据 $y = \mathbf{w}^T\mathbf{x} + \epsilon$, 从 MAP 的角度来看, 参数必然服从某个分布, 故假设 $w_i \sim \mathcal{N}(0, \sigma_0^2)(i = 1, 2, \cdots, p)$

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} p(\mathbf{w}|\mathbf{Y}) \tag{18}$$

根据贝叶斯公式有

$$p(\mathbf{w}|\mathbf{Y}) = p(w_1, w_2, \cdots, w_p|y_1, y_2, \cdots, y_N) = \frac{p(\mathbf{w})p(\mathbf{Y}|\mathbf{w})}{p(\mathbf{Y})} \tag{19}$$

$$
\begin{aligned}
\hat{\mathbf{w}} &= \arg\max_{\mathbf{w}} \log p(\mathbf{w}|\mathbf{Y}) \\
&\Rightarrow \arg\max_{\mathbf{w}} \log p(\mathbf{w})p(\mathbf{Y}|\mathbf{w}) \\
&= \arg\max_{\mathbf{w}} \log p(w_1, w_2, \cdots, w_p)p(y_1, y_2, \cdots, y_N|w_1, w_2, \cdots, w_p) \\
&= \arg\max_{\mathbf{w}} \log p(w_1)p(w_2)\cdots p(w_p)p(y_1|\mathbf{w})p(y_2|\mathbf{w})\cdots p(y_N|\mathbf{w}) \\
&= \arg\max_{\mathbf{w}} \log \prod_{j=1}^{p} p(w_j) \prod_{i=1}^{N} p(y_i|\mathbf{w})
\end{aligned} \tag{20}
$$

又因为已知 $p(w)$ 和 $p(y_i|\mathbf{w})$ 的分布

$$
\begin{aligned}
p(y_i|\mathbf{w}) &= \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(y_i - \mathbf{w}^T\mathbf{x}_i)^2}{2\sigma^2}) \\
p(w_j) &= \frac{1}{\sqrt{2\pi}\sigma_0} \exp(-\frac{||w_j||^2}{2\sigma_0^2})
\end{aligned}
$$

将其带入式 (20) 得

$$
\begin{aligned}
\hat{\mathbf{w}} &= \arg\max_{\mathbf{w}}(\sum_{i=1}^{N} \log p(y_i|\mathbf{w}) + \sum_{j=1}^{p} \log p(w_j)) \\
&= \arg\max_{\mathbf{w}}(\sum_{i=1}^{N} \log \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(y_i - \mathbf{w}^T\mathbf{x}_i)^2}{2\sigma^2}) + \sum_{j=1}^{p} \log \frac{1}{\sqrt{2\pi}\sigma_0} \exp(-\frac{||w_j||^2}{2\sigma_0^2})) \\
&= \arg\max_{\mathbf{w}}(N \log \frac{1}{\sqrt{2\pi}\sigma} + p \log \frac{1}{\sqrt{2\pi}\sigma_0} - \sum_{i=1}^{N} \frac{(y_i - \mathbf{w}^T\mathbf{x}_i)^2}{2\sigma^2} - \sum_{j=1}^{p} \frac{||w_j||^2}{2\sigma_0^2}) \\
&\Rightarrow \arg\min_{\mathbf{w}} \sum_{i=1}^{N} \frac{(y_i - \mathbf{w}^T\mathbf{x}_i)^2}{2\sigma^2} + \sum_{j=1}^{p} \frac{||w_j||^2}{2\sigma_0^2} \\
&\Rightarrow \arg\min_{\mathbf{w}} \sum_{i=1}^{N} (y_i - \mathbf{w}^T\mathbf{x}_i)^2 + \sum_{j=1}^{p} \frac{\sigma^2}{\sigma_0^2} ||w_j||^2 \\
&\Rightarrow \arg\min_{\mathbf{w}} \sum_{i=1}^{N} (y_i - \mathbf{w}^T\mathbf{x}_i)^2 + \frac{\sigma^2}{\sigma_0^2} ||\mathbf{w}||^2
\end{aligned} \tag{21}
$$

令 $\frac{\sigma^2}{\sigma_0^2}$ 为 λ, 则上式子与式 (16) 等价, 即 $\arg\min_{\mathbf{w}} \sum_{i=1}^{N} (y_i - \mathbf{w}^T\mathbf{x}_i)^2 + \lambda \mathbf{w}^T\mathbf{w}$.

由此可以得出如下结论: 最小二乘法估计等价于噪声服从正态分布的极大似然估计, 而加入了正则项的最小二乘估计与包含服从高斯分布的噪声和先验的 MAP 是等价的.

4 代码实现

代码实现使用更加一般的方法求解最优参数, 即梯度下降法, 一般损失函数改写为

$$
\mathcal{L}(\mathbf{w}) = \frac{1}{2N} \sum_{i=1}^{N} (\mathbf{w}^T\mathbf{x}_i - y_i)^2 \tag{22}
$$

那么对于特定的参数 w_i 的跟新规则如下

$$
w_i := w_i - \eta \frac{\partial}{\partial w_i} \mathcal{L}(\mathbf{w}) \tag{23}
$$

又

$$\frac{\partial}{\partial w_j}\mathcal{L}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}_i - y_i)x_{ij} \tag{24}$$

基于 numpy 的代码实现如下

```
import numpy as np
import random

def synthetic_data(w, b, num_examples):
    "生成y = wx + b + \epsilon的数据集"
    X = np.random.normal(0, 1, size = (num_examples, w.shape[0]))
    y = np.matmul(X, w) + b
    y += np.random.normal(0, 0.01, size = y.shape)
    return X, y.reshape(-1, 1)

def loss_total(X, Y, w, b):
    "均方损失"
    totalLoss = 0
    for i in range(0, Y.shape[0]):
        totalLoss += (Y[i] - (np.matmul(X[i], w)+b)) ** 2 / 2
    return float(totalLoss / Y.shape[0])

def SGD(w, b, lr, X, Y):
    "小批量梯度下降"
    b_grad = 0
    w_grad = 0
    N = float(Y.shape[0])
    for i in range(0, Y.shape[0]):
        x = X[i]
        y = Y[i]
        b_grad += (1 / N) * (np.matmul(x, w) + b - y)
        w_grad += (1 / N) * (np.matmul(x, w) + b - y) * x
    new_b = b - lr * b_grad
    new_w = w - lr * np.reshape(w_grad, w.shape)
    return new_b, new_w

def data_iter(batch_szie, X, Y):
    "批量读取数据"
    num_examples = Y.shape[0]
    indices = list(range(num_examples))
    random.shuffle(indices)
    for i in range(0, num_examples, batch_szie):
        batch_indices = indices[i : min(i + batch_szie, num_examples)]
        yield X[batch_indices], Y[batch_indices]

def main():
    # 生成数据集
```

```
    true_w = np.array([[2], [-3.4]])
    true_b = 4.2
    num_examples = 1000
    X, Y = synthetic_data(true_w, true_b, num_examples)

    # 初始化模型参数
    w = np.random.normal(0, 0.01, size = (X.shape[1], Y.shape[1]))
    b = np.zeros(Y.shape[1])

    # 开始训练
    lr = 0.03
    num_epochs = 10
    batch_size = 10
    for epoch in range(num_epochs):
        for step, (X_b, Y_b) in enumerate(data_iter(batch_size, X, Y)):
            loss = loss_total(X, Y, w, b)
            b, w = SGD(w, b, lr, X_b, Y_b)
            if (step + 1) % 20 == 0:
                print(epoch + 1, step + 1, 'loss', float(loss))

    print('线性回归估计值w', w.reshape(1,-1), 'b', b)

if __name__ == '__main__':
    main()
```