机器学习-第四讲课后作业

姓名：周宝航 学号：2120190442 专业：计算机科学与技术

一、问题描述

1. 考虑最小二乘法的公式推导中，令损失函数关于参数的两个偏导数为零获得了最优参数值的必要条件，它们意味着预测误差是零均值且与输入不相关。试说明“零均值”和“不相关”的根据是什么？

2. 线性回归问题：假设存在直线方程 $y = 1.4x + 0.9$，噪声服从 $N(0,5)$

1) 生成满足条件的 100 个独立同分布样本，选择其中 80%作为训练样本，剩余为测试样本。

2) 绘出训练样本和测试样本的散布图。

3) 编程实现求解线性回归的最小二乘法和采用 Huber 损失函数的鲁棒线性回归算法。

4) 使用 3）中实现的线性回归算法求解训练样本的线性回归方程，并给出测试样本的均方差。

5) 解释 4）中所得到的结果，并对两种求解方法进行比较。

二、基本思路

1. 在考虑最小二乘法公式的最优参数求解问题时，我们令损失函数对参数求偏导数，并令它们为零。对此，我们依据高斯-马尔可夫定理以及无偏估计对该问题进行讨论。

2. 题中给定回归直线方程和噪声分布。

1） 我们假定样本特征服从均匀分布，并采样得到一定数量的样本。然后，利用直线方程和采样得到的噪声值便得到样本的观测值。最后，利用 Matplotlib 库函数进行散点图的绘制。

2） 对于最小二乘法，我们利用推导得出的正规方程对参数进行求解。而面对采用 Huber 损失函数的鲁棒线性回归算法，我们通过推导得到损失函数对参数的偏导数，并利用梯度下降法进行参数求解。

3） 利用 2）中得到的回归算法模型，我们利用训练样本对参数进行求解。待求解完成后，我们将模型应用在测试集并以均方误差作为评价指标。

三、解题步骤

问题 1：

根据高斯马尔可夫定理，我们可以知道：在线性回归模型中，如果预测误差满足零均值、同方差且互不相关，则回归系数的最佳线性无偏估计就是普通最小二乘法估计。所以，在噪声模型中，我们规定误差符合某种高斯分布。我们在根据损失函数推导对参数的偏导数时必须令它们为零，即可得到对参数的最优估计。

问题 2：

1. 生成数据

我们假设样本特征服从均匀分布，从其中采样 100 个数据点 x_i 作为样本。除此之外，我们从噪声分布 $\mathrm{N}(0,5)$ 采用 100 个数据点 ε_i 作为噪声数据。然后，根据直线方程 $\mathrm{y_i} = 1.4x_i + 0.9 + \varepsilon_i$ 便可以得到生成样本

的目标值。代码如下：

```
# 生成数据
data_size = 100
x = np.random.rand(data_size).reshape((-1, 1))
noise = np.random.normal(0, np.sqrt(5), size=data_size).reshape((-1, 1))
y = 1.4 * x + 0.9 + noise
```

为了直观展示生成的数据，我们使用 Matplotlib 库函数绘制出数据分布的散点图。代码如下：

```
def plot_data(x, y):
    '''''
        生成数据图
    '''
    plt.scatter(x, y)
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.title('Data')
    plt.show()
```

在数据散点图中，我们可以看到：由于噪声的存在，样本数据中存在一些离群点，会对回归模型的建立造成一些影响。

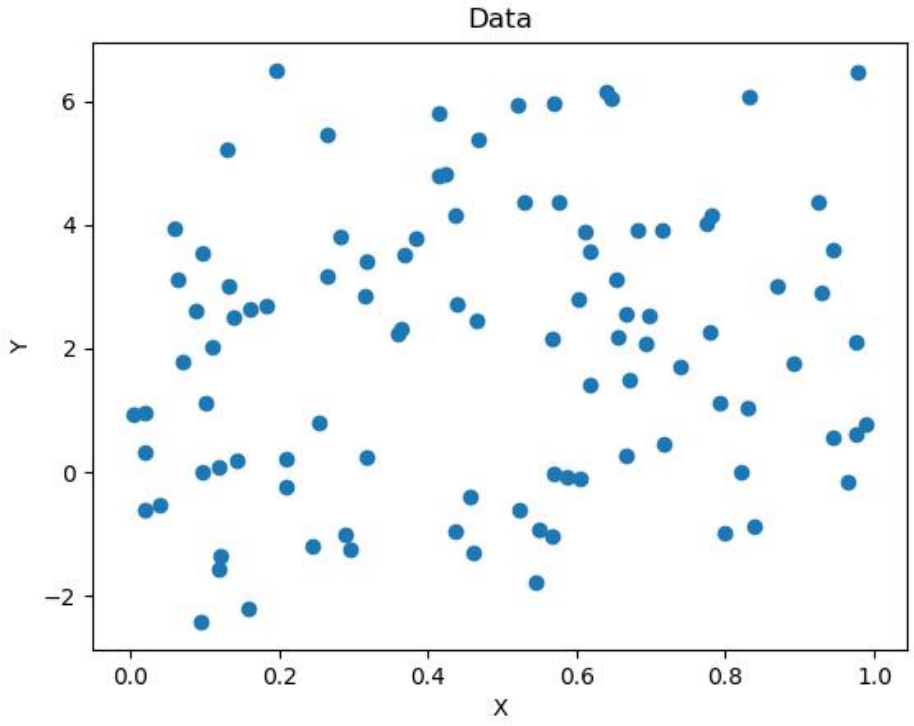

图 1 数据分布散点图

2. 算法描述

2.1 最小二乘法

我们假设有：样本数据矩阵 $\mathrm{X}\in\Re^{m\times(n+1)}$，其中 m 为样本数量、n 为样本特征数量。对应所有样本数据，我们还有需要进行回归预测的目标值向量 $\mathrm{y}\in\Re^{m\times1}$。而线性回归的模型参数为：$\mathrm{w}\in\Re^{(n+1)\times1}$，由此我们可以定义最小二乘法的损失函数：$\frac{1}{2}\|\mathrm{y}-\mathrm{Xw}\|^2$。

令上面损失函数的导数为零，可以得到正规方程：$\mathrm{X^T y - X^T X w = 0}$，移项变为：$\mathrm{w=(X^TX)^{-1}X^Ty}$。如此一来，我们便得到线性回归模型的最优参数。

2.2 基于 Huber 损失函数的鲁棒线性回归法

上面提到的最小二乘法求解线性回归问题时，其假设噪声模型为零均值高斯分布。然而，这种模型对含有离群点的数据较为敏感，会对数据的拟合较差。因此，我们引入 Huber 函数作为求解鲁棒线性回归模型的损失函数，其定义为：

$$\mathrm{L_{Huber}}(r,\delta)=\begin{cases}\dfrac{r^2}{2} & if\,|r|\le\delta\\ \delta|r|-\dfrac{\delta^2}{2} & otherwise\end{cases}。$$

我们通过引入一个关于每个样本的损失权重矩阵 $\mathrm{W}\in\Re^{m\times m}$ 来进行参数估计，其中 $\mathrm{W}=\mathrm{diag}(\omega_1,\omega_2,...,\omega_m)$。通过简化求解损失函数，我们可以得到关于每个样本的损失权重值：$\omega_{\mathrm{i}}=\begin{cases}1 & if\,|r_i|\le\delta\\ \dfrac{\delta}{|r_i|} & otherwise\end{cases}$。如此一来，关于参数的求解公式为：$\mathrm{w=(X^TWX)^{-1}X^TWy}$。最后，我们通过迭代求解对最优参数的估计值。

3. 算法实现

3.2 最小二乘法

```
1.  class LeastSquareMethod(object):
2.      '''''
3.          最小二乘法求解回归问题
4.      '''
5.      def __init__(self):
6.          self.params = None
7.      def fit(self, data, verbose=True):
8.          # 样本维度
9.          x = data[:, :-1].reshape((-1, 1))
10.         # 预测值维度
11.         y = data[:, -1].reshape((-1, 1))
12.         # 加入常数项
13.         c_x = np.hstack((np.ones_like(x), x))
14.         # 求解正规方程
15.         w = np.linalg.inv(c_x.T @ c_x) @ c_x.T @ y
16.         self.params = w
17.         if verbose:
18.             x_min = np.min(x) - 1.
19.             pred_y = c_x @ self.params
20.             plt.plot(x, pred_y, '-', label='least squares')
21.     def predict(self, data):
22.         # 样本维度
23.         x = data[:, :-1].reshape((-1, 1))
24.         # 预测值维度
25.         y = data[:, -1].reshape((-1, 1))
26.         # 加入常数项
27.         c_x = np.hstack((np.ones_like(x), x))
28.         # 预测结果
29.         pred = c_x @ self.params
30.         return pred
```

上面是我们对最小二乘算法的实现类，其中第 15 行是利用正规方程对最优参数进行求解。

3.2 基于 Huber 损失函数的鲁棒线性回归法

```
1.  class RobustnessLinearRegression(object):
2.      '''''
3.          鲁棒线性回归算法
4.          采用 Huber 损失函数进行训练
```

```
5.      '''
6.      def __init__(self, sigma=1., threshold=1e-3):
7.          # 方差参数
8.          self.sigma = sigma
9.          # 迭代误差阈值
10.         self.threshold = threshold
11.     def fit(self, data, verbose=True):
12.         # 样本维度
13.         x = data[:, :-1].reshape((-1, 1))
14.         # 预测值维度
15.         y = data[:, -1].reshape((-1, 1))
16.         # 加入常数项
17.         c_x = np.hstack((np.ones_like(x), x))
18.         w = np.random.normal(size=(c_x.shape[1], 1))
19.         threshold = 100.
20.         while threshold >= self.threshold:
21.             # 求得损失
22.             r = np.abs(c_x @ w - y)
23.             W = np.ones((c_x.shape[0], 1))
24.             W[r > self.sigma] = self.sigma / r[r > self.sigma]
25.             # 转换为对角矩阵
26.             W = np.eye(W.shape[0]) * W
27.             # 求解参数
28.             new_w = np.linalg.inv(c_x.T @ W @ c_x) @ c_x.T @ W @ y
29.             # 参数是否收敛
30.             threshold = np.sqrt(np.sum(np.square(w - new_w)))
31.             w = new_w
32.         self.params = w
33.         if verbose:
34.             pred_y = c_x @ self.params
35.             plt.plot(x, pred_y, '-', label='Huber')
36.     def predict(self, data):
37.         # 样本维度
38.         x = data[:, :-1].reshape((-1, 1))
39.         # 预测值维度
40.         y = data[:, -1].reshape((-1, 1))
41.         # 加入常数项
42.         c_x = np.hstack((np.ones_like(x), x))
43.         # 预测结果
44.         pred = c_x @ self.params
45.         return pred
```

我们定义了鲁棒线性回归求解类，其中第 20-31 行是迭代求解参数的主要流程。该方法还涉及一个参数δ的选择问题，我们在实验中

采用余一交叉验证的方法进行选择。代码如下：

```
def cross_validation(model, data, params=[]):
    '''''
        余一交叉验证
    '''
    result = {}
    for param in params:
        model.sigma = param
        errors = []
        for i in range(len(data)):
            tmp = data.tolist()
            # 余一测试样本
            test_data = np.array(tmp[i]).reshape((1, -1))
            del tmp[i]
            # 训练样本
            train_data = np.array(tmp)
            # 模型训练
            model.fit(train_data, verbose=False)
            # 模型预测
            pred = model.predict(test_data)
            # 预测误差
            errors.append(np.mean(np.square(pred.flatten() - test_data[:, -1]
))))
        # 每个超参数平均误差
        result[param] = np.mean(errors)
    min_error = 100.
    best_param = 0.
    for k, v in result.items():
        print(f"参数:{k} 误差:{v}")
        if v < min_error:
            best_param = k
            min_error = v
    return best_param
```

四、结果与分析

1. 实验内容与步骤

针对最小二乘算法，我们需要按照正规方程求解参数后，将回归模型用于测试集样本，并采用均方误差作为评价指标衡量模型的优

劣。而基于 Huber 损失的鲁棒线性回归算法涉及参数δ的选择。因此，我们需要设置一组待选参数，并利用余一交叉验证法进行最优δ值的选择。然后，固定最优δ值后，将全部训练集用于回归模型的训练。最后，同样将模型应用于测试集，并采用均方误差作为评价指标衡量模型的优劣。均方误差定义如下：

$$\mathrm{MSE}=\frac{1}{m}\sum_{i=1}^{m}(y_i-\hat{y}_i)^2 。$$

2. 实验结果

关于鲁棒线性回归方法涉及的参数δ选择问题，我们采用余一交叉验证方法进行实验，得到结果如下：

δ取值	1.5	2.5	2.6	2.7	2.8
均方误差	5.428	5.372	5378	5.383	5.379

表 1 参数δ交叉验证结果

根据上面的交叉验证结果，我们在δ取值为：2.5 时，取得了交叉验证的最优结果。然后，我们固定该值重新训练鲁棒线性模型并与最小二乘法进行性能比较。

方法	最小二乘法	Huber 鲁棒线性回归
均方误差	4.500	4.439

表 2 测试集预测评价

上表是对最小二乘法与基于 Huber 损失函数的鲁棒线性回归算法性能评价的比较结果。我们可以看到：在采用相同训练集与测试集的情况下，后者比前者的预测误差更小。这表明在对含有离群点数据建模时，Huber 鲁棒线性回归算法更具有优越性。

3. 结果分析

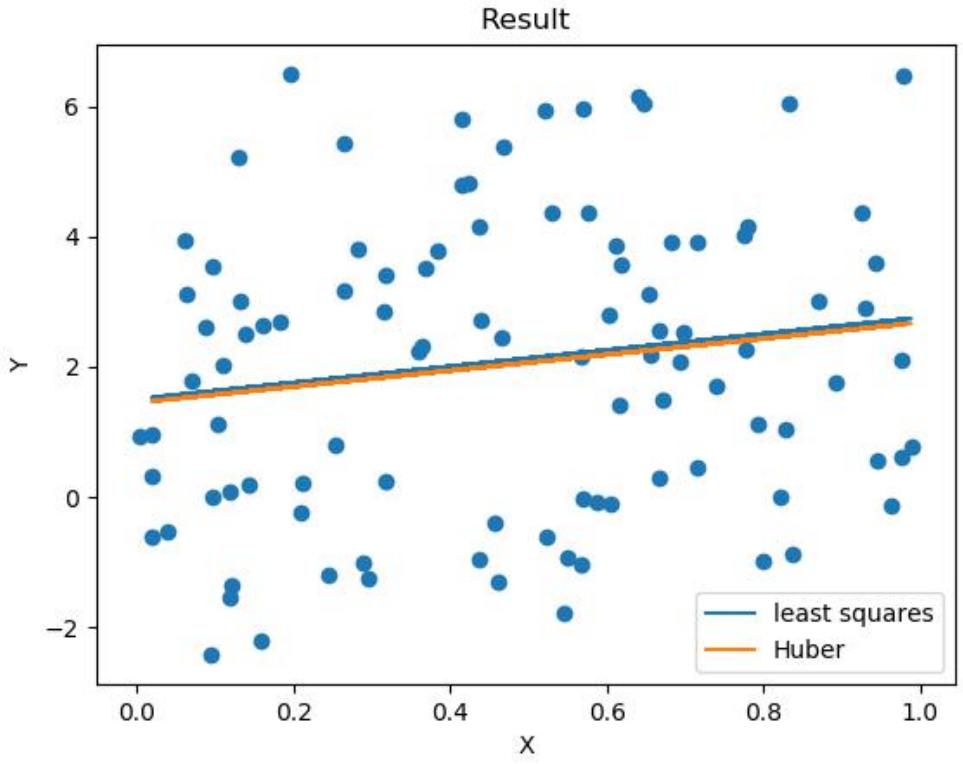

图 2 回归结果图

表 2 展示了最小二乘法和基于 Huber 损失函数的鲁棒线性回归算法性能评价的比较结果。为了更直观的看到两种方法的结果差异，我们在图 2 绘制了它们各自的回归直线。由于存在离群点，最小二乘法得到的回归直线更靠近上方的离群点。而引入了 Huber 损失函数的方法，其依靠参数δ来减小离群点对回归模型的影响。这点反映在数值结果上，我们看到：相同的训练数据包含了离群点，经过模型训练，最小二乘法的测试结果比 Huber 鲁棒线性回归方法要差，前者比后者的泛化能力要差。