2020机器学习

实验报告2

姓名：刘言政

学号：20307130167

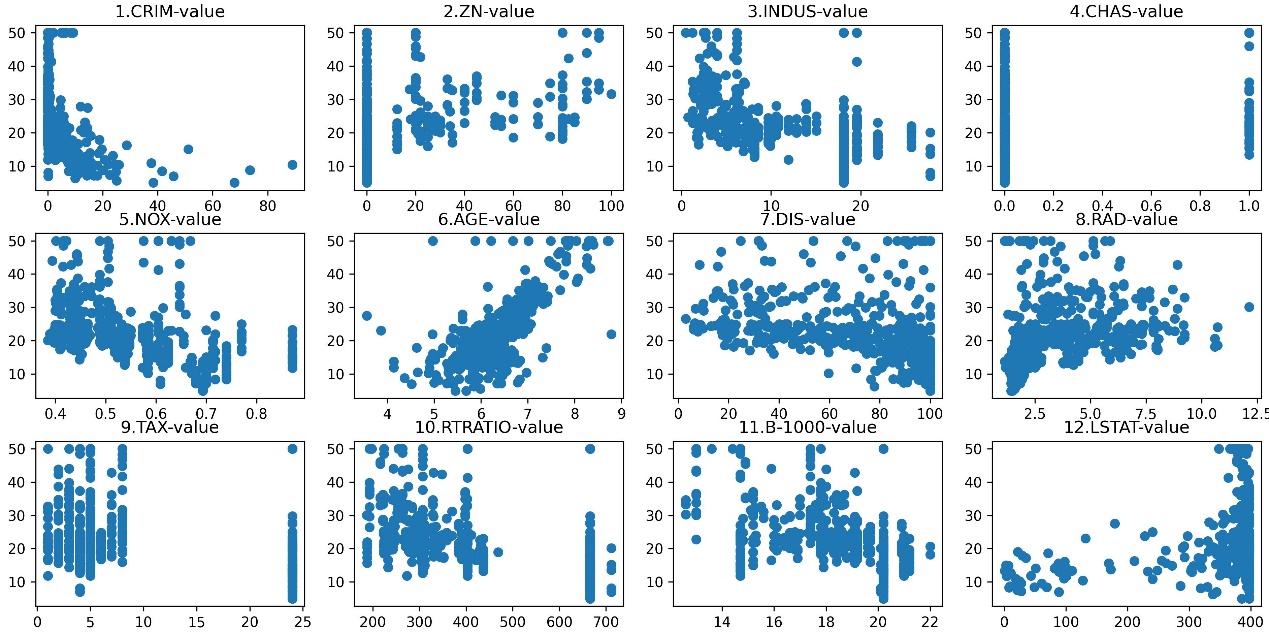
1. 任务描述
2. 手写以下三种线性回归模型：
3. 普通的标准线性回归模型
4. Ridge回归（岭回归）模型
5. Lasso回归模型
6. 在Boston房价数据集上测试模型的回归效果。
7. 以可视化的方式，与python现有库的回归模型分析比较。
8. 数据描述
9. 来源：<https://www.kaggle.com/vikrishnan/boston-house-price>

也可以直接通过sklearn.dataset.load\_boston()导入。

1. 简介数据集共包含506条数据，每条数据可以看做一个14×1的向量。前13个数值表示房子相关的13个属性，即x；最后一个为房价，即y。x包含13个属性：CRIM、ZN、INDUS、CHAS、NOX、AGE、DIS、RAD、TAX、RTRATIO、B-1000、LSTAT、MEDV，含义可参照上述链接。我们的模型目标即，根据这13个属性，预测y的值
2. 数据集预处理
3. 数据集可视化

该部分只是针对数据集的可视化处理，用于直观感受x的每个属性和y的相关性。与后面的算法并无直接关联。

功能通过pre\_vsl.py下的函数实现。也可直接运行该文件。



1. 载入并划分数据集

数据集的载入在pre\_work.py文件下实现。

def load\_dataset():

    # 获取数据集

    # x表示房屋属性，共13项，y表示房价

    train\_x, train\_y = datasets.load\_boston(return\_X\_y = True)

    return get\_array(train\_x,train\_y)

该函数实现数据集的导入，同时会将数组转换为numpy.array的形式。

def load\_trainset():

    X, Y = load\_dataset()

    return X[0:450], Y[0:450]

def load\_testset():

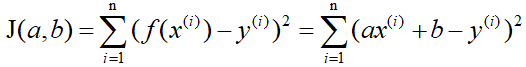
    X, Y = load\_dataset()

    return X[451:506], Y[451:506]

此处两个函数分别载入训练集和测试集，实现了原数据集的划分。

1. 算法介绍
2. Linear回归

普通的线性回归方法。通过最小化损失函数：



来实现线性回归。由于该问题已存在数学上的最优解，即：



所以线性回归的实现相较简单，直接代入公式即可。

class Linear:

    def \_\_init\_\_(self):

        pass

    def fit(self, x, y):

        m = x.shape[0]

        self.x = np.concatenate((np.ones((m,1)),x),axis=1)

        self.y = copy.copy(y)

        self.w = np.linalg.inv(np.transpose(self.x).dot(self.x)).dot(np.transpose(self.x)).dot(self.y)

    def predict(self, x):

        m = x.shape[0]

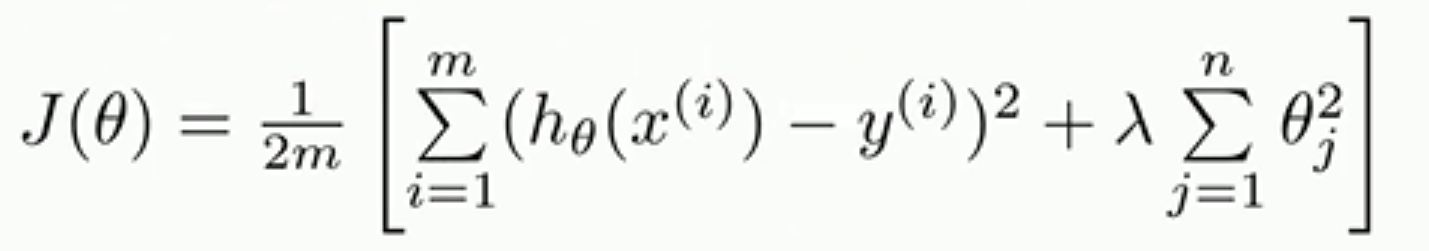
        X = np.concatenate((np.ones((m,1)),x),axis=1)

        y = X.dot(self.w)

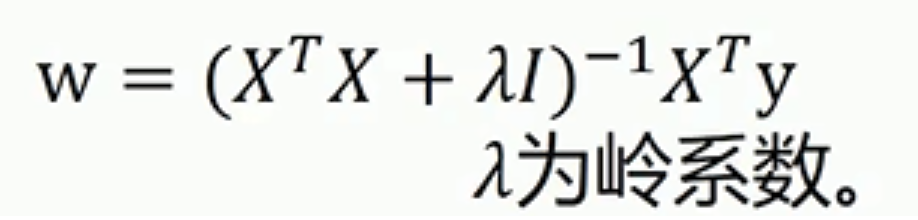
        return y

1. Ridge回归

线性回归在数据集较小的情况下，尤其是数据集不足以概括全部数据特征的情况下，由于矩阵不满秩，存在较明显的缺陷。因此，ridge回归改善了损失函数：



与普通的线性回归一样，该回归方程也存在数学上的最优解，公式可直接给出：



具体的实现如下：

class Ridge:

    def \_\_init\_\_(self):

        pass

    def fit(self, x, y, lmd = 0.2):

        m = x.shape[0]

        self.x = np.concatenate((np.ones((m,1)),x),axis=1)

        self.y = copy.copy(y)

        idty =  np.identity(len(self.x[0]))

        self.w = np.linalg.inv(np.transpose(self.x).dot(self.x)-lmd\*idty).dot(np.transpose(self.x)).dot(self.y)

    def predict(self, x):

        m = x.shape[0]

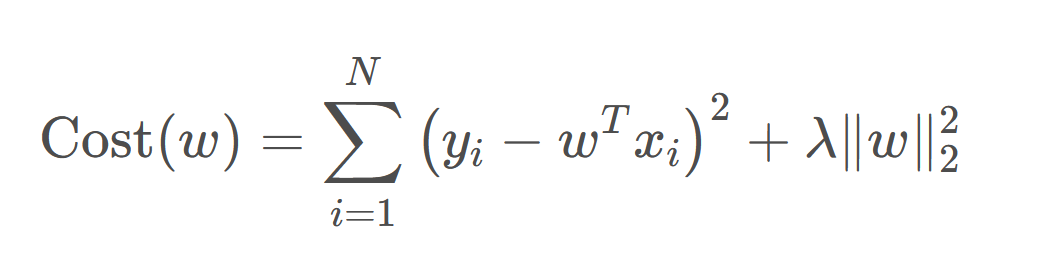
        X = np.concatenate((np.ones((m,1)),x),axis=1)

        y = X.dot(self.w)

        return y

1. Lasso回归

Lasso回归是改善普通线性回归的缺陷的另一种方式，其损失函数如下：



与前两者不同的是，lasso回归的损失函数并非处处可导，这也就意味着不能通过前两者相似的方法，直接求出数学上的最优解。所以我们使用梯度下降法来逼近最优解：

class Lasso:

    def \_\_init\_\_(self):

        pass

    def fit(self, x, y, lmd = 0.2, learning\_rate = 0.000005, epochs = 50000):

        m = x.shape[0]

        self.x = np.concatenate((np.ones((m,1)),x),axis=1)

        self.y = copy.copy(y)

        xMAT = np.mat(self.x)

        yMAT = np.mat(self.y.reshape(-1,1))

        self.w = np.ones(self.x.shape[1]).reshape(-1,1)

        for i in range(epochs):

            gradient = xMAT.T \* (xMAT\*self.w - yMAT)/m + lmd\*np.sign(self.w)

            self.w = self.w -  learning\_rate\*gradient

    def predict(self, x):

        m = x.shape[0]

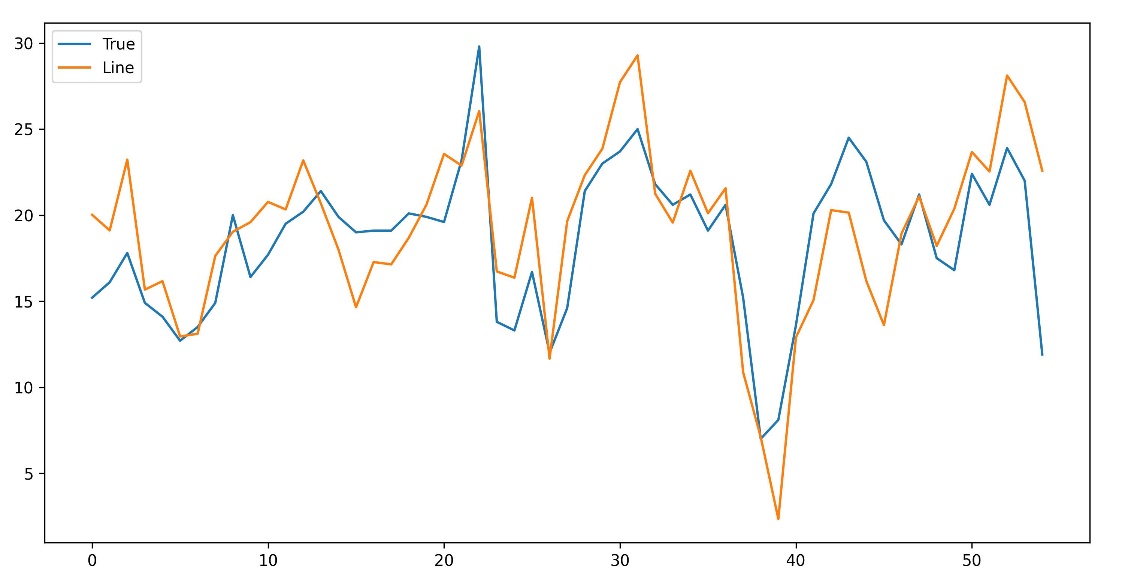
        X = np.concatenate((np.ones((m,1)),x),axis=1)

        y = X.dot(self.w)

        return y

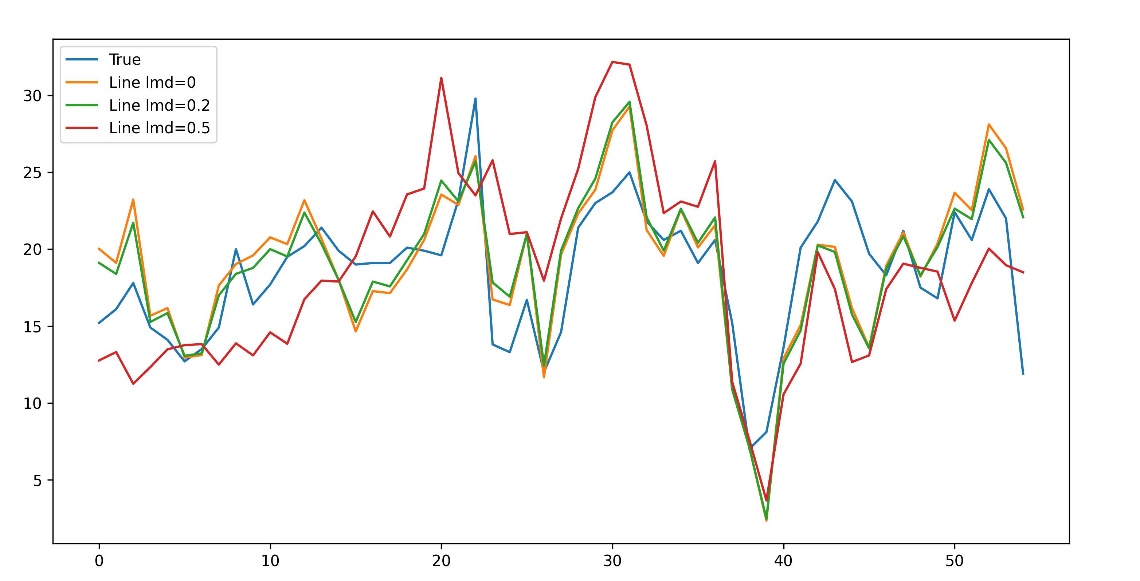
此处，有三个参数可以调整。实验过程中发现，学习率过高时，回归模型无法正常工作，数组中会出现Nan（not a number）的情况。当学习率小于0.00001时，在Boston数据集上才能够正常工作。所以此处选择了默认学习率为0.000005。关于lmd的分析，将在后续部分提到。

1. 实验结果及分析
2. 回归结果可视化
3. 线性回归



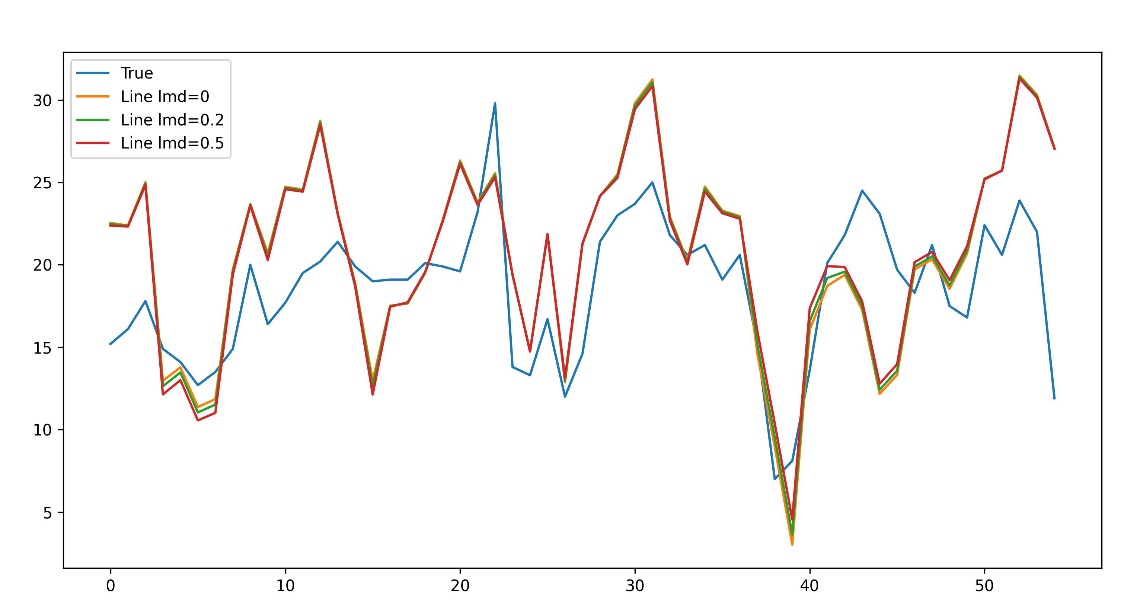
回归结果较好，符合真实数据集的变化趋势，与真实值接近。

1. Ridge回归



整体上回归结果较好。对比不同的lmd（λ）参数，当lmd取值为0.2时，回归结果的score最高。故模型中lmd参数默认值设置为0.2。

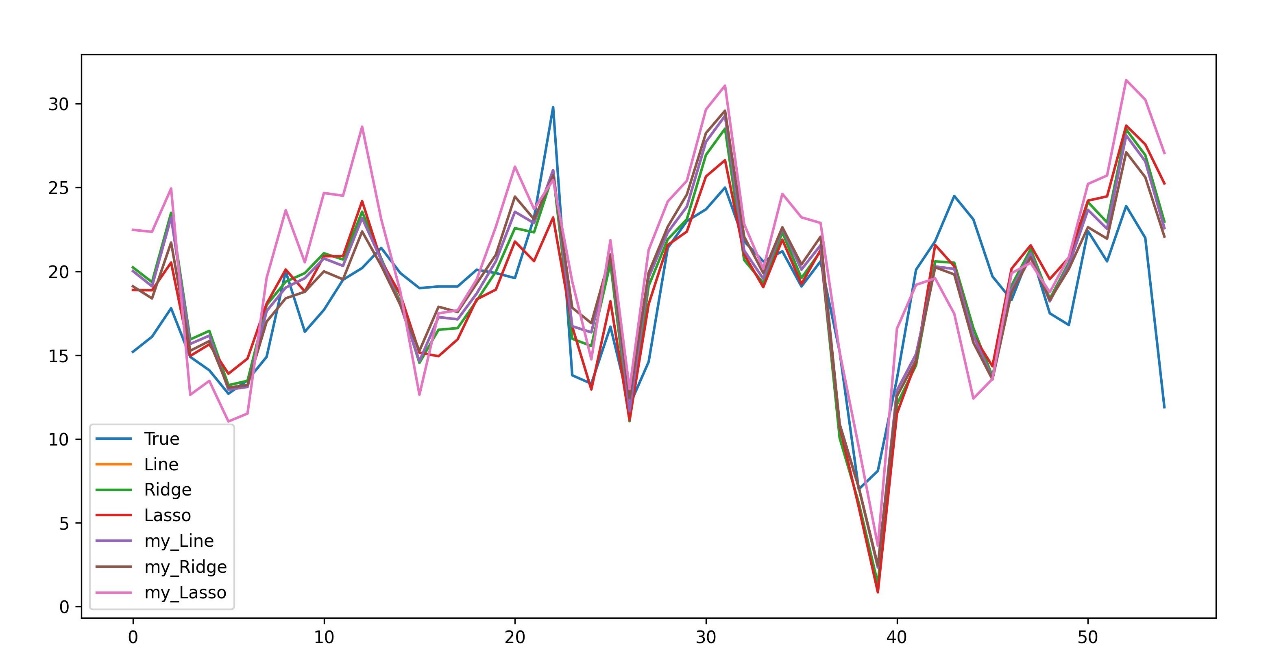
1. Lasso回归



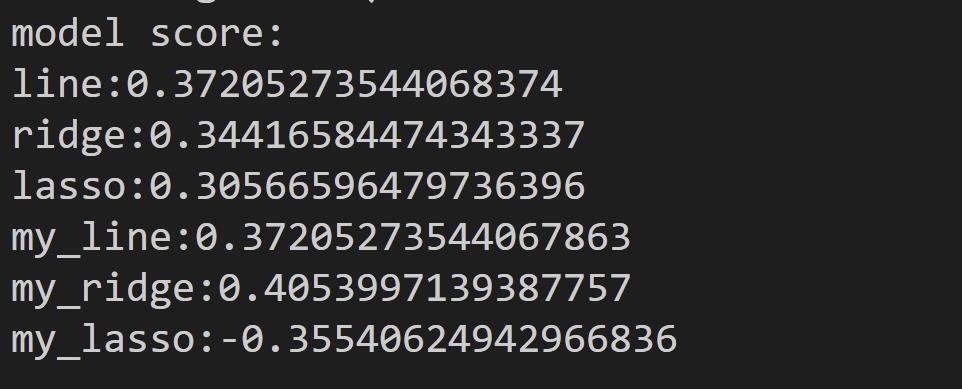
回归结果与前两者相比略逊一筹。一部分原因在于该方法使用了梯度下降的方法，与前两者有本质上的差异，且在参数的调整上还有较大的提升空间。Lmd的取值对回归结果的影响并不大。模型中默认值取0.2。

1. 与python已有函数库的模型对比

相关代码在test.py文件下。



直观上来看，6个回归模型的优劣并不明显。所以我们求助于sklearn的模型打分机制：



可以看到：

1. 普通的线性回归模型，官方库和自写的模型差异极小，可以忽略不计。
2. 自写的ridge回归模型在Boston数据集上的回归效果最优，远超其他模型。
3. 自写的lasso回归模型的回归效果最差，远低于其他模型。
4. 总结
5. 线性回归模型是一种常用的简单机器学习算法，在线性相关性较高的场景下具有很好的回归效果。但在数据集较小的情况下不可行。
6. Ridge回归和lasso回归都是对普通线性回归缺陷做出改进的回归模型。其中ridge回归存在理论上的最优解，但lasso回归由于损失函数不可导的原因，只能使用梯度下降法等方法逼近最优解。
7. 三种回归模型在Boston房价数据集上均具有良好的表现。