模式识别与统计学习实验一：Linear Regression实验报告

郭宇航 2016100104014

一　实验原理

1.1　单变量线性回归

以本次实验的波士顿房价数据集作为实例进行实验原理的阐述，考虑一维的线性回归的问题，假设影响房价的预测变量为单变量x，响应变量为房价h_0，则，我们可以尝试构建简单的线性回归模型：

$$h_\theta = \theta_0 + \theta_1 x \tag{1}$$

其中θ_0为常数截距，θ_1为预测变量的常系数。在构建起简单的线性模型后，假设数据集中有m个样本，通过这些样本对θ_0与θ_1进行参数估计。

假设通过线性模型预测得到m个样本中第i个样本的预测变量$x^{(i)}$对应的预测结果为$h_0(x^{(i)})$，而数据集中$x^{(i)}$实际对应的响应变量结果为$y^{(i)}$，我们不难得到一个通过线性回归预测模型得到的损失函数(Loss Function)，这里定义为J：

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

其中$1/2m$作为一个常系数，不影响损失函数J的变化趋势。此系数服务于之后的优化求解极值问题。显然，这里将回归方程系数的求解问题转化为了一个最优化问题，若想要得到最优的线性回归模型，则我们需要最小化损失函数J，下面考虑如何求解损失函数的最小值，我们引入梯度下降法。

1.2　梯度下降法

如何求解损失函数J的极小值。我们考虑使用梯度下降法进行求解。在微积分（下）的多元函数求偏导问题中我们涉及过梯度(Gradient)，其定义如下：设函数$f(x,y)$的定义域为D，f在点$P(x,y) \in D$可微，则称向量$\frac{\partial f}{\partial x}\vec{i} + \frac{\partial f}{\partial y}\vec{j}$为$f(x,y)$在点$P(x,y)$的梯度，记为$\nabla f$：

$$\nabla f(x,y) = \frac{\partial f}{\partial x}\vec{i} + \frac{\partial f}{\partial y}\vec{j} \tag{3}$$

沿着梯度的方向，方向导数存在最大值，为梯度的模，因此函数在一点沿梯度方向的变化率最大。梯度方向是函数值增长最快的方向，则负梯度方向为函数值减少最快的方向，梯度下降法通过这样的原理建立起来。

梯度下降法是求解无约束优化问题的一种常用的方法，这是一种迭代算法，选取合适的初值$x^{(0)}$，不断迭代更新x的值，进行目标函数的极小化直至收敛。通过这种方法考虑求解损失函数的极小值。明确目标为：

$$\theta_0^*, \theta_1^* = \underset{\theta_0,\theta_1}{\operatorname{argmin}} J(\theta_0, \theta_1)$$

梯度下降法算法流程：

输入：目标函数$J(\theta)$，梯度函数$\nabla J(\theta)$，计算精度ϵ，学习率λ；

输出：目标函数J的极小值点θ^*；

(1) 取初始值$\theta^{(0)} \in \mathcal{R}^n$，置$k=0$

(2) 计算$J(\theta^{(k)})$，计算梯度$\nabla J(\theta^{(k)})$

(3) 置$\theta^{(k+1)} = \theta^{(k)} - \lambda \nabla J(\theta^{(k)})$，计算$J(\theta^{(k+1)})$，当$||J(\theta^{(k+1)}) - J(\theta^{(k)})|| < \epsilon$或$||\theta^{(k+1)} - \theta^{(k)}|| < \epsilon$时，停止迭代，令$\theta^* = \theta^{(k+1)}$

(4) 否则：置$k = k+1$，转(3)

梯度下降法的解以及其收敛情况对于目标函数和输入参数有一定的要求，当目标函数不为凸函数时，梯度下降法得到的最终结果不一定为全局最优解，同时计算精度以及学习率的参数设置以及初始参数的设置对算法的收敛情况有较大的影响。

1.3 多变量线性回归

在此次实验中，我们采取的实验数据集规模如下：共计样本506例，每个样本的预测变量有13个，均为关于房价的各项指标，而响应变量有1个，为实际房价。因此问题转化为一个多元的线性回归模型，参考上述1.1和1.2中的单变量线性回归模型以及梯度下降算法，我们可以得到如下结论：

线性回归模型可以表示为：

$$h_\theta = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \tag{4}$$

若定义$x_0 = 1$，则可以表示：$h_\theta = \boldsymbol{\theta}^T \boldsymbol{X}$，其中$\boldsymbol{\theta} = [\theta_0, \theta_1, \cdots, \theta_n]^T$,$\boldsymbol{X} = [x_0, x_1, \cdots, x_n]$

对θ进行迭代的公式可以表示为：

$$\begin{aligned} \theta_j &:= \theta_j - \alpha \frac{\partial}{\partial \theta_j} \quad j = 0, 1, \cdots, n \\ &:= \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \quad j = 0, 1, \cdots, n \end{aligned} \tag{5}$$

二 Python代码实现

2.1 读取数据

实验给出的数据为“housing.data”，通过python自带的open函数对数据进行读入，使用正则表达式和一些简单的数组处理函数对数据进行简单的清洗，实现代码如下：

```
def reading_data(self,filepath,data_set):
    with open(filepath,'r') as data:
        #print(data)
        for line in data:
            # using regular expression
            #print(len(line))
            line = re.split(r'\s+',line)
            #print(len(line),line)
            if line[0] == '':
                line.pop(0)
            #print(len(line),line)
            line.pop()
            #str2int
            line = list(map(eval,line))
            data_set.append(line)
    return data_set
```

2.2 数据标准化即训练集测试集划分

将房价数据读入后形成array型的二维数组，维度为506×14，其中最后一列为响应变量y。下面首先对数据进行标准化处理，这里采用Z-score标准化，同时通过sklearn库中的model_selection类中的train_test_split函数进行数据集的随机划分，实现代码如下：

```
    def feature_scalling(self,X):
        mean = X.mean(axis=0)
        std = X.std(axis=0)
        return (X - mean) / std
    def data_divide(self,dataset):
        dataset = np.array(dataset)
        x = dataset[:,:-1]
        y = dataset[:,-1]
        x = Linear_regression.feature_scalling(self,x)
        x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3,random_state = 0)
        data_one_1 = np.ones(len(x_train))
        data_one_2 = np.ones(len(x_test))
        x_train = np.c_[data_one_1,x_train]
        x_test = np.c_[data_one_2,x_test]
        print(x_train.shape,x_test.shape,y_train.shape,y_test.shape)
        return x_train,x_test,y_train,y_test
```

2.3 梯度下降法实现

梯度下降法的原理前文已经给出，这里不再加以赘述，其输入参数主要为：数据训练集，初始化系数，迭代步长，计算精度以及最大迭代次数，输出为最优系数。给出python的实现代码：

```
    def Gradient_Descent(self,data_x,data_y,theta,alpha,MaxIteration,epsilon):
        data_x = np.array(data_x)
        data_y = np.array(data_y)
        m,n = np.shape(data_x)
        print(m,n)
        counter = 0
        theta = np.array(theta)
        #begin iteration
        error_0 = 0
        diff = []
        while counter <= MaxIteration:
            counter += 1
            error_1 = 0
            diff = np.zeros(n)
            for element in diff:
                element = 0
            for i in range(m):
                for j in range(n):
                    diff[j] += (np.dot(theta,data_x[i]) - data_y[i]) * data_x[i,j]
            for a in range(n):
                theta[a] -= alpha * diff[a]
            for k in range(m):
                error_1 += (data_y[k] - np.dot(theta,data_x[k]))**2 / (2*m)
            print("error:",error_1)
            print(error_1 - error_0)
            if abs(error_1 - error_0) < epsilon:
                break
            else:
                error_0 = error_1
        return theta
```

2.4 测试集预测及图形绘制

通过梯度下降法得到多元线性回归的各项系数，利用求解得到的最优回归方程计算测试集中数据的房价预测值，对比理论值与实际值，绘制散点图，进行结果可视化，给出实现代码：

```
    def Prediction(self,testing_x,theta):
        predcting_y = []
        for i in range(len(testing_x)):
            predcting_y.append(np.dot(testing_x[i],theta.T))
        print(predcting_y)
        return predcting_y

    #plot_1
    def plot_data(self,predicting_data,real_data):
        length = len(predicting_data)
        x_data = range(1,length+1)
        plt.figure(num = "Predicting_Data",figsize = (8,5),dpi = 120)
        x_value = list(range(0,len(predicting_data)))
        plt.grid(True)
        plt.title('True data and predict data scatter plot')
        plt.scatter(x_data,real_data,marker='*',label='True data')
        plt.scatter(x_data,predicting_data,marker='.',label='Predict data')
        plt.legend()
        plt.xlabel('data_index')
        plt.ylabel('data_value')
        plt.savefig('scatter_value.png')
        plt.show()

    #plot2
    def f(self,a,b):
        return (a,b) if a>b else(b,a)
    def plot_contract(self,predict_data,true_data):
        a = int(np.max(predict_data))
        b = int(np.max(true_data))
        c = Linear_regression.f(self,a,b)[0]
        x = range(c)
        y = range(c)
        plt.figure(num="Predicting_Data", figsize=(6, 4), dpi=120)
        plt.title('True data and predict data')
        plt.plot(x,y)
        plt.scatter(true_data,predict_data,marker='*',c='r',label='data')
        plt.savefig('zhexian.png')
        plt.show()
```

三 结果及图形展示

给出主函数的代码：

```
if __name__ == '__main__':
    test = Linear_regression
    data_set = []
    data_test = test.reading_data(test,'house.txt',data_set)
    training_x,testing_x,training_y,testing_y = test.data_divide(test,data_set)
    length = len(training_x[0])
    theta_init = np.random.random(size = length)
    alpha = 0.0005
    MaxIteration_1 = 1000
    epsilon_1 = 0.005
```

```
#print(theta_init)
theta_result = test.Gradient_Descent(test,training_x,training_y,theta_init,alpha,
                                                    MaxIteration_1,epsilon_1)
print(theta_result)
result_y = test.Prediction(test,testing_x,theta_result)
print(testing_y)
print(result_y)
test.plot_data(test,result_y,testing_y)
test.plot_contract(test,result_y,testing_y)
```

参数的设置如下：学习率即不步长设为：0.0005，最大迭代次数为：1000次，退出迭代的参数：0.005（前后两次误差之间的差值低于阈值即退出迭代）我们首先得到的是开始迭代与最终迭代完成后的数据的损失函数的值以及两次迭代之间误差的差值图像（图1）

```
error: 10.13676189166253
-0.006567700122337428
error: 10.130522993442531
-0.006238898219999456
error: 10.12459331742949
-0.005929676013041174
error: 10.11895465215331
-0.005638665276180177
error: 10.113590037727748
-0.005364614425561243
error: 10.108483661999346
-0.005106375728402668
error: 10.103620767648243
-0.004862894351102653
```

```
error: 185.1957618735459
185.1957618735459
error: 127.91989297262293
-57.27586890092297
error: 89.73702749223965
-38.182865480383285
error: 64.15944549401095
-25.577581998228695
error: 46.980831832235275
-17.178613661775678
error: 35.4127970526759
-11.568034779559376
error: 27.60136098016609
-7.81143607250981
```

图 1: 迭代开始与结束损失函数比较

最终的系数向量可以表示为图2：

```
[22.4670476  -0.89778615  0.82699716 -0.26159611  0.70588799 -1.24429977
  2.92545379 -0.38487585 -2.69612794  1.01738182 -0.92639008 -2.06803477
  0.63523203 -3.39880696]
```

图 2: Linear Regression Coefficients

最后我们对误差进行检查：

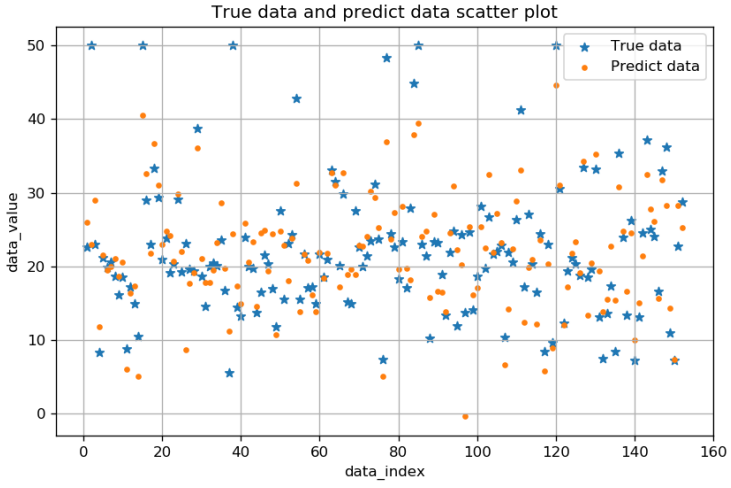

图 3: Linear Regression Coefficients

图 4: Linear Regression Coefficients

图3表示的是真实值和预测值之间的散点示意图，图4表示的折线为$y = x$，横轴表示真实值，纵轴表示预测值。

四　总结体会

线性回归的总体实验比较简单，但是还是有很多细节可以去深究，比如说在进行梯度下降法求解时，需要考虑一些参数的设置，尤其是步长的设置，中间出现了由于步长较大，越过收敛点极值点，误差反而变大的情况，需要进行检测。另外本次实验的一开始没有对数据进行归一化，导致数据难以收敛，因此走了不少弯路，这个值得吸取教训。除此以外还有一点，在进行数值计算的时候应该减少循环的使用，尽量用矩阵进行运算可以提升不少速度。