山东大学计算机科学与技术学院

机器学习与模式识别课程实验报告

学号：201600181073	姓名： 唐超	班级： 16 人工智能
实验题目：Multivariate Linear Regression		
实验学时：2	实验日期： 2018.9.30	

实验目的：

已给出 47 组房屋的面积、卧室数及价格的训练数据，建立一个价格与面积、卧室数的多元线性回归模型。

硬件环境：

DELL 台式机

软件环境：

MATLAB2016a

实验步骤与内容：

1. 下载、解压实验所需的数据文件，并加载至 MATLAB 中，将房屋的面积和卧室数分别作为第二列、第三列放入 47×3 的矩阵 x 中，x 的第一列全为 1；房屋价格作为 47 维向量 y；
2. 由于房屋面积的数值几乎是卧室数的 1000 倍，为提高梯度下降算法的效率，需要将两类数据作标准差标准化处理；
3. 此多元线性回归模型的方程为

$$h_\theta(x)=\theta^T x=\sum_{i=0}^{n}\theta_i x_i \tag{1}$$

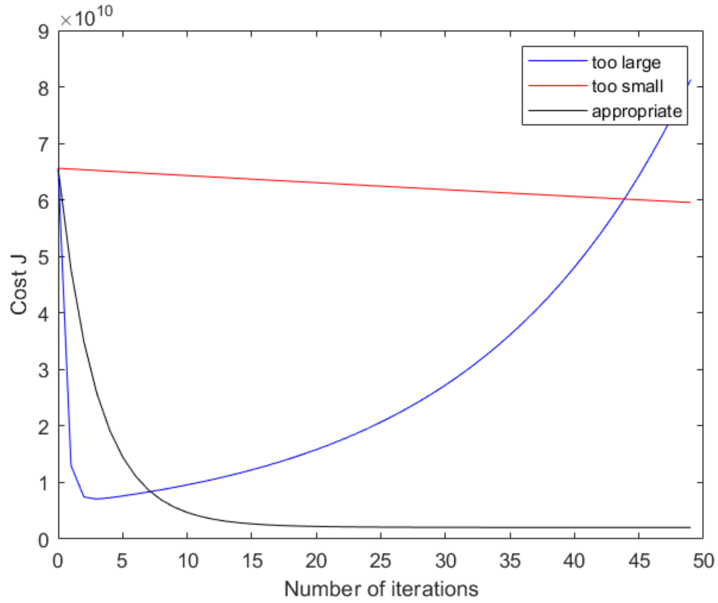
其中 $n=2$，θ 是我们要求的线性回归方程的系数组成的三维向量，根据损失函数 $J(\theta)$ 对 θ 的导数为零，有 θ 梯度下降的迭代公式如下：

$$\theta_j := \theta_j-\alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}-y^{(i)})x_j^{(i)} \tag{2}$$

这里我们首先要确定学习率 $\alpha(0.001\le\alpha\le10)$ ，通过判断损失函数：

$$J(\theta)=\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}-y^{(i)})^2 \tag{3}$$

的下降速度是否合适，选择合适的 α 。通过多次尝试我们发现 α 取 0.15 是比较合适的。



4. 将上一步得到的最优的α值代入（2）中进行迭代，经过 120 次迭代后θ几乎不再变化，此时$\theta = (340412.6584202795, 110611.6481846759, -6630.072176652521)$，即为所求；
5. 用得到的该模型的多元线性回归方程预测 1650 平方英尺、带有 3 个卧室的房间的价格，将相关信息代入方程得预测价格 $price0 = 293085.685$ ；
6. 尝试采用另一种方法直接求解θ，正规方程：

$$\theta = (X^T X)^{-1} X^T \vec{y} \tag{4}$$

采用此方程求解不必对原始数据进行标准化处理，但考虑到前面的步骤已经对数据做过处理，仍采用之前的 x，y 求解也无妨。得到 $\theta_1 = (340412.6595744681, 110631.0502788460, -6649.474270819764)$，与$\theta$ 相差很小，通过θ_1 决定的回归方程预测同样 1650 平方英尺、带有 3 个卧室的房间的价格为 $price1 = 293081.464$，与 $price0$ 仅相差 4. 221。

结论分析与体会：

- 在这种数据集较小的情况下，用正规方程可以直接求解θ，相比梯度下降方法更加简单有效，数据集较大时用正规方程就会变得难以求解，而梯度下降的使用范围就更加广泛，而且通过多次迭代也能满足一定的精度要求；
- 在选择合适的学习率α时，发现当α太小时，$J(\theta)$的下降速度很慢，达到收敛所需的次数会非常高。当α太大时，可能使$J(\theta)$超过局部最小值导致无法收敛；
- 两种方法所求得的θ相差很小，所预测的房屋价格也几乎一样，两种方法相互印证，说明了所得到的回归方程的正确性；

附录：程序源代码

```
x = load('ex2x.dat');

y = load('ex2y.dat');


m = length(y);

x = [ones(m,1),x];


sigma = std(x);

mu = mean(x);

x(:,2) = (x(:,2)-mu(2))./sigma(2);

x(:,3) = (x(:,3)-mu(3))./sigma(3);


theta = [0 0 0]';

alpha = 1.33;   %alpha 太大

J1 = zeros(50,1);

for num_iterations = 1:50

    J1(num_iterations) = (x*theta-y)'*(x*theta-y)/(2*m);

    theta = theta-(alpha/m)*x'*(x*theta-y);

end


theta = [0 0 0]';

alpha = 0.001;    %alpha 太小

J2 = zeros(50,1);

for num_iterations = 1:50

    J2(num_iterations) = (x*theta-y)'*(x*theta-y)/(2*m);

    theta = theta-(alpha/m)*x'*(x*theta-y);

end
```

```
theta = [0 0 0]';
alpha = 0.15;      %alpha 合适
J3 = zeros(120,1);
for num_iterations = 1:120
    J3(num_iterations) = (x*theta-y)'*(x*theta-y)/(2*m);
    theta = theta-(alpha/m)*x'*(x*theta-y);     %final theta
end

figure;
plot(0:49,J1(1:50),'b-');
hold on
plot(0:49,J2(1:50),'r-');
plot(0:49,J3(1:50),'k-');
xlabel('Number of iterations')
ylabel('Cost J')
legend('too large','too small','appropriate')

figure;
plot3(x(:,2),x(:,3),y,'s')
hold on
grid on
plot3(x(:,2),x(:,3),x*theta,'s')

%normal equation
theta1 = ((x'*x)^(-1))*x'*y;
%predict
p = [1,1650,3];
p(:,2) = (p(:,2)-mu(2))./sigma(2);
p(:,3) = (p(:,3)-mu(3))./sigma(3);
price0 = p*theta;
price1 = p*theta1;
```

```
deltap=price0-price1;
```