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：202300130183 | 姓名： 宋浩宇 | | 班级：23级人工智能班 |
| 实验题目：Multivariate Linear Regression | | | |
| 实验学时：2 | | 实验日期：2025/3/4 | |
| 实验环境：  软件环境：  系统：Windows 11 家庭中文版23H2 22631.4317  计算软件：MATLAB 版本: 9.8.0.1323502 (R2020a)  Java 版本: Java 1.8.0\_202-b08 with Oracle Corporation Java HotSpot(TM) 64-Bit Server VM mixed mode  硬件环境：  CPU：13th Gen Intel(R) Core(TM) i9-13980HX 2.20 GHz  内存：32.0 GB (31.6 GB 可用)  磁盘驱动器：NVMe WD\_BLACKSN850X2000GB  显示适配器：NVIDIA GeForce RTX 4080 Laptop GPU | | | |
| 1. 实验内容   In this exercise, you will investigate multivariate linear regression using gradient descent and the normal equations. You will also examine the relationship  between the cost function J(θ), the convergence of gradient descent, and the  learning rate α.   1. 实验步骤 2. 获取实验使用的数据。 3. 构造模型、损失函数以及设置梯度下降方式。      1. 用matlab代码实现并进行计算。 2. 用模型计算预测结果。 3. 测试不同学习率的效果。 4. 计算标准化方程。 5. 用标准化模型计算预测结果。 6. 测试结果   (1)J(θ)的表现    （2）多个学习率的J(θ)比较 .    （3）选择好学习率后计算出来的θ：    以及预测结果也在其中  （4）标准化计算的θ    预测结果：    精确的数值为：  从结果可以看出梯度下降的方式和标准化计算的方式会有一定的差距并不一样，但差的并不是很多。     1. 附录：实现源代码  |  | | --- | | %% 清空工作区  clc;  clear;  format long g;  %% 声明全局变量  % global x;  % global y;  global theta;  global alpha;  global num\_iterations;  theta = zeros(3,1);  num\_iterations = 50;  alpha = 0.07;  %% 加载数据集  x = load("ex2Data/ex2x.dat");  y = load("ex2Data/ex2y.dat");  % disp(x);  % disp(y);  %% 数据标准化  m = size(x, 1);  x = [ones(m,1),x];  % disp(x);  sigma = std(x);  mu = mean(x);  % disp(sigma);  % disp(mu);  x(:,2) = (x(:,2) - mu(2))./sigma(2);  x(:,3) = (x(:,3) - mu(3))./sigma(3);  %% 梯度下降  % J\_figure\_disp(x,y,theta);  % learning\_rate\_compare(x,y);  % training(x,y);  % disp(theta);  % theta\_normal(x,y);  training(x,y);  disp(theta);  test\_data = [1,1650,3];  normal\_test\_data = [1,1650,3];  normal\_test\_data(2) = (test\_data(2) - mu(2))./sigma(2);  normal\_test\_data(3) = (test\_data(3) - mu(3))./sigma(3);  disp(h(normal\_test\_data));  %% h(x)  function result = h(x)      global theta;      result = x \* theta;  end  %% gradientDecent  function result = gradientDecent(x, y)      global theta;      global alpha;      m = size(x, 1);      sum1 = 0;      sum2 = 0;      sum3 = 0;      for i = 1:m          sum1 = sum1 + (h(x(i,:)) - y(i,:))\*x(i,1);          sum2 = sum2 + (h(x(i,:)) - y(i,:))\*x(i,2);          sum3 = sum3 + (h(x(i,:)) - y(i,:))\*x(i,3);      end      sum1 = sum1 / m;      sum2 = sum2 / m;      sum3 = sum3 / m;      theta(1) = theta(1) - alpha \* sum1;      theta(2) = theta(2) - alpha \* sum2;      theta(3) = theta(3) - alpha \* sum3;  end  %% J(θ)  function result = J(x,y)      sum = 0;      for i = 1:size(x,1)          sum = sum + (h(x(i,:)) - y(i,:))^2;      end      result = sum / (2\*size(x,1));  end  %% training  function result = training(x,y)      global num\_iterations;      for i = 1:num\_iterations          gradientDecent(x, y);      end  end  %% J\_figure\_disp  function J\_figure\_disp(x,y,theta)      global num\_iterations;      J\_vals = zeros(num\_iterations,1);      for i = 1:num\_iterations          J\_vals(i) = J(x,y);          gradientDecent(x, y);      end      figure;      plot(0:49,J\_vals(1:50),'-');      xlabel('Number of iterations');      ylabel('Cost J');  end  %% learning\_rate\_compare  function learning\_rate\_compare(x,y)      global alpha;      global num\_iterations;      global theta;      figure;      alpha\_vals = [0.07,0.1,1];      alpha = 0.07;      theta = zeros(3,1);      J\_vals = zeros(num\_iterations,1);      for i = 1:num\_iterations          J\_vals(i) = J(x,y);          gradientDecent(x, y);      end      plot(0:49,J\_vals(1:50),'-');      hold on;      J\_vals = zeros(num\_iterations,1);      theta = zeros(3,1);      alpha = alpha\_vals(1);      for i = 1:num\_iterations          J\_vals(i) = J(x,y);          gradientDecent(x, y);      end      plot(0:49,J\_vals(1:50),'b-');      J\_vals = zeros(num\_iterations,1);      theta = zeros(3,1);      alpha = alpha\_vals(2);      for i = 1:num\_iterations          J\_vals(i) = J(x,y);          gradientDecent(x, y);      end      plot(0:49,J\_vals(1:50),'r-');      J\_vals = zeros(num\_iterations,1);      theta = zeros(3,1);      alpha = alpha\_vals(3);      for i = 1:num\_iterations          J\_vals(i) = J(x,y);          gradientDecent(x, y);      end      plot(0:49,J\_vals(1:50),'k-');      xlabel('Number of iterations');      ylabel('Cost J');  end  %% 正规方程求theta  function theta\_normal(x,y)      global theta;      theta = inv((x'\*x))\*(x'\*y);  end | | | | |