# 实验二：线性回归

|  |  |
| --- | --- |
| 姓名： 郭帆 | 学号：2021112240 |

* 实验目的

理解和掌握线性回归模型基本原理和方法，学会使用线性回归模型对分析问题进行建模和预测，掌握线性问题上模型评估方法。

* 实验内容

1. 假设线性模型为，在给定数据集上训练模型，得到模型参数，计算模型在测试集上均方误差，并将训练数据、测试数据、训练模型绘制在一张图中。
2. 假设二次线性模型为，在给定数据集上训练模型，得到模型参数，计算模型在测试集上均方误差，并将训练数据、测试数据、训练模型绘制在一张图中。

* 实验环境

python

numpy

matplotlib

* 实验代码

1. 代码
2. **import** matplotlib.pyplot as plt
3. **import** numpy as np
5. test\_pth = "C:/Users/Dell/Desktop/MLexp/exp2/data/experiment\_02\_testing\_set.csv"  ### 测试集路径
6. train\_pth = "C:/Users/Dell/Desktop/MLexp/exp2/data/experiment\_02\_training\_set.csv"  ### 训练集路径
8. train\_set = np.loadtxt(train\_pth, delimiter=',', dtype=float)  ### 读入训练集
9. test\_set = np.loadtxt(test\_pth, delimiter=',', dtype=float)  ### 读入测试集

12. **class** Linear:
13. """
14. 定义线性回归类形式为 y = wx + b
15. """
16. **def** \_\_init\_\_(self, data):
17. """
18. 初始化线性回归模型的权重参数 w，从随机生成的数据中获取维度信息。
20. :param data: 训练集数据，numpy数组
21. """
22. self.weight = np.mat(np.random.random((data.shape[1], 1)))
24. **def** forward(self, input):
25. """
26. 前向传播计算模型的预测值
28. :param input: 输入特征数据
29. :return: 模型的预测值
30. """
31. data = np.insert(input, input.shape[1], np.ones(self.weight.shape[0] - input.shape[1]), axis=0)
32. output = np.matmul(self.weight.T, data)
33. **return** output
35. **def** backward(self, input, y):
36. """
37. 反向传播更新模型的权重参数
39. :param input: 输入特征数据
40. :param y: 真实标签数据
41. """
42. input = np.insert(input, input.shape[1], np.ones(self.weight.shape[0] - input.shape[1]), axis=1)
43. self.weight = np.matmul(input.T, input)
44. self.weight = np.linalg.inv(self.weight)
45. self.weight = np.matmul(np.matmul(self.weight, input.T), y)

48. # 创建线性回归模型对象
49. model = Linear(train\_set)
51. # 从训练集中提取特征 x 和标签 y
52. x = train\_set[:, 0].reshape(-1, 1)
53. y = train\_set[:, 1].reshape(-1, 1)
55. # 使用反向传播算法更新模型参数
56. model.backward(x, y)
58. train\_loss = []
59. pre\_y = []
60. x\_label = []
61. # 遍历训练集数据
62. **for** i, (x, y\_true) **in** enumerate(train\_set):
63. x\_label.append(x)
64. x = x.reshape(-1, 1)
65. # 使用训练好的模型进行预测
66. y\_pre = model.forward(x)
67. pre\_y.append(y\_pre[0][0])
68. # 计算训练集上的损失
69. train\_loss.append((y\_true - y\_pre[0][0]) \* (y\_true - y\_pre[0][0]))
71. **print**("train loss: ", sum(train\_loss) / len(train\_loss))
73. test\_loss = []
74. # 遍历测试集数据
75. **for** i, (x, y\_true) **in** enumerate(test\_set):
76. x\_label.append(x)
77. x = x.reshape(-1, 1)
78. # 使用训练好的模型进行预测
79. y\_pre = model.forward(x)
80. pre\_y.append(y\_pre[0][0])
81. # 计算测试集上的损失
82. test\_loss.append((y\_true - y\_pre[0][0]) \* (y\_true - y\_pre[0][0]))
84. **print**("test loss: ", sum(test\_loss) / len(test\_losss))
86. # 绘制训练集和测试集的散点图，并画出线性回归拟合线
87. fig = plt.Figure(figsize=(20, 8), dpi=80)
88. x = train\_set[:, 0]
89. y = train\_set[:, 1]
90. plt.scatter(x, y, label='train\_set')   ### 训练集可视化
92. x = test\_set[:, 0]
93. y = test\_set[:, 1]
94. plt.scatter(x, y, label='test\_set')   ### 测试集可视化
96. plt.plot(x\_label, pre\_y, label='linear\_regression', c='r')  ### 模型可视化
98. plt.xlabel('x')
99. plt.ylabel('y')
100. plt.title("Linear Regression: y = wx + b")
102. plt.legend()
103. plt.show()
105. print(“模型的参数为 : -----------\n”, model.weight)
106. 代码
107. **import** numpy as np
108. **from** matplotlib **import** pyplot as plt
110. test\_pth = "C:/Users/Dell/Desktop/MLexp/exp2/data/experiment\_02\_testing\_set.csv"  ### 测试数据路径
111. train\_pth = "C:/Users/Dell/Desktop/MLexp/exp2/data/experiment\_02\_training\_set.csv"  ### 训练数据路径
113. train\_set = np.loadtxt(train\_pth,delimiter=',', dtype=float)   ### 读入测试集
114. test\_set = np.loadtxt(test\_pth,delimiter=',', dtype=float)     ### 读入训练集

117. **class** Linear:
118. """
119. 使用闭式解的二次线性回归模型。
120. """
121. **def** \_\_init\_\_(self):
122. """
123. 初始化 Linear 类，设定随机权重。
124. """
125. self.weight = np.mat(np.random.random((3, 1)))
127. **def** backward(self, input, y\_train):
128. """
129. 使用闭式解计算权重。
131. :param input: 输入数据。
132. :param y\_train: 目标值。
133. """
134. X\_train\_augmented = np.c\_[input \*\* 2, input, np.ones\_like(input)]  # 构建增广矩阵，包括 x^2 和 x
135. self.weight = np.linalg.inv(X\_train\_augmented.T.dot(X\_train\_augmented)).dot(X\_train\_augmented.T).dot(y\_train) ### 计算最终的参数

138. model = Linear()
140. ### 构建特征和目标值
141. X\_train = train\_set[:, 0].reshape(-1,1)
142. X\_test = test\_set[:, 0].reshape(-1,1)
143. y\_train = train\_set[:, 1].reshape(-1,1)
144. y\_test = test\_set[:, 1].reshape(-1,1)
146. model.backword(X\_train , y\_train)   ### 构造闭式解
148. # 构建增广矩阵，包括 x^2 和 x
149. X\_train\_augmented = np.c\_[X\_train\*\*2, X\_train, np.ones\_like(X\_train)]
150. X\_test\_augmented = np.c\_[X\_test\*\*2, X\_test, np.ones\_like(X\_test)]
152. y\_pred\_train = X\_train\_augmented.dot(model.weight)   ### 计算训练集合上的预测值
153. train\_loss = np.mean((y\_train - y\_pred\_train)\*\*2)    ### 计算训练集上的损失函数
154. **print**("train loss : " , train\_loss)
156. y\_pred\_test = X\_test\_augmented.dot(model.weight)   ### 计算训练集合上的预测值
157. test\_loss = np.mean((y\_test - y\_pred\_test)\*\*2)     ### 计算训练集上的损失函数
158. **print**("test loss : " , test\_loss)
160. ### 对结果和数据集进行可视化
161. fig = plt.Figure(figsize=(20,8) , dpi = 80)
162. x = train\_set[:,0]
163. y = train\_set[:,1]
164. plt.scatter(x,y, label = 'train\_set')   ### 训练集可视化
166. x = test\_set[:,0]
167. y = test\_set[:,1]
168. plt.scatter(x,y, label = 'test\_set')   ### 测试集可视化
170. X = np.insert(X\_train , X\_train.shape[0] , X\_test , axis=0).reshape(-1,)  ### 将训练集和测试集的特征进行拼接
171. sorted\_indices = np.argsort(X)
172. X = X[sorted\_indices].reshape(-1,1)   ### 对不同样本的特征进行排序
173. x\_label = np.c\_[X\*\*2, X, np.ones\_like(X)]  # 构建增广矩阵，包括 x^2 和 x
174. pre\_y = x\_label.dot(model.weight)
175. x\_label = np.insert(X\_train , X\_train.shape[0] , X\_test , axis=0).reshape(-1,)
176. sorted\_indices = np.argsort(x\_label)
177. x\_label = x\_label[sorted\_indices].reshape(-1,1)
178. plt.plot(x\_label , pre\_y , label = 'Fitted Model' , c = 'g' , linewidth=3)   ### 画出最终拟合的模型
179. ### 对图像进行标注
180. plt.xlabel('x')
181. plt.ylabel('y')
182. plt.title("Linear Regression : y = wx^2 + wx + b")
184. plt.legend()
186. plt.show()
188. print(“模型的权重为 : -----------\n”, model.weight)

* 结果分析

1. 模型参数为：

测试集均方误差为：

4.625633774506116

绘图结果为：

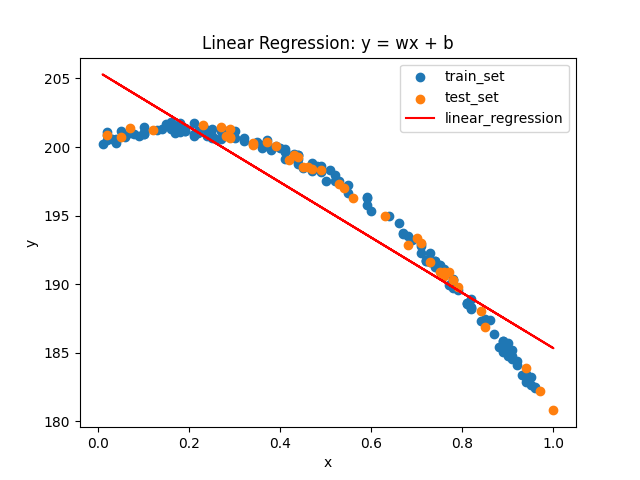


图1 线性模型为对数据集的拟合

1. 模型参数为：

测试集均方误差为：

0.1031414687653784

绘图结果为：

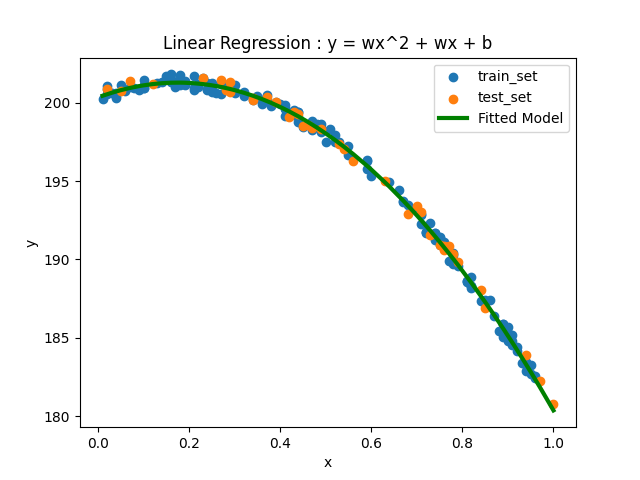


图2 线性模型为对数据集的拟合

将看成新的特征，构建增广矩阵 ，，可以利用最小二乘法通过求得参数得闭式解。