平时作业——航班预测

# Pytorch代码

1. # %%
2. #导入包以及设置随机种子
3. **import** numpy as np
4. **import** pandas as pd
5. **import** random
6. **import** matplotlib.pyplot as plt
7. **import** torch
8. **import** torch.nn as nn
10. seed = 42
11. torch.manual\_seed(seed)
12. np.random.seed(seed)
13. random.seed(seed)
14. torch.cuda.manual\_seed\_all(seed)
16. # %%
17. # 读取第二列数据
18. all\_data  = pd.read\_csv('./data.csv',usecols=[1],header=None)
19. # 数据可视化
20. **print**(all\_data.describe())
21. plt.plot(all\_data)
23. # %%
24. # 数据预处理
25. **from** sklearn.preprocessing **import** MinMaxScaler
26. **def** sliding\_windows(data, seq\_length):
27. x = []
28. y = []
30. **for** i **in** range(len(data)-seq\_length):
31. \_x = data[i:(i+seq\_length)]
32. \_y = data[i+seq\_length]
33. x.append(\_x)
34. y.append(\_y)
36. **return** np.array(x),np.array(y)
38. sc = MinMaxScaler()
39. training\_data = sc.fit\_transform(all\_data)
41. seq\_length = 2
42. x, y = sliding\_windows(training\_data, seq\_length)
44. #train\_size = int(len(y) \* 0.67)
45. #test\_size = len(y) - train\_size
46. # 使用143个月的数据训练 不划分训练集和测试集
47. dataX = torch.Tensor(np.array(x))
48. dataY = torch.Tensor(np.array(y))
50. #trainX = torch.Tensor(np.array(x[0:train\_size]))
51. #trainY = torch.Tensor(np.array(y[0:train\_size]))
53. #testX = torch.Tensor(np.array(x[train\_size:len(x)]))
54. #testY = torch.Tensor(np.array(y[train\_size:len(y)]))
56. # %%
57. # 定义模型
58. **class** RNN(nn.Module):
59. **def** \_\_init\_\_(self, rnn\_layer, seq\_length):
60. super(RNN, self).\_\_init\_\_()
61. self.rnn = rnn\_layer
62. **if** **not** self.rnn.bidirectional:
63. self.num\_directions = 1
64. self.linear = nn.Linear(self.rnn.hidden\_size\*seq\_length, 1)
65. **else**:
66. self.num\_directions = 2
67. self.linear = nn.Linear(self.rnn.hidden\_size\*self.num\_directions\*seq\_length, 1)
69. **def** forward(self, x, state):
70. out, state = self.rnn(x, state)
71. output = self.linear(out.reshape(out.size(0), -1))
72. **return** output, state
74. **def** begin\_state(self, batch\_size=1):
75. **if** **not** isinstance(self.rnn, nn.LSTM):
76. **return**  torch.zeros((self.num\_directions \* self.rnn.num\_layers,
77. batch\_size, self.rnn.hidden\_size))
78. **else**:
79. **return** (torch.zeros((
80. self.num\_directions \* self.rnn.num\_layers,
81. batch\_size, self.rnn.hidden\_size)),
82. torch.zeros((
83. self.num\_directions \* self.rnn.num\_layers,
84. batch\_size, self.rnn.hidden\_size)))
86. # %%
87. num\_epochs = 1000
88. learning\_rate = 0.01
90. input\_size = 1
91. hidden\_size = 2
92. num\_layers = 1
93. multi\_layers = 5

96. criterion = torch.nn.MSELoss()    # 定义损失函数
98. # %%
99. **def** train\_model(model,
100. criterion,
101. x\_train,
102. y\_train,
103. epochs=1000):
104. state = model.begin\_state(x\_train.size(0))
106. optimizer = torch.optim.Adam(model.parameters(), lr=learning\_rate)
107. **for** epoch **in** range(epochs):
109. # put default model grads to zero
110. optimizer.zero\_grad()
112. # predict the output
113. pred,\_ = model(x\_train, state)
115. # calculate the loss
116. error = criterion(pred,y\_train)
118. # backpropagate the error
119. error.backward()
121. # update the model parameters
122. optimizer.step()



127. **if** epoch % 100 == 0:
128. **print**('Epoch :{}    Train Loss :{}'.format((epoch+1)/epochs, error.item()))
129. # 测试模型
130. **def** test\_model(model):
131. model.eval()
133. x = []
134. **for** i **in** range(len(training\_data)-seq\_length+1):
135. x.append(training\_data[i:i+seq\_length])
136. testX = torch.Tensor(np.array(x))
137. state = model.begin\_state(testX.size(0))
138. train\_predict,\_ = model(testX, state)
140. data\_predict = train\_predict.detach().numpy()
141. dataY\_plot = dataY.data.numpy()
143. data\_predict = sc.inverse\_transform(data\_predict)
144. dataY\_plot = sc.inverse\_transform(dataY\_plot)
146. # 数据可视化
147. plt.plot(dataY\_plot)
148. # 蓝色是真实值
149. plt.plot(data\_predict)
150. # 橘色是预测值
151. plt.suptitle('Flight-passengers Prediction')
152. # 增加图例
153. plt.legend([ 'Real', 'Pred'])
154. plt.show()
155. **print**('第144个月的乘客人数为：%d k'%data\_predict[-1])

158. # %%
159. #单层GRU
160. gru = nn.GRU(input\_size, hidden\_size, num\_layers, bidirectional=False, batch\_first=True)
161. gru = RNN(gru, seq\_length)
162. train\_model(gru,criterion,dataX,dataY,epochs=num\_epochs)
163. test\_model(gru)
165. # %%
166. #单层LSTM
167. lstm = nn.LSTM(input\_size, hidden\_size, num\_layers, bidirectional=False, batch\_first=True)
168. lstm = RNN(lstm, seq\_length)
169. train\_model(lstm,criterion,dataX,dataY,epochs=num\_epochs)
170. test\_model(lstm)
172. # %%
173. #多层LSTM
174. lstm\_multi = nn.LSTM(input\_size, hidden\_size, multi\_layers, bidirectional=False, batch\_first=True)
175. lstm\_multi = RNN(lstm\_multi, seq\_length)
176. train\_model(lstm\_multi,criterion,dataX,dataY,epochs=num\_epochs)
177. test\_model(lstm\_multi)
179. # %%
180. #双向LSTM
181. lstm\_bi = nn.LSTM(input\_size, hidden\_size, num\_layers, bidirectional=True, batch\_first=True)
182. lstm\_bi = RNN(lstm\_bi, seq\_length)
183. train\_model(lstm\_bi,criterion,dataX,dataY,epochs=num\_epochs)
184. test\_model(lstm\_bi)
186. # %%
187. #多层双向LSTM
188. lstm\_multi\_bi = nn.LSTM(input\_size, hidden\_size, multi\_layers, bidirectional=True, batch\_first=True)
189. lstm\_multi\_bi = RNN(lstm\_multi\_bi, seq\_length)
190. train\_model(lstm\_multi\_bi,criterion,dataX,dataY,epochs=num\_epochs)
191. test\_model(lstm\_multi\_bi)

# 预测结果

|  |  |
| --- | --- |
| 使用方法 | 144月乘客人数（k） |
| 实际值 | 432 |
| 单层GRU | 362 |
| 单层LSTM | 355 |
| 多层LSTM | 330 |
| 双向LSTM | 356 |
| 多层双向LSTM | 340 |