

```

1 # -*- coding: utf-8 -*-
2 """computation_lab1ipynb.ipynb
3
4 Automatically generated by Colaboratory.
5
6 Original file is located at
7     https://colab.research.google.com/drive/19NI1IgxLwPShKVoi79nCleyeJ8nXGfyh
8
9 # 计算方法实验
10
11 ## 矩阵运算的相关函数定义
12 """
13
14 # matrix operations
15 def mul_matrix(MA,MB):
16     if len(MA[0])!=len(MB):
17         print('dim error')
18         return
19     return [[sum(map(lambda a: a[0]*a[1], zip(l, s))) for l in zip(*MB)] for s in MA
20 ]
21
22 def naive_swap(M,row_zero_index):
23     swapA = M[row_zero_index]
24     # 需要与index下的非0行交换
25     for i in range(row_zero_index+1,len(M)):
26         if M[i][row_zero_index]!=0:
27             return swap(M,i,row_zero_index)
28     print("不满足初等变换条件")
29     return [[]]
30
31 def transpose(M):
32     Mt = []
33     for i in zip(*M):
34         Mt.append(list(i))
35     return Mt
36
37 # row operations
38 def sub(rowA,rowB):
39     return [i-j for i,j in zip(rowA,rowB)]
40 def add(rowA,rowB):
41     return [i+j for i,j in zip(rowA,rowB)]
42 def div(rowA,rowB):
43     return [i/j for i,j in zip(rowA,rowB)]
44 def times(rowA,rowB):
45     return [i*j for i,j in zip(rowA,rowB)]
46 def div_dig(row,dig):
47     dig_l = [dig]*len(row)
48     return div(row,dig_l)
49 def times_dig(row,dig):
50     dig_l = [dig]*len(row)
51     return times(row,dig_l)
52
53 def M_add(MA,MB):
54     assert (len(MA)==len(MB))
55     assert (len(MA[0])==len(MB[0]))
56     C = []
57     for i in range(len(MA)):
58         C.append(add(MA[i],MB[i]))
59     return C

```

```

60
61
62 def naive_tran(rowA,rowB,main_dig):
63     '''
64     @rowA updatetrowA
65     @rowB 主元
66     @main_dig主元位置
67     '''
68     rowA_dig = rowA[main_dig]
69     rowB_dig = rowB[main_dig]
70     if rowA_dig == 0:
71         return rowA
72     fac = rowB_dig/rowA_dig
73     rowA_update = sub(rowB.copy(),times_dig(rowA.copy(),fac))
74     return rowA_update
75
76 def swap(A,i,j):
77     tmp = A[i]
78     A[i] = A[j]
79     A[j] = tmp
80     return A
81 # 求逆
82 def r(M):
83     resM = M.copy()
84     row_element = len(M[0])
85     for index in range(len(M)):
86         new_row = [0]*row_element
87         new_row[index] = 1
88         resM[index] = resM[index]+new_row
89     for main_i in range(len(M)):
90         main_row = resM[main_i].copy()
91         diag_dig = resM[main_i][main_i]
92         for tra in range(len(M)):
93             if tra == main_i:
94                 continue
95             if diag_dig == 0:
96                 resM = naive_swap(resM,main_i)
97             if resM == [[]]:
98                 print("不满足求逆条件")
99                 return [[]]
100             tran_row = resM[tra].copy()
101             resM[tra]=naive_tran(main_row,tran_row,main_i)
102             resM[main_i] = div_dig(resM[main_i],resM[main_i][main_i])
103     return [row[row_element:] for row in resM]
104
105 def make_diag(shape):
106     diag = [[0]*shape for i in range(shape)]
107     for i in range(shape):
108         for j in range(shape):
109             if i==j:
110                 diag[i][j]=1
111             else :
112                 diag[i][j] = 0
113     return diag
114
115 def mul_const(lmd,diag):
116     size = len(diag)
117     return [[ diag[i][j]*lmd for i in range(size)] for j in range(size)]
118
119 """### 使用numpy库验证运算结果

```

```

120
121 结果一致
122 """
123
124 import numpy as np
125 B = np.array([[1,2,3,4],[2,3,1,2],[1,1,1,-1],[1,0,-2,6]])
126 print("numpy result: ")
127 print(np.linalg.inv(B))
128
129 B = [[1,2,3,4],[2,3,1,2],[1,1,1,-1],[1,0,-2,6]]
130 BI = r(B)
131 print("my result: ")
132 for i in BI:
133     print(i)
134
135 """## 导入红酒品质的数据集"""
136
137 def try_float(x):
138     try:
139         return float(x)
140     except ValueError:
141         return x
142
143 import csv
144 readin = []
145 with open("winequality-red.csv") as csvfile:
146     spamreader = csv.reader(csvfile, delimiter=',')
147     for row in spamreader:
148         readin.append([try_float(i) for i in row])
149 readin = readin[1:]
150
151 """## 划分数据集"""
152
153 def into_group(data_X,data_Y,group_num):
154     num_per_group = int(len(data_X)/group_num)
155     x=[]
156     y=[]
157     for i in range(group_num):
158         x.append(data_X[i*num_per_group:(i+1)*num_per_group])
159         y.append(data_Y[i*num_per_group:(i+1)*num_per_group])
160     return x,y
161
162 def regression_func(coff,X,y):
163     # X y are batch
164     tar = transpose(y.copy())[0]
165     pred = []
166     for i in X:
167         pred.append(sum(times(coff,i)))
168     sqrt_error = [ (t-p)**2 for t,p in zip(tar,pred)]
169     error = sum(sqrt_error)/len(sqrt_error)
170     return error, pred
171
172 """## 通过musk的方法定义五折交叉验证的函数 """
173
174 def train_regression(batch_X, batch_y, musk):
175     test_X = None
176     test_y = None
177     train_X = []
178     train_y = []
179     index = -1

```

```

180 for x,y in zip(batch_X,batch_y):
181     index+=1
182     if musk[index] == 0:
183         test_X = x
184         test_y = y
185         continue
186     else:
187         train_X+=x
188         train_y+=y
189     theta =
transpose(mul_matrix(mul_matrix(r(mul_matrix(transpose(train_X),train_X)),transpose
(transpose(train_X)),train_y))[0]
190 # train_error
191 train_error,_ = regression_func(theta,train_X,train_y)
192 # test_error
193 test_error,_ = regression_func(theta,test_X,test_y)
194 return train_error,test_error
195
196 def train_regression_L2(batch_X,batch_y,lmd,musk):
197     test_X = None
198     test_y = None
199     train_X = []
200     train_y = []
201     index = -1
202     for x,y in zip(batch_X,batch_y):
203         index+=1
204         if musk[index] == 0:
205             test_X = x
206             test_y = y
207             continue
208         else:
209             train_X+=x
210             train_y+=y
211     XTX = mul_matrix(transpose(train_X),train_X)
212     L2 = mul_const(lmd,make_diag(len(XTX)))
213     Xtrain_L2 = M_add(XTX,L2)
214     theta = transpose(mul_matrix(mul_matrix(r(    Xtrain_L2
),transpose(train_X)),train_y))[0]
215 # train_error
216 train_error,_ = regression_func(theta,train_X,train_y)
217 # test_error
218 test_error,_ = regression_func(theta,test_X,test_y)
219 return train_error,test_error
220
221 """## 机器参数"""
222
223 ! cat /proc/cpuinfo
224
225 """# 用多变量线性模型拟合数据，并计算测试集的 _平方误差和平均值_ 、 _运行时间_ 和 _内存_
226
227 ## 运行结果和耗时，avg-time cost是每一折使用的时间
228 """
229
230 data_X = [ [1]+row[:-1] for row in readin]
231 data_y = [ [row[-1]] for row in readin]
232
233 batch_X,batch_y = into_group(data_X,data_y,5)
234
235 names = ["01111","10111","11011","11101","11110"]
236 import time

```

```

237 import matplotlib.pyplot as plt
238 train_errors=[]
239 test_errors=[]
240 time_start=time.time()
241
242 for musk_test in range(5):
243     musk = [1]*5
244     musk[musk_test]=0
245     train_error,test_error = train_regression(batch_X,batch_y,musk)
246     train_errors.append(train_error)
247     test_errors.append(test_error)
248
249 time_end=time.time()
250
251 print('avg-time cost',(time_end-time_start)/5,'s')
252
253 plt.plot(train_errors,'-^')
254 plt.plot(test_errors,'-o')
255 plt.xticks(range(5),names)
256
257 plt.legend(['train_errors','test_errors'])
258 plt.show()
259
260 """# 用多变量线性模型拟合数据，使用L2正则项，重复上述实验"""
261
262 data_X = [ [1]+row[:-1] for row in readin]
263 data_y = [ [row[-1]] for row in readin]
264
265 batch_X,batch_y = into_group(data_X,data_y,5)
266
267 names = ["01111","10111","11011","11101","11110"]
268
269 import time
270 import matplotlib.pyplot as plt
271 train_errors=[]
272 test_errors=[]
273 time_start=time.time()
274 l2s = [0.0,0.1,0.3,0.5,0.7,0.9]
275 train_errors_in_l2=[]
276 test_errors_in_l2 = []
277 for l2 in l2s:
278     train_errors=[]
279     test_errors=[]
280     for musk_test in range(5):
281         musk = [1]*5
282         musk[musk_test]=0
283         train_error,test_error = train_regression_L2(batch_X,batch_y,l2,musk)
284         train_errors.append(train_error)
285         test_errors.append(test_error)
286     train_errors_in_l2.append(train_errors)
287     test_errors_in_l2.append(test_errors)
288
289 time_end=time.time()
290
291 print('avg-time cost',(time_end-time_start)/(5*len(l2s)),'s')
292
293 fig = plt.figure(figsize=(20, 8))
294 for i in range(1, 7):
295     index = i -1
296     ax = fig.add_subplot(2, 3, i)

```

```
297     ax.plot(train_errors_in_l2[index], '-^')
298     ax.plot(test_errors_in_l2[index], '-o')
299     ax.set_xticks(range(5), names)
300     ax.set_title("l2 = {}".format(l2s[index]))
301     ax.legend(['train_errors', 'test_errors'])
302
303
304 plt.show()
305
306 fig = plt.figure(figsize=(15, 6))
307 for i in range(1,7):
308     index = i-1
309     plt.plot(train_errors_in_l2[index], '-o')
310     plt.legend(['l2 = {}'.format(n) for n in l2s] )
311
312 plt.xticks(range(5), names)
313
314 plt.title('train_errors in l2={}'.format(l2s))
315
316 plt.show()
317
318 fig = plt.figure(figsize=(15, 6))
319 for i in range(1,7):
320     index = i-1
321     plt.plot(test_errors_in_l2[index], '-o')
322     plt.legend(['l2 = {}'.format(n) for n in l2s] )
323
324 plt.xticks(range(5), names)
325
326 plt.title('test_errors in l2={}'.format(l2s))
327
328 plt.show()
329
330 """## 检查消耗的内存"""
331
332 !pip install memory-profiler
333
334 !mprof run --include-children --multiprocess memory_record.py
335
336 !mprof plot --flame -o pic.png
```