```python
# -*- coding: utf-8 -*-
"""computation_lab1ipynb.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/19NI1IgxLwPShKVoi79nCleyeJ8nXGfyh

# 计算方法实验

## 矩阵运算的相关函数定义
"""

# matrix operations
def mul_matrix(MA,MB):
  if len(MA[0])!=len(MB):
    print('dim error')
    return
  return [[sum(map(lambda a: a[0]*a[1], zip(l, s))) for l in zip(*MB)] for s in MA
]


def naive_swap(M,row_zero_index):
  swapA = M[row_zero_index]
  # 需要与index下的非0行交换
  for i in range(row_zero_index+1,len(M)):
    if M[i][row_zero_index]!=0:
      return swap(M,i,row_zero_index)
  print("不满足初等变换条件")
  return[[]]

def transpose(M):
  Mt = []
  for i in zip(*M):
    Mt.append(list(i))
  return Mt

# row operations
def sub(rowA,rowB):
  return [i-j for i,j in zip(rowA,rowB)]
def add(rowA,rowB):
  return [i+j for i,j in zip(rowA,rowB)]
def div(rowA,rowB):
  return [i/j for i,j in zip(rowA,rowB)]
def times(rowA,rowB):
  return [i*j for i,j in zip(rowA,rowB)]
def div_dig(row,dig):
  dig_l = [dig]*len(row)
  return div(row,dig_l)
def times_dig(row,dig):
  dig_l = [dig]*len(row)
  return times(row,dig_l)

def M_add(MA,MB):
  assert (len(MA)==len(MB))
  assert (len(MA[0])==len(MB[0]))
  C = []
  for i in range(len(MA)):
    C.append(add(MA[i],MB[i]))
  return C
```

```
60
61
62  def naive_tran(rowA,rowB,main_dig):
63    '''
64    @rowA updaterowA
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
```

```
  for x,y in zip(batch_X,batch_y):
    index+=1
    if musk[index] == 0:
      test_X = x
      test_y = y
      continue
    else:
      train_X+=x
      train_y+=y
  theta = transpose(mul_matrix(mul_matrix(r(mul_matrix(transpose(train_X),train_X)),transpose(train_X)),train_y))[0]
  # train_error
  train_error,_ = regression_func(theta,train_X,train_y)
  # test_error
  test_error,_ = regression_func(theta,test_X,test_y)
  return train_error,test_error

def train_regression_L2(batch_X,batch_y,lmd,musk):
  test_X = None
  test_y = None
  train_X = []
  train_y = []
  index = -1
  for x,y in zip(batch_X,batch_y):
    index+=1
    if musk[index] == 0:
      test_X = x
      test_y = y
      continue
    else:
      train_X+=x
      train_y+=y
  XTX = mul_matrix(transpose(train_X),train_X)
  L2 = mul_const(lmd,make_diag(len(XTX)))
  Xtrain_L2 = M_add(XTX,L2)
  theta = transpose(mul_matrix(mul_matrix(r(   Xtrain_L2 ),transpose(train_X)),train_y))[0]
  # train_error
  train_error,_ = regression_func(theta,train_X,train_y)
  # test_error
  test_error,_ = regression_func(theta,test_X,test_y)
  return train_error,test_error

"""## 机器参数"""

! cat /proc/cpuinfo

"""# 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_

## 运行结果和耗时，avg-time cost是每一折使用的时间
"""

data_X = [ [1]+row[:-1] for row in readin]
data_y = [ [row[-1]] for row in readin]

batch_X,batch_y = into_group(data_X,data_y,5)

names = ["01111","10111","11011","11101","11110"]
import time
```

```
import matplotlib.pyplot as plt
train_errors=[]
test_errors=[]
time_start=time.time()

for musk_test in range(5):
  musk = [1]*5
  musk[musk_test]=0
  train_error,test_error = train_regression(batch_X,batch_y,musk)
  train_errors.append(train_error)
  test_errors.append(test_error)

time_end=time.time()

print('avg-time cost',(time_end-time_start)/5,'s')

plt.plot(train_errors,'-^')
plt.plot(test_errors,'-o')
plt.xticks(range(5),names)

plt.legend(['train_errors','test_errors'])
plt.show()

"""# 用多变量线性模型拟合数据，使用L2正则项，重复上述实验"""

data_X = [ [1]+row[:-1] for row in readin]
data_y = [ [row[-1]] for row in readin]

batch_X,batch_y = into_group(data_X,data_y,5)

names = ["01111","10111","11011","11101","11110"]

import time
import matplotlib.pyplot as plt
train_errors=[]
test_errors=[]
time_start=time.time()
l2s = [0.0,0.1,0.3,0.5,0.7,0.9]
train_errors_in_l2=[]
test_errors_in_l2 = []
for l2 in l2s:
    train_errors=[]
    test_errors=[]
    for musk_test in range(5):
        musk = [1]*5
        musk[musk_test]=0
        train_error,test_error = train_regression_L2(batch_X,batch_y,l2,musk)
        train_errors.append(train_error)
        test_errors.append(test_error)
    train_errors_in_l2.append(train_errors)
    test_errors_in_l2.append(test_errors)

time_end=time.time()

print('avg-time cost',(time_end-time_start)/(5*len(l2s)),'s')

fig = plt.figure(figsize=(20, 8))
for i in range(1, 7):
    index = i -1
    ax = fig.add_subplot(2, 3, i)
```

```
    ax.plot(train_errors_in_l2[index],'-^')
    ax.plot(test_errors_in_l2[index],'-o')
    ax.set_xticks(range(5),names)
    ax.set_title("l2 = {}".format(l2s[index]))
    ax.legend(['train_errors','test_errors'])


plt.show()

fig = plt.figure(figsize=(15, 6))
for i in range(1,7):
    index = i-1
    plt.plot(train_errors_in_l2[index],'-o')
    plt.legend(['l2 = {}'.format(n) for n in l2s] )

plt.xticks(range(5),names)

plt.title('train_errors in l2={}'.format(l2s))

plt.show()

fig = plt.figure(figsize=(15, 6))
for i in range(1,7):
    index = i-1
    plt.plot(test_errors_in_l2[index],'-o')
    plt.legend(['l2 = {}'.format(n) for n in l2s] )

plt.xticks(range(5),names)

plt.title('test_errors in l2={}'.format(l2s))

plt.show()

"""## 检查消耗的内存"""

!pip install memory-profiler

!mprof run --include-children --multiprocess memory_record.py

!mprof plot --flame -o pic.png
```