

**计算机与信息 学院实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验课程： | Python机器学习实验 | | | | |
| 实验编号： | 实验三 | | | | |
| 实验名称： | **FM在分类问题上的应用** | | | | |
| 实验人员： | 学号 | 18111207248 | | | |
| 姓名 | 吴钰 | | | |
| 班级 | 创新班 | | | |
| 实验日期： | 2021.4.7 | | | | |
| 实验室： | 2060301 | | | | |
|  |  | | | | |
| 实验评价： |  | | | | |
| 实验成绩： | |  | 评价日期： |  |
|  | 指导教师： | |  | | |

# 实验目的

掌握FM在分类问题上的应用。

# 实验要求

# 编程实现FM在分类问题上的应用，提交开题报告、训练集和测试集（均为附件）

# 实验内容

FM\_test

import numpy as np

from FM\_train import getPrediction

def loadDataSet(data):

'''导入测试数据集

input: data(string)测试数据

output: dataMat(list)特征

'''

dataMat = []

fr = open(data) # 打开文件

for line in fr.readlines():

lines = line.strip().split("\t")

lineArr = []

for i in range(len(lines)):

lineArr.append(float(lines[i]))

dataMat.append(lineArr)

fr.close()

return dataMat

def loadModel(model\_file):

'''导入FM模型

input: model\_file(string)FM模型

output: w0, np.mat(w).T, np.mat(v)FM模型的参数

'''

f = open(model\_file)

line\_index = 0

w0 = 0.0

w = []

v = []

for line in f.readlines():

lines = line.strip().split("\t")

if line\_index == 0: # w0

w0 = float(lines[0].strip())

elif line\_index == 1: # w

for x in lines:

w.append(float(x.strip()))

else:

v\_tmp = []

for x in lines:

v\_tmp.append(float(x.strip()))

v.append(v\_tmp)

line\_index += 1

f.close()

return w0, np.mat(w).T, np.mat(v)

def save\_result(file\_name, result):

'''保存最终的预测结果

input: file\_name(string)需要保存的文件名

result(mat):对测试数据的预测结果

'''

f = open(file\_name, "w")

f.write("\n".join(str(x) for x in result))

f.close()

if \_\_name\_\_ == "\_\_main\_\_":

# 1、导入测试数据

dataTest = loadDataSet("test\_data.txt")

# 2、导入FM模型

w0, w , v = loadModel("weights")

# 3、预测

result = getPrediction(dataTest, w0, w, v)

# 4、保存最终的预测结果

save\_result("predict\_result", result)

FM\_train

# coding:UTF-8

'''

Date:20160831

@author: zhaozhiyong

'''

import numpy as np

from random import normalvariate # 正态分布

def loadDataSet(data):

'''导入训练数据

input: data(string)训练数据

output: dataMat(list)特征

labelMat(list)标签

'''

dataMat = []

labelMat = []

fr = open(data) # 打开文件

for line in fr.readlines():

lines = line.strip().split("\t")

lineArr = []

for i in range(len(lines) - 1):

lineArr.append(float(lines[i]))

dataMat.append(lineArr)

labelMat.append(float(lines[-1]) \* 2 - 1) # 转换成{-1,1}

fr.close()

return dataMat, labelMat

def sigmoid(inx):

return 1.0 / (1 + np.exp(-inx))

def initialize\_v(n, k):

'''初始化交叉项

input: n(int)特征的个数

k(int)FM模型的超参数

output: v(mat):交叉项的系数权重

'''

v = np.mat(np.zeros((n, k)))

for i in range(n):

for j in range(k):

# 利用正态分布生成每一个权重

v[i, j] = normalvariate(0, 0.2)

return v

def stocGradAscent(dataMatrix, classLabels, k, max\_iter, alpha):

'''利用随机梯度下降法训练FM模型

input: dataMatrix(mat)特征

classLabels(mat)标签

k(int)v的维数

max\_iter(int)最大迭代次数

alpha(float)学习率

output: w0(float),w(mat),v(mat):权重

'''

m, n = np.shape(dataMatrix)

# 1、初始化参数

w = np.zeros((n, 1)) # 其中n是特征的个数

w0 = 0 # 偏置项

v = initialize\_v(n, k) # 初始化V

# 2、训练

for it in range(max\_iter):

for x in range(m): # 随机优化，对每一个样本而言的

inter\_1 = dataMatrix[x] \* v

inter\_2 = np.multiply(dataMatrix[x], dataMatrix[x]) \* \

np.multiply(v, v) # multiply对应元素相乘

# 完成交叉项

interaction = np.sum(np.multiply(inter\_1, inter\_1) - inter\_2) / 2.

p = w0 + dataMatrix[x] \* w + interaction # 计算预测的输出

loss = sigmoid(classLabels[x] \* p[0, 0]) - 1

w0 = w0 - alpha \* loss \* classLabels[x]

for i in range(n):

if dataMatrix[x, i] != 0:

w[i, 0] = w[i, 0] - alpha \* loss \* classLabels[x] \* dataMatrix[x, i]

for j in range(k):

v[i, j] = v[i, j] - alpha \* loss \* classLabels[x] \* \

(dataMatrix[x, i] \* inter\_1[0, j] -\

v[i, j] \* dataMatrix[x, i] \* dataMatrix[x, i])

# 计算损失函数的值

if it % 1000 == 0:

print ("\t------- iter: ", it, " , cost: ", \

getCost(getPrediction(np.mat(dataMatrix), w0, w, v), classLabels))

# 3、返回最终的FM模型的参数

return w0,w,v

def getCost(predict, classLabels):

'''计算预测准确性

input: predict(list)预测值

classLabels(list)标签

output: error(float)计算损失函数的值

'''

m = len(predict)

error = 0.0

for i in range(m):

error -= np.log(sigmoid(predict[i] \* classLabels[i] ))

return error

def getPrediction(dataMatrix, w0, w, v):

'''得到预测值

input: dataMatrix(mat)特征

w(int)常数项权重

w0(int)一次项权重

v(float)交叉项权重

output: result(list)预测的结果

'''

m = np.shape(dataMatrix)[0]

result = []

for x in range(m):

inter\_1 = dataMatrix[x] \* v

inter\_2 = np.multiply(dataMatrix[x], dataMatrix[x]) \* \

np.multiply(v, v) # multiply对应元素相乘

# 完成交叉项

interaction = np.sum(np.multiply(inter\_1, inter\_1) - inter\_2) / 2.

p = w0 + dataMatrix[x] \* w + interaction # 计算预测的输出

pre = sigmoid(p[0, 0])

result.append(pre)

return result

def getAccuracy(predict, classLabels):

'''计算预测准确性

input: predict(list)预测值

classLabels(list)标签

output: float(error) / allItem(float)错误率

'''

m = len(predict)

allItem = 0

error = 0

for i in range(m):

allItem += 1

if float(predict[i]) < 0.5 and classLabels[i] == 1.0:

error += 1

elif float(predict[i]) >= 0.5 and classLabels[i] == -1.0:

error += 1

else:

continue

return float(error) / allItem

def save\_model(file\_name, w0, w, v):

'''保存训练好的FM模型

input: file\_name(string):保存的文件名

w0(float):偏置项

w(mat):一次项的权重

v(mat):交叉项的权重

'''

f = open(file\_name, "w")

# 1、保存w0

f.write(str(w0) + "\n")

# 2、保存一次项的权重

w\_array = []

m = np.shape(w)[0]

for i in range(m):

w\_array.append(str(w[i, 0]))

f.write("\t".join(w\_array) + "\n")

# 3、保存交叉项的权重

m1 , n1 = np.shape(v)

for i in range(m1):

v\_tmp = []

for j in range(n1):

v\_tmp.append(str(v[i, j]))

f.write("\t".join(v\_tmp) + "\n")

f.close()

if \_\_name\_\_ == "\_\_main\_\_":

# 1、导入训练数据

print ("---------- 1.load data ---------")

dataTrain, labelTrain = loadDataSet("D:\\大三（下）专业课\\机器学习实验\\Python-Machine-Learning-Algorithm-master\\Chapter\_3 Factorization Machine\\data.txt")

print ("---------- 2.learning ---------")

# 2、利用随机梯度训练FM模型

w0, w, v = stocGradAscent(np.mat(dataTrain), labelTrain, 3, 10000, 0.01)

predict\_result = getPrediction(np.mat(dataTrain), w0, w, v) # 得到训练的准确性

print( "----------training accuracy: %f" % (1 - getAccuracy(predict\_result, labelTrain)))

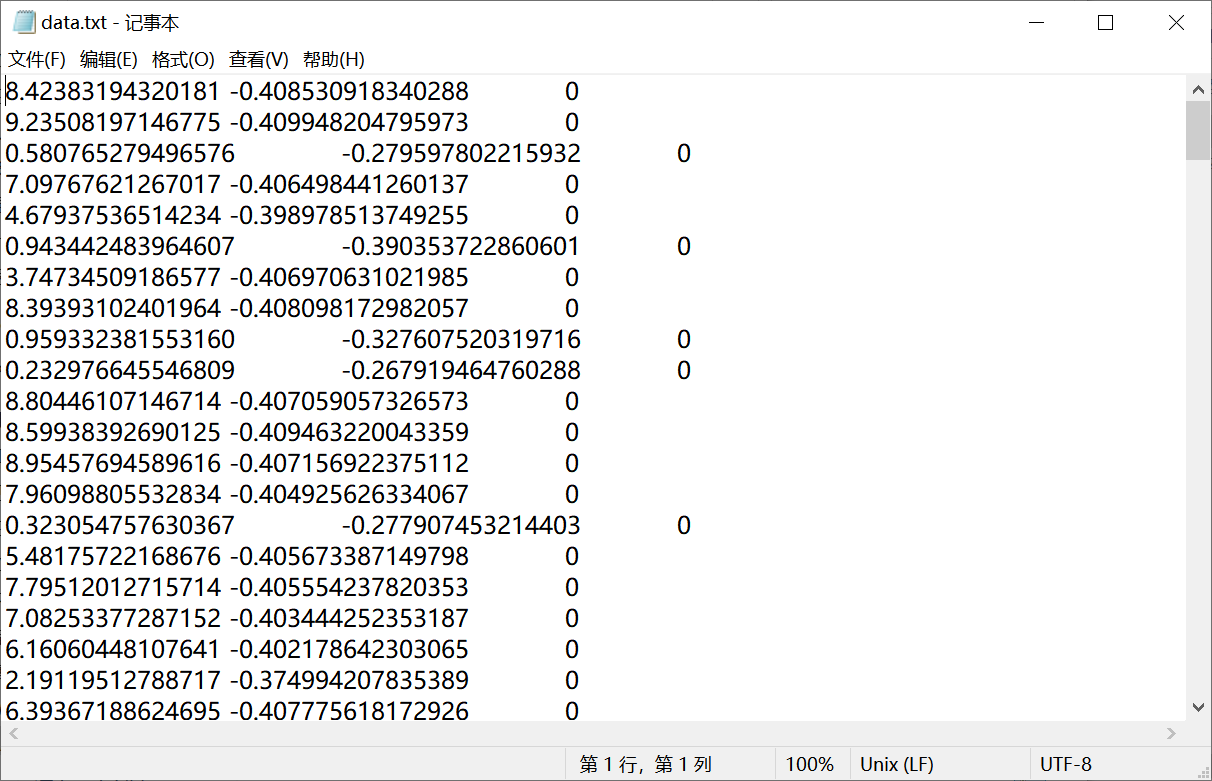
print ("---------- 3.save result ---------")

# 3、保存训练好的FM模型

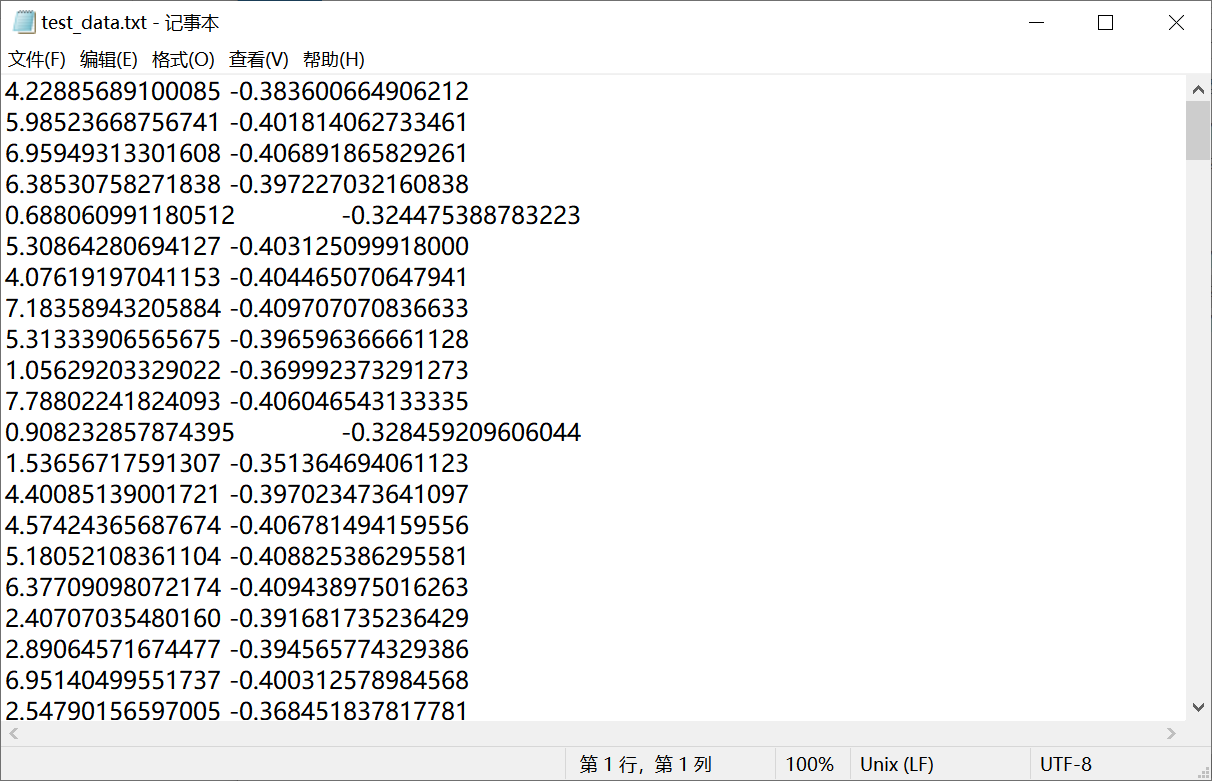
save\_model("weights", w0, w, v)

# 实验过程及结果

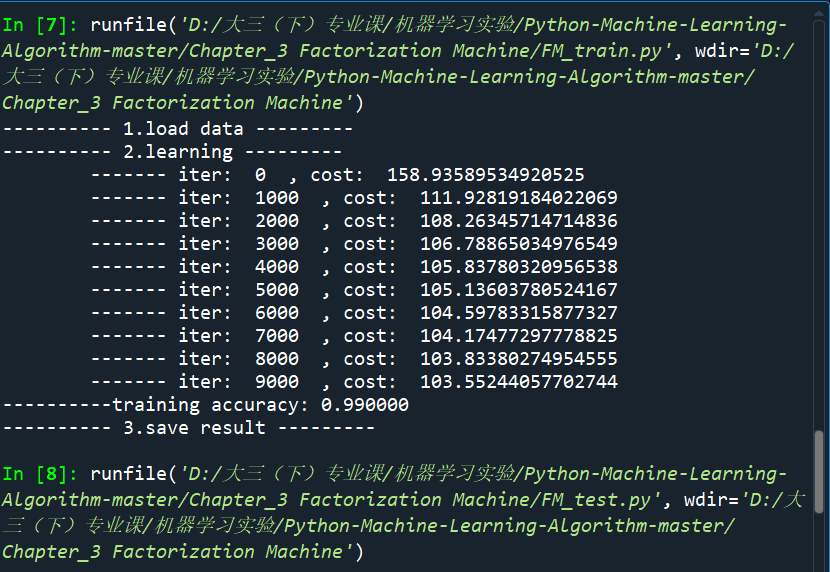
训练集data.txt



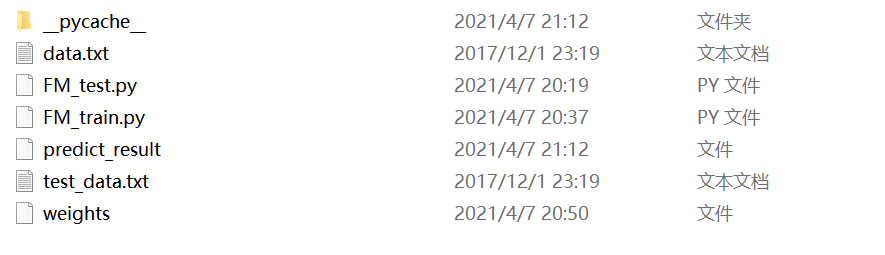
测试集：test\_data.txt



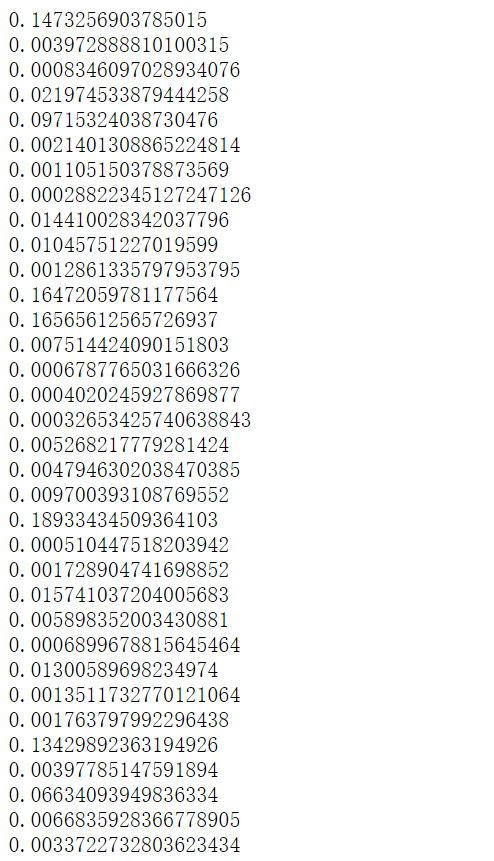
运行截图：



生成文件：



predict\_result



weights

