**本 科 实 验 报 告**

实验名称： LDA的python实现

学 员： 孙源辰 学 号： 201802001067

培养类型： 无军籍本科学员 年 级： 大三

专 业：人工智能与大数据 所属学院： 计算机学院

指导教员： 刘新旺 职 称： 副教授

实 验 室： 305-205 实验日期： 2021春季学期

国防科学技术大学训练部制

《本科实验报告》填写说明

1．学员完成人才培养方案和课程标准要所要求的每个实验后，均须提交实验报告。

2．实验报告封面必须打印，报告内容可以手写或打印。

3．实验报告内容编排及打印应符合以下要求：

（1）采用A4（21cm×29.7cm）白色复印纸，单面黑字打印。上下左右各侧的页边距均为3cm；缺省文档网格：字号为小4号，中文为宋体，英文和阿拉伯数字为Times New Roman，每页30行，每行36字；页脚距边界为2.5cm，页码置于页脚、居中，采用小5号阿拉伯数字从1开始连续编排，封面不编页码。

（2）报告正文最多可设四级标题，字体均为黑体，第一级标题字号为4号，其余各级标题为小4号；标题序号第一级用“一、”、“二、”……，第二级用“（一）”、“（二）” ……，第三级用“1.”、“2.” ……，第四级用“（1）”、“（2）” ……，分别按序连续编排。

（3）正文插图、表格中的文字字号均为5号。

一、实验目的和要求

通过代码实现，了解并掌握IDA常用算法，从而更加深入的理解线性模型。

二、实验内容和原理

通过python对LDA算法进行实现并使其可视化。

三、操作方法与实验步骤

（1）进行数据集的生成

（2）对数据计算协方差矩阵

（3）对协方差矩阵求类内散度矩阵

（4）计算两类的均值和方差（虽然已经在生成里面定义了）

（5）对类内散度矩阵分解并求逆

（6）计算ω

（7）用训练出的LDA进行分类预测并检测性能

（8）可视化

四、实验结果及分析

1.代码及运行结果如下

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

from sklearn import datasets

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.pipeline import make\_pipeline

class LDA():

def \_\_init\_\_(self):

self.omega = None

def calculate\_covariance\_matrix(self, X, Y=None):#协方差阵计算

m = X.shape[0]

X = X - np.mean(X, axis=0)

Y = X if Y == None else Y - np.mean(Y, axis=0)

return 1 / m \* np.matmul(X.T, Y)

def fit(self,X,y):#前几个为PPT中列出的拟合必要的参数

#按照之前的label标准对数据进行分类

X0 = X[y.reshape(-1) == 0]

X1 = X[y.reshape(-1) == 1]

#协方差矩阵计算

Sigma0 = self.calculate\_covariance\_matrix(X0)

Sigma1 = self.calculate\_covariance\_matrix(X1)

#类内散度矩阵计算

Sw = Sigma0 + Sigma1

#均值方差计算

miu0, miu1 = X0.mean(0), X1.mean(0)

mean\_diff = np.atleast\_1d(miu0 - miu1)

#奇异值分解

U, S, V = np.linalg.svd(Sw)

#Sw的逆

Sw\_ = np.dot(np.dot(V, np.linalg.pinv(np.diag(S))), U.T)#求逆真的没有函数

self.omega = Sw\_.dot(mean\_diff)

return self.omega

def predict(self,X):

y\_pred = []

for matrix in X:

h = matrix.dot(self.omega)

y = 1 \* (h<0)#这个也太巧了吧

y\_pred.append(y)

return y\_pred

def class\_visu(self,X, y):

X1 = np.array([X[i] for i in range(len(X)) if y[i] == 0])

X2 = np.array([X[i] for i in range(len(X)) if y[i] == 1])

X1\_new = np.dot(X1, self.omega)

X2\_new = np.dot(X2, self.omega)

y1\_new = [1 for i in range(len(X1))]

y2\_new = [1 for i in range(len(X2))]

return X1\_new, X2\_new, y1\_new, y2\_new

dataset\_iris = datasets.load\_iris()

X = dataset\_iris.data

y = dataset\_iris.target

dataset\_normalizer = StandardScaler().fit(X)

X = dataset\_normalizer.transform(X)

x\_train,x\_test,y\_train,y\_test=train\_test\_split(X, y, random\_state=1)

x\_test\_0 = x\_test[y\_test == 0]

x\_test\_1 = x\_test[y\_test == 1]

x\_test\_2 = x\_test[y\_test == 2]

unique\_targets = np.unique(y, return\_index=True, return\_counts=True)

def OvR\_class(x\_train, y\_train):

models = {}

y\_train\_copy = y\_train.copy()

unique\_targets = np.unique(y\_train\_copy, return\_index=True, return\_counts=True)

for target in unique\_targets[0]:

models[target] = LDA()

y\_train\_list = y\_train\_copy.tolist()

for i in range(len(y\_train\_list)):

if y\_train\_list[i] == target:

y\_train\_list[i] = 1

else:

y\_train\_list[i] = 0

y\_train = np.array(y\_train\_list)

models[target].fit(x\_train, y\_train)

return models

test\_probs = {}

test\_class = {}

models = OvR\_class(x\_train, y\_train)

for target in unique\_targets[0]:

test\_probs[target] = models[target].predict(x\_test)

test\_class[target] = sum(test\_probs[target])

fig = plt.figure()

ax = Axes3D(fig)

fig = plt.figure()

ax = Axes3D(fig)

x = np.array(test\_probs[0])

y = np.array(test\_probs[1])

z = np.array(test\_probs[2])

ax.scatter(x, y, z)

plt.show()

1. 结果分析

可以看到，模型可以比较好的进行分类。