# 第十章源代码

**#<程序：运用最小二乘分类器对糖尿病数据进行分类>**

import csv

import numpy as np

from sklearn import metrics

TRAIN\_SET\_NUM = 232

trans\_table = {"Yes":1,"No":-1}

def arr\_float(data):

out = []

for elem in data:

out.append(float(elem))

return out

def read\_data():

with open('Pima.te.csv') as csvfile:

csv\_reader = csv.reader(csvfile)

tx = [] ; ty = [] ;vx = [];vy = [] ; counter = 0

for row in csv\_reader:

if counter == 0 : # 跳过表头

counter += 1

continue

if counter > TRAIN\_SET\_NUM :

vx.append(arr\_float(row[1:8]))

vy.append(trans\_table[row[8]])

else:

tx.append(arr\_float(row[1:8]))

ty.append(trans\_table[row[8]])

counter += 1

return tx,ty,vx,vy

tx,ty,vx,vy = read\_data()

#分类器训练

L = np.array([ty]) ; L = L.T #将ty从list转换为矩阵并转置，构造列向量L

A = []

for i in range(TRAIN\_SET\_NUM):

A.append(tx[i])

A[-1].insert(0,1)

A = np.array(A) ; A\_T = A.T #将A从list转化为二维矩阵A.T代表A的转置

# 利用公式计算参数w’ （在程序里面为w）

temp = np.linalg.inv(A\_T.dot(A)) # dot表示矩阵乘法，linalg.inv表示求逆

temp = temp.dot(A\_T) ; u = temp.dot(L)

#分类器测试

predicted = []

for sample in vx:

sample.insert(0,1) ;sp = np.array([sample])

value = sp.dot(u)

if value > 0:

predicted.append(1)

else:

predicted.append(-1)

print(metrics.confusion\_matrix(vy, predicted)) # 输出分类结果的矩阵

print(metrics.classification\_report(vy, predicted)) # 输出分类性能指标

###############################################################################

**# <程序：运用Logistic分类器对糖尿病数据进行分类>**

import csv

from sklearn import metrics

from sklearn.linear\_model import LogisticRegression

TRAIN\_SET\_NUM = 232

trans\_table = {"Yes":1,"No":0}

def arr\_float(data):

out = []

for elem in data:

out.append(float(elem))

return out

def read\_data():

with open('Pima.te.csv') as csvfile:

csv\_reader = csv.reader(csvfile)

tx = [] ; ty = [] ;vx = [];vy = [] ; counter = 0

for row in csv\_reader:

if counter == 0 : # 跳过表头

counter += 1

continue

if counter > TRAIN\_SET\_NUM :

vx.append(arr\_float(row[1:8]))

vy.append(trans\_table[row[8]])

else:

tx.append(arr\_float(row[1:8]))

ty.append(trans\_table[row[8]])

counter += 1

return tx,ty,vx,vy

tx,ty,vx,vy = read\_data()

# 分类器训练和测试

model = LogisticRegression() ; model.fit(tx, ty)

predicted = model.predict(vx)

print(metrics.confusion\_matrix(vy, predicted))

print(metrics.classification\_report(vy, predicted))

###############################################################################

**# <程序：运用朴素贝叶斯分类器对糖尿病数据进行分类>**

import csv

from sklearn import metrics

from sklearn.naive\_bayes import GaussianNB #更改的地方 1

TRAIN\_SET\_NUM = 232

trans\_table = {"Yes":1,"No":0}

def arr\_float(data):

out = []

for elem in data:

out.append(float(elem))

return out

def read\_data():

with open('Pima.te.csv') as csvfile:

csv\_reader = csv.reader(csvfile)

tx = [] ; ty = [] ;vx = [];vy = [] ; counter = 0

for row in csv\_reader:

if counter == 0 : # 跳过表头

counter += 1

continue

if counter > TRAIN\_SET\_NUM :

vx.append(arr\_float(row[1:8]))

vy.append(trans\_table[row[8]])

else:

tx.append(arr\_float(row[1:8]))

ty.append(trans\_table[row[8]])

counter += 1

return tx,ty,vx,vy

tx,ty,vx,vy = read\_data()

# 分类器训练和测试

model = GaussianNB () ; model.fit(tx, ty) # 更改的地方 2

predicted = model.predict(vx)

print(metrics.confusion\_matrix(vy, predicted))

print(metrics.classification\_report(vy, predicted))

###############################################################################

**#<程序：运用Bayes分类器识别手写体数字>**

from copy import deepcopy

from sklearn import metrics

from sklearn.naive\_bayes import BernoulliNB

TRAIN\_NUM=10000;VERIFY\_NUM=1000

def read\_file\_data(fname,count,size,offset):

f = open(fname,'rb') ; filedata = f.read()

f.close() ; fdata = bytearray(filedata)

ret = [] ; cur = offset

for i in range(0,count):

if size == 1:

ret.append(fdata[cur])

cur+=1 ; continue

temp = []

for r in range(0,size):

temp.append(fdata[cur]);cur+=1

ret.append(deepcopy(temp))

return ret

tx=read\_file\_data("./MNIST/train-images.idx3-ubyte",TRAIN\_NUM,28\*28,16)

ty=read\_file\_data("./MNIST/train-labels.idx1-ubyte",TRAIN\_NUM,1,8)

vx=read\_file\_data("./MNIST/t10k-images.idx3-ubyte",VERIFY\_NUM,28\*28,16)

vy=read\_file\_data("./MNIST/t10k-labels.idx1-ubyte",VERIFY\_NUM,1,8)

model = BernoulliNB(); model.fit(tx,ty)

predicted = model.predict(vx)

print(metrics.classification\_report(vy, predicted))

###############################################################################

**#<程序：用感知器判断是否患病>**

from numpy import exp, array, random, dot

train\_set = array([[0, 0, 1], [1, 1, 1], [1, 0, 1], [0, 1, 1]]).T

label\_set = array([[0, 1, 1, 0]]).T

weight = random.random((3, 1))

# train

for i in range(200):

out = 1 / (1+exp(-(dot(train\_set.T, weight))))

weight += dot(train\_set, (label\_set - out) \* out \* (1 - out))

# classify

new\_sample = array([1, 0, 0])

print('possibility:', 1.0 / float(1 + exp(-dot(new\_sample, weight))))

###############################################################################

**#<程序：利用深度学习进行手写数字识别 p1，p2>**

from keras.models import Sequential

from keras.layers import Dense, Dropout, Flatten

from keras.layers.convolutional import Conv2D, MaxPooling2D

from keras.datasets import mnist

from keras.utils import np\_utils

from keras.optimizers import Adadelta

batch\_size = 128

num\_classes = 10

epochs = 10

(X\_train, Y\_train), (X\_test, Y\_test) = mnist.load\_data()

X\_train = X\_train.reshape(X\_train.shape[0], 28, 28 ,1).astype('float32')

X\_test = X\_test.reshape(X\_test.shape[0], 28, 28, 1).astype('float32')

X\_train /= 255

X\_test /= 255

Y\_train = np\_utils.to\_categorical(Y\_train, num\_classes)

Y\_test = np\_utils.to\_categorical(Y\_test, num\_classes )

model = Sequential()

model.add(Conv2D(filters=64, kernel\_size=(3, 3), activation='relu', input\_shape=(28, 28, 1)))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Conv2D(filters=64, kernel\_size=(3, 3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.5))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dense(num\_classes, activation='softmax'))

model.compile(loss='categorical\_crossentropy', optimizer=Adadelta(), metrics=['accuracy'])

model.fit(X\_train, Y\_train, batch\_size=batch\_size, epochs= epochs, verbose=1, validation\_data=(X\_test,Y\_test))

score = model.evaluate(X\_test, Y\_test, verbose=0)

print('Test loss:', score[0])

print('Test accuracy:', score[1])

###############################################################################

**#<程序：利用深度学习进行手写数字识别 p3>**

import numpy as np

import skimage.io

img = skimage.io.imread(('/path/\*.jpg'),as\_gray = True)

img = np.reshape(img, (1, 28, 28, 1)).astype('float32')

proba = model.predict\_proba(img, verbose=0)

result = model.predict\_classes(img, verbose=0)

print(proba[0])

print(result[0])

**运行此程序时，需要把上面模型的代码加入，整体代码如下：**

**#注意：自己手写的图片需要处理成大小为28\*28的图片，并放在当前文件夹下。**

**#（不需要手动将图片进行灰度处理，只需要修改图片大小即可）**

#<程序：利用深度学习进行手写数字识别 p3>

import numpy as np

import skimage.io

from keras.models import Sequential

from keras.layers import Dense, Dropout, Flatten

from keras.layers.convolutional import Conv2D, MaxPooling2D

from keras.datasets import mnist

from keras.utils import np\_utils

from keras.optimizers import Adadelta

batch\_size = 128

num\_classes = 10

epochs = 10

(X\_train, Y\_train), (X\_test, Y\_test) = mnist.load\_data()

X\_train = X\_train.reshape(X\_train.shape[0], 28, 28 ,1).astype('float32')

X\_test = X\_test.reshape(X\_test.shape[0], 28, 28, 1).astype('float32')

X\_train /= 255

X\_test /= 255

Y\_train = np\_utils.to\_categorical(Y\_train, num\_classes)

Y\_test = np\_utils.to\_categorical(Y\_test, num\_classes )

model = Sequential()

model.add(Conv2D(filters=64, kernel\_size=(3, 3), activation='relu', input\_shape=(28, 28, 1)))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Conv2D(filters=64, kernel\_size=(3, 3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.5))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dense(num\_classes, activation='softmax'))

model.compile(loss='categorical\_crossentropy', optimizer=Adadelta(), metrics=['accuracy'])

model.fit(X\_train,Y\_train,batch\_size=batch\_size,epochs=epochs,verbose=1,validation\_data=(X\_test,Y\_test))

#score = model.evaluate(X\_test, Y\_test, verbose=0)

#print('Test loss:', score[0])

#print('Test accuracy:', score[1])

img = skimage.io.imread(('fig1.jpg'),as\_gray = True)

img = np.reshape(img, (1, 28, 28, 1)).astype('float32')

proba = model.predict\_proba(img, verbose=0)

result = model.predict\_classes(img, verbose=0)

print(proba[0])

print(result[0])