**1**、

**import** keras

**from** dataprocess **import** get\_data\_one

**from** keras.models **import** Sequential,Model,Input

**from** keras.callbacks **import** TensorBoard

**from** keras.layers **import** Dense

**from** keras.metrics **import** \*

**from** keras.losses **import** mse

**def** getmodel(inunit,outunit):

x=Input(shape=(10,))

o=Dense(outunit,activation=**None**)(x)

model=Model(x,o,name=**'model'**)

**return** model

model=getmodel(10,1)

ten=TensorBoard(log\_dir=**'./xxxx'**)

model.compile(optimizer=**'sgd'**,loss=mse)

x,y=get\_data\_one(**'/home/cooper/PycharmProjects/sxyl/Assignments/diabetes.txt'**)

model.fit(x,y,

batch\_size=32,

epochs=1000,

callbacks=[ten],

validation\_split=0.2)

2、**from** sklearn.linear\_model **import** LogisticRegressionCV,LogisticRegression

**from** sklearn.preprocessing **import** StandardScaler

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

**from** dataprocess **import** get\_data\_two

**from** sklearn.metrics **import** precision\_recall\_curve,precision\_score,recall\_score,f1\_score

**import** numpy **as** np

stand=StandardScaler()

x,y=get\_data\_two(**'/home/cooper/PycharmProjects/sxyl/Assignments/iris2.txt'**)

x=stand.fit\_transform(X=x)

print(x.shape)

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2)

print(x\_train.shape,x\_test.shape,y\_train.shape,y\_test.shape)

**import** keras

**from** dataprocess **import** get\_data\_one

**from** keras.models **import** Sequential,Model,Input

**from** keras.callbacks **import** TensorBoard

**from** keras.layers **import** Dense

**from** keras.metrics **import** \*

**from** keras.losses **import** mse,binary\_crossentropy

**def** getmodel(inunit,outunit):

x=Input(shape=(inunit,))

o=Dense(outunit,activation=**'sigmoid'**)(x)

model=Model(x,o,name=**'model'**)

**return** model

model=getmodel(4,1)

ten=TensorBoard(log\_dir=**'./sigmoid'**,write\_graph=**True**,write\_grads=**True**,write\_images=**True**)

model.compile(optimizer=**'sgd'**,loss=binary\_crossentropy,metrics=[binary\_accuracy])

model.fit(x,y,

batch\_size=32,

epochs=1000,

callbacks=[ten],

validation\_split=0.2)

3、

**from** sklearn.linear\_model **import** LogisticRegressionCV,LogisticRegression,SGDClassifier

**from** sklearn.preprocessing **import** StandardScaler,OneHotEncoder

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

**from** dataprocess **import** get\_data\_two,get\_data\_three

**from** sklearn.metrics **import** precision\_recall\_curve,precision\_score,recall\_score,f1\_score

**import** numpy **as** np

**import** keras

**from** dataprocess **import** get\_data\_one

**from** keras.models **import** Sequential,Model,Input

**from** keras.callbacks **import** TensorBoard,ModelCheckpoint

**from** keras.layers **import** Dense

**from** keras.metrics **import** \*

**from** keras.losses **import** mse,categorical\_crossentropy

X,Y=get\_data\_three(**'/home/cooper/PycharmProjects/sxyl/Assignments/iris3.txt'**)

stand=StandardScaler()

X=stand.fit\_transform(X)

*# onehot=OneHotEncoder()*

*# Y=onehot.fit\_transform(Y)*

print(X.shape,Y.shape)

*# Y=np.asarray(Y,np.int)*

YY=np.zeros([150,3],np.int64)

**for** i **in** range(Y.shape[0]):

YY[i,Y[i]]=1

print(X.shape,YY.shape)

x\_train,x\_test,y\_train,y\_test=train\_test\_split(X,YY,test\_size=0.2,random\_state=1)

print(x\_train.shape,x\_test.shape,y\_train.shape,y\_test.shape)

**def** getmodel(inunit,outunit):

x=Input(shape=(4,))

o=Dense(outunit,activation=**'softmax'**)(x)

model=Model(x,o,name=**'model'**)

**return** model

model=getmodel(4,3)

ten=TensorBoard()

model.compile(optimizer=**'sgd'**,loss=categorical\_crossentropy,metrics=[categorical\_accuracy])

*# x,y=get\_data\_one('/home/cooper/PycharmProjects/sxyl/Assignments/diabetes.txt')*

model.fit(x\_train,y\_train,

batch\_size=32,

epochs=10000,

callbacks=[ten],validation\_data=[x\_test,y\_test])

model.save\_weights(**'3\_weight\_model.hdf5'**)

*# model.evaluate(x\_test,y\_test)*

*4*、

*# encoding:utf-8*

**import** numpy **as** np

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

**from** sklearn.datasets **import** load\_breast\_cancer

**from** sklearn.preprocessing **import** MinMaxScaler

**from** sklearn.metrics **import** accuracy\_score

**from** sklearn.linear\_model **import** LogisticRegression

**import** matplotlib **as** mpl

**import** matplotlib.pyplot **as** plt

**from** sklearn.linear\_model **import** LogisticRegressionCV,LogisticRegression,SGDClassifier

**from** sklearn.preprocessing **import** StandardScaler,OneHotEncoder

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

**from** dataprocess **import** get\_data\_two,get\_data\_two

**from** sklearn.metrics **import** precision\_recall\_curve,precision\_score,recall\_score,f1\_score

**import** keras

**from** dataprocess **import** get\_data\_one

**from** keras.models **import** Sequential,Model,Input

**from** keras.callbacks **import** TensorBoard,ModelCheckpoint

**from** keras.layers **import** Dense

**from** keras.metrics **import** \*

**from** keras.losses **import** mse,categorical\_crossentropy

stand=StandardScaler()

x,y=get\_data\_two(**'/home/cooper/PycharmProjects/sxyl/Assignments/iris2.txt'**)

x=stand.fit\_transform(X=x)

print(x.shape)

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2)

print(x\_train.shape,x\_test.shape,y\_train.shape,y\_test.shape)

**class** GDA:

**def** \_\_init\_\_(self,train\_data,train\_label):

self.Train\_Data = train\_data

self.Train\_Label = train\_label

self.postive\_num = 0 *# 正样本个数*

self.negetive\_num = 0 *# 负样本个数*

postive\_data = [] *# 正样本数组*

negetive\_data = [] *# 负样本数组*

**for** (data,label) **in** zip(self.Train\_Data,self.Train\_Label):

**if** label == 1: *# 正样本*

self.postive\_num += 1

postive\_data.append(list(data))

**else**: *# 负样本*

self.negetive\_num += 1

negetive\_data.append(list(data))

*# 计算正负样本的二项分布的概率*

row,col = np.shape(train\_data)

self.postive = self.postive\_num\*1.0/row *# 正样本的二项分布概率*

self.negetive = 1-self.postive *# 负样本的二项分布概率*

*# 计算正负样本的高斯分布的均值向量*

postive\_data = np.array(postive\_data)

negetive\_data = np.array(negetive\_data)

postive\_data\_sum = np.sum(postive\_data, 0)

negetive\_data\_sum = np.sum(negetive\_data, 0)

self.mu\_positive = postive\_data\_sum\*1.0/self.postive\_num *# 正样本的高斯分布的均值向量*

self.mu\_negetive = negetive\_data\_sum\*1.0/self.negetive\_num *# 负样本的高斯分布的均值向量*

*# 计算高斯分布的协方差矩阵*

positive\_deta = postive\_data-self.mu\_positive

negetive\_deta = negetive\_data-self.mu\_negetive

self.sigma = []

**for** deta **in** positive\_deta:

deta = deta.reshape(1,col)

ans = deta.T.dot(deta)

self.sigma.append(ans)

**for** deta **in** negetive\_deta:

deta = deta.reshape(1,col)

ans = deta.T.dot(deta)

self.sigma.append(ans)

self.sigma = np.array(self.sigma)

*#print(np.shape(self.sigma))*

self.sigma = np.sum(self.sigma,0)

self.sigma = self.sigma/row

self.mu\_positive = self.mu\_positive.reshape(1,col)

self.mu\_negetive = self.mu\_negetive.reshape(1,col)

**def** Gaussian(self, x, mean, cov):

*"""*

*这是自定义的高斯分布概率密度函数*

**:param** *x: 输入数据*

**:param** *mean: 均值向量*

**:param** *cov: 协方差矩阵*

**:return***: x的概率*

*"""*

dim = np.shape(cov)[0]

*# cov的行列式为零时的措施*

covdet = np.linalg.det(cov + np.eye(dim) \* 0.001)

covinv = np.linalg.inv(cov + np.eye(dim) \* 0.001)

xdiff = (x - mean).reshape((1, dim))

*# 概率密度*

prob = 1.0 / (np.power(np.power(2 \* np.pi, dim) \* np.abs(covdet), 0.5)) \* \

np.exp(-0.5 \* xdiff.dot(covinv).dot(xdiff.T))[0][0]

**return** prob

**def** predict(self,test\_data):

predict\_label = []

**for** data **in** test\_data:

positive\_pro = self.Gaussian(data,self.mu\_positive,self.sigma)

negetive\_pro = self.Gaussian(data,self.mu\_negetive,self.sigma)

**if** positive\_pro >= negetive\_pro:

predict\_label.append(1)

**else**:

predict\_label.append(0)

**return** predict\_label

**def** run\_main():

*"""*

*这是主函数*

*"""*

*# 导入乳腺癌数据*

breast\_cancer = load\_breast\_cancer()

data = np.array(breast\_cancer.data)

label = np.array(breast\_cancer.target)

print(data.shape,label.shape)

**if** \_\_name\_\_==**'\_\_main\_\_'**:

gda=GDA(x\_train,y\_train)

y\_pre=gda.predict(x\_test)

print(**'recall: '**, recall\_score(y\_test, y\_pre))

print(**'precision'**, precision\_score(y\_test, y\_pre))

print(**'f1\_score'**, f1\_score(y\_test, y\_pre))

数据预处理代码：

*# coding:utf-8*

**import** numpy **as** np

**def** get\_data\_one(path):

X=[]

Y=[]

**with** open(path) **as** f:

**for** line **in** f:

words=line.split()

x=[float(i) **for** i **in** words[:-1]]

y=float(words[-1])

X.append(x)

Y.append(y)

X=np.array(X)

Y=np.array(Y)

Y=Y[:,np.newaxis]

print(X.shape,Y.shape)

**return** X,Y

**def** get\_data\_two(path):

X=[]

Y=[]

**with** open(path) **as** f:

**for** line **in** f:

words=line.split()

**if not** len(words)==5:

**break**

x=[float(i) **for** i **in** words[:-1]]

y=float(words[4])

*# print(x,y)*

X.append(x)

Y.append(y)

X=np.array(X,np.float32)

Y=np.array(Y,np.int)

Y=Y[:,np.newaxis]

print(X.shape,Y.shape)

**return** X,Y

**def** get\_data\_three(path):

X=[]

Y=[]

**with** open(path) **as** f:

**for** line **in** f:

words=line.split()

**if not** len(words)==5:

**break**

x=[float(i) **for** i **in** words[:-1]]

y=float(words[-1])

X.append(x)

Y.append(y)

X=np.array(X,np.float32)

Y=np.array(Y,np.int)

Y=Y[:,np.newaxis]

print(X.shape,Y.shape)

**return** X,Y

**if** \_\_name\_\_==**'\_\_main\_\_'**:

*# x,y=get\_data\_one('/home/cooper/PycharmProjects/sxyl/Assignments/diabetes.txt')*

x,y=get\_data\_three(**'/home/cooper/PycharmProjects/sxyl/Assignments/iris3.txt'**)

详细代码见：github：<https://github.com/xkp793003821/sxyl>