from \_\_future\_\_ import print\_function

import numpy as np

import math

import struct

import sys

import utilities as util

from random import random

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

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score

from sklearn.metrics import f1\_score

def getdata():

trainsize = 5000

testsize = 1000

folder = ""

# read training labels

with open(folder + "train-labels.idx1-ubyte", "rb") as f:

data = f.read(4) # magic number

data = f.read(4) # number of samples

training\_samples = struct.unpack('>HH',data)[1]

traininglabels = np.empty(trainsize)

for i in range(0,trainsize):

data = f.read(1)

traininglabels[i] = int(struct.unpack('>B',data)[0])

# read training images

with open(folder + "train-images.idx3-ubyte", "rb") as f:

data = f.read(4) # magic number

data = f.read(4) # number of samples

trainingsample = np.zeros(784)

trainingsamples = np.empty((trainsize,196))

data = f.read(4) # y dimension

data = f.read(4) # x dimension

for i in range(0,trainsize):

for j in range(0,784):

data = f.read(1)

trainingsample[j] = util.round(struct.unpack('>B',data)[0])

trainingsamples[i] = util.subsample(util.resize(trainingsample), 14)

# read test labels

with open(folder + "t10k-labels.idx1-ubyte", "rb") as f:

data = f.read(4) # magic number

data = f.read(4) # number of samples

test\_samples = struct.unpack('>HH',data)[1]

testlabels = np.empty(testsize)

for i in range(0,testsize):

data = f.read(1)

testlabels[i] = int(struct.unpack('>B',data)[0])

# read test images

with open(folder + "t10k-images.idx3-ubyte", "rb") as f:

data = f.read(4) # magic number

data = f.read(4) # number of samples

testsample = np.zeros(784)

testsamples = np.empty((testsize,196))

data = f.read(4) # y dimension

data = f.read(4) # x dimension

for i in range(0,testsize):

for j in range(0,784):

data = f.read(1)

testsample[j] = util.round(struct.unpack('>B',data)[0])

testsamples[i] = util.subsample(util.resize(testsample), 14)

y\_train = traininglabels

X\_train = trainingsamples

y\_test = testlabels

X\_test = testsamples

return X\_train, X\_test, y\_train, y\_test

# return X, y

# def run(X, y):

def run(X\_train, X\_test, y\_train, y\_test, epochs, gam, C\_pen):

# X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42, shuffle=True)

#epochs

clfs\_o = list()

acc = np.zeros(10)

#classifiers

for k in range(10):

clfs\_o.append(SVC(gamma=gam))

for p in range (epochs):

r\_X\_train, r\_y\_train = util.shuffle(X\_train, y\_train, 1000)

r\_X\_test, r\_y\_test = util.shuffle(X\_test, y\_test, 200)

X\_train\_folds = np.empty((800,196))

y\_train\_folds = np.empty(800)

X\_test\_fold = np.empty((200,196))

y\_test\_fold = np.empty(200)

k\_scores = np.zeros(10)

#folds

for i in range(5):

flag = False

for k in range(5):

if k == i:

X\_test\_fold = r\_X\_train[k\*200:k\*200 + 200]

y\_test\_fold = r\_y\_train[k\*200:k\*200 + 200]

else:

if flag == False:

X\_train\_folds = r\_X\_train[k\*200:k\*200 + 200]

y\_train\_folds = r\_y\_train[k\*200:k\*200 + 200]

flag = True

else:

X\_train\_folds = np.concatenate((X\_train\_folds, r\_X\_train[k\*200:k\*200 + 200]), axis= 0)

y\_train\_folds = np.concatenate((y\_train\_folds, r\_y\_train[k\*200:k\*200 + 200]), axis= 0)

clfs = list()

#train and test classifiers

for k in range(10):

clfs.append(SVC(gamma=gam, C=C\_pen))

for k in range(10):

y\_e\_train\_folds = np.empty(800)

for j in range (len(y\_train\_folds)):

if y\_train\_folds[j] == k:

y\_e\_train\_folds[j] = 1

else:

y\_e\_train\_folds[j] = -1

clfs[k].fit(X\_train\_folds, y\_e\_train\_folds)

pred = clfs[k].predict(X\_test\_fold)

#print(clfs[k].get\_params())

y\_e\_test\_fold = np.empty(200)

for j in range (len(y\_test\_fold)):

if y\_test\_fold[j] == k:

y\_e\_test\_fold[j] = 1

else:

y\_e\_test\_fold[j] = -1

tp = fp = tn = fn = 0.0

for j in range (len(y\_test\_fold)):

if pred[j] == y\_e\_test\_fold[j]:

if pred[j] == 1:

tp += 1

else:

tn += 1

else:

if pred[j] == 1:

fp += 1

else:

fn +=1

k\_scores[k] += (tn + tp) /len(y\_test\_fold)

#to do precision, recall etc

precision = tp / (tp + fp)

recall = tp/(tp + fn)

k\_scores = k\_scores/5.0

for i in range(10):

if k\_scores[i] > acc[i]:

acc[i] = k\_scores[i]

clfs\_o[i] = clfs[i]

print(k\_scores)

# acc = accuracy\_score(y\_test, pred, normalize=True)

# acc = f1\_score(y\_test, pred, average="weighted")

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

X\_train, X\_test, y\_train, y\_test = getdata()

run(X\_train, X\_test, y\_train, y\_test, 2, 0.01, 0.5)