# wine.py

import numpy as np   
from python3.plotDecBoundaries import plotDecBoundaries  
from tools.nearest\_centroid\_classifier import \*  
  
  
  
train\_data = np.genfromtxt('python3/wine\_train.csv', delimiter = ',')  
test\_data = np.genfromtxt('python3/wine\_test.csv' , delimiter = ',')  
  
train\_data\_labels = []  
test\_data\_labels = []  
  
train\_data\_labels= getLabels(train\_data)  
test\_data\_labels= getLabels(test\_data)  
print(train\_data\_labels)  
#print(train\_data\_labels)  
train\_error\_set\_for\_features = []  
test\_error\_set\_for\_features = []  
feature1 = 0  
feature2 = 1  
mean\_set\_label1 = train\_classifer\_label(train\_data,1,feature1,feature2)  
mean\_set\_label2 = train\_classifer\_label(train\_data,2,feature1,feature2)  
mean\_set\_label3 = train\_classifer\_label(train\_data,3,feature1,feature2)  
sample\_mean\_set\_unlabelled = np.array([mean\_set\_label1,mean\_set\_label2,mean\_set\_label3])  
mean\_set\_label1 = np.append(mean\_set\_label1,1)  
mean\_set\_label2 = np.append(mean\_set\_label2,2)  
mean\_set\_label3 = np.append(mean\_set\_label3,3)  
sample\_mean\_set\_labelled = np.array( [mean\_set\_label1,mean\_set\_label2,mean\_set\_label3])  
estimate\_label\_set= nearest\_classifier(train\_data,sample\_mean\_set\_labelled,feature1,feature2)  
TrainDataErrorRate = computeErrorRate(estimate\_label\_set,train\_data\_labels)  
plotDecBoundaries(train\_data[:,[feature1,feature2]],train\_data\_labels,sample\_mean\_set\_unlabelled)  
minErrorRate = 1000000000000000000000  
minFeature1 = 0  
minFeature2 = 0  
ErrorRateSet = []  
minUnlabelledSampleMean = []  
for i in range(13):  
 for j in range(i+1,13):  
 ErrorRate,sample\_mean\_set\_unlabelled , sample\_mean\_set\_labelled = searchFeature(train\_data,test\_data, i , j)  
 ErrorRateSet.append(ErrorRate)  
 if minErrorRate > ErrorRate:  
 minErrorRate = ErrorRate  
 minFeature1 = i  
 minFeature2 = j  
 minUnlabelledSampleMean = sample\_mean\_set\_unlabelled  
  
errormean = 0  
sumerror = 0  
for error in ErrorRateSet:  
 sumerror += error  
variance = 0  
errormean = sumerror/len(ErrorRateSet)  
for error in ErrorRateSet:  
 variance += (error-errormean)\*(error-errormean)  
  
print("The most suitable feature")  
print(minFeature1, minFeature2)  
print(min(ErrorRateSet))  
print(errormean)  
print(variance)  
plotDecBoundaries(train\_data[:,[minFeature1,minFeature2]],train\_data\_labels,minUnlabelledSampleMean)  
'''   
for i in range(len(train\_data)-1):  
 for j in range(i+1,len(train\_data)-1):  
 mean\_set = []  
 train\_mean\_set\_nearest\_classifier(train\_data,mean\_set,i,j)  
 train\_error\_set\_for\_features.append(computeErrorRate())  
'''

wine2.py

import numpy as np  
from python3.plotDecBoundaries import plotDecBoundaries  
from tools.nearest\_centroid\_classifier import \*  
  
train\_data = np.genfromtxt('python3/wine\_train.csv', delimiter = ',')  
test\_data = np.genfromtxt('python3/wine\_test.csv' , delimiter = ',')  
  
feature1 = 0  
feature2 = 1  
train\_data\_labels= getLabels(train\_data)  
test\_data\_labels= getLabels(test\_data)  
  
  
TrainDataErrorRate,sample\_mean\_set\_unlabelled , sample\_mean\_set\_labelled = searchFeature(train\_data,test\_data, feature1 , feature2)  
  
estimatedTestDataLabel = nearest\_classifier(test\_data,sample\_mean\_set\_labelled,0,1)  
computeErrorRate(test\_data\_labels,estimatedTestDataLabel)

nearest\_centroid\_classifier.py

import numpy as np  
from math import \*  
  
  
def \_\_init\_\_(self):  
 print("This nearest\_centroid\_classifier")  
  
  
def computeVectorEuclideanDistance(v1 ,v2):  
 sum = 0  
 difference =np.array(v1) -np.array(v2)  
 for feature in difference:  
 sum += feature\*feature  
  
 return sqrt(sum)  
  
def nearest\_centroid\_classifier(test\_data, sample\_mean\_set,estimate\_label\_set):  
 for data in test\_data:  
 minClass = 0   
 minDistance = 99  
 for mean in sample\_mean\_set:  
 mean\_feature\_set = mean[:, [0,1]]  
 if computeVectorEuclideanDistance(mean\_feature\_set,data)<minDistance :  
 minDistance = computeVectorEuclideanDistance(mean\_feature\_set,data)  
 minClass = mean[2]  
 estimate\_label\_set.append(minClass)  
  
def nearest\_classifier(dataset , sample\_mean\_set, f1,f2):  
 estimated\_label\_set = []  
 for data in dataset :  
 specificData = [data[f1],data[f2]]  
 minDistance = 10000000000000000000000000000000000000000000  
 minClass = 0  
 for mean in sample\_mean\_set:  
 specificMean = [mean[0],mean[1]]  
 if minDistance>computeVectorEuclideanDistance(specificData,specificMean):  
 minClass = mean[2]  
 minDistance = computeVectorEuclideanDistance(specificMean,specificData)  
 estimated\_label\_set.append(minClass)  
 return estimated\_label\_set  
def computeErrorRateForDataSet(dataset,sample\_mean\_set, f1,f2):  
 estimated\_label\_set = nearest\_centroid\_classifier(dataset,sample\_mean\_set,f1,f2)  
  
  
def computeErrorRate(labelset1, labelset2):  
 LabelCount = 0   
 ErrorCount = 0  
 for i in range(len(labelset1)):  
 if(labelset1[i]!=labelset2[i]):  
 ErrorCount+=1  
 LabelCount+=1  
 result = ErrorCount/LabelCount  
 print("The error rate: ",result," The total test data: ",LabelCount)  
 return result  
  
  
def getLabels(dataset):  
 labelset = []  
 for data in dataset:  
 labelset=np.append(labelset,data[13])  
 return labelset  
# input data\_set you can select 2 features  
# return the mean vector  
def train\_classifier(data\_set,f1 ,f2):  
 sum = 0  
 count = 0  
 mean\_set = []  
 data\_set\_train = data\_set[:,[f1,f2]]  
 for data in data\_set\_train:  
 sum+= data  
 count+= 1  
 mean\_set= sum/count  
 return mean\_set  
# input original data set and label only a number  
  
def train\_classifer\_label(data\_set, label , f1 ,f2 ):  
 sum = 0  
 count = 0  
  
 for i in range(len(data\_set)):  
 data = data\_set[i]  
 #print(data[f1:f2+1])  
 if data[13] == label :  
 sum+= np.array([data[f1],data[f2]])  
 count += 1  
 mean\_set = sum/count  
 return mean\_set  
  
def get\_data\_set\_by\_label(dataset,label):  
 result = []  
 for data in dataset:  
 if data[13] == label:  
 dataset.append(data)  
 return result  
  
  
  
# input:  
def classify\_with\_two\_feature(dataset=[[0,0]]):  
 result = []  
 for data in dataset :  
 result.append(data)  
  
  
def model\_validation(test\_dataset, mean\_set,f1,f2,errorrate):  
 test\_dataset = []  
  
  
def searchFeature(train\_data\_set, test\_Data\_set, feature1, feature2):  
 mean\_set\_label1 = train\_classifer\_label(train\_data\_set, 1, feature1, feature2)  
 mean\_set\_label2 = train\_classifer\_label(train\_data\_set, 2, feature1, feature2)  
 mean\_set\_label3 = train\_classifer\_label(train\_data\_set, 3, feature1, feature2)  
 sample\_mean\_set\_unlabelled = np.array([mean\_set\_label1, mean\_set\_label2, mean\_set\_label3])  
  
 mean\_set\_label1 = np.append(mean\_set\_label1, 1)  
 mean\_set\_label2 = np.append(mean\_set\_label2, 2)  
 mean\_set\_label3 = np.append(mean\_set\_label3, 3)  
  
 sample\_mean\_set = [mean\_set\_label1, mean\_set\_label2, mean\_set\_label3]  
 estimate\_label\_set = nearest\_classifier(train\_data\_set, sample\_mean\_set, feature1, feature2)  
 train\_data\_labels = getLabels(train\_data\_set)  
 TrainDataErrorRate = computeErrorRate(estimate\_label\_set, train\_data\_labels)  
  
 return TrainDataErrorRate,sample\_mean\_set\_unlabelled ,sample\_mean\_set  
  
def findBestFeatureByTrainData(train\_data, test\_data):  
 ErrorRateSet = []  
 train\_data\_labels = getLabels(train\_data)  
 for i in range(13):  
 for j in range(i + 1, 13):  
 ErrorRate, sample\_mean\_set\_unlabelled,sample\_mean\_set\_labelled = searchFeature(train\_data, test\_data, i, j)  
 ErrorRateSet.append(ErrorRate)  
 if minErrorRate > ErrorRate:  
 minErrorRate = ErrorRate  
 minFeature1 = i  
 minFeature2 = j  
 minUnlabelledSampleMean = sample\_mean\_set\_unlabelled  
 print("The most suitable feature")  
 print(minFeature1, minFeature2)  
 plotDecBoundaries(train\_data[:,[minFeature1,minFeature2]],train\_data\_labels,minUnlabelledSampleMean)

synthetic.py

import numpy as np   
from python3.plotDecBoundaries import plotDecBoundaries  
from tools.nearest\_centroid\_classifier import \*  
  
  
  
train\_data = np.genfromtxt('python3/synthetic2\_train.csv',delimiter=',')  
test\_data = np.genfromtxt('python3/synthetic2\_test.csv',delimiter = ',')  
#train\_data\_2 = np.genfromtxt('python3/synthetic2\_train.csv', delimiter = ',')  
#print(train\_data\_1)  
  
  
  
sumClass1 = np.array([0,0])  
sumClass2 = np.array([0,0])  
countClass1 = 0   
countClass2 = 0  
train\_labels = []  
test\_labels = []   
for data in train\_data:  
 train\_labels = np.append(train\_labels,data[2])  
for data in test\_data:  
 test\_labels = np.append(test\_labels,data[2])  
  
for data in train\_data:  
 if data[2] == 1 :   
 sumClass1[0] += data[0]  
 sumClass1[1] += data[1]  
 countClass1+=1  
 else:   
 sumClass2[0] += data[0]  
 sumClass2[1] += data[1]  
 countClass2+=1  
estimate\_labels = []   
test\_data\_unlabelled = test\_data[:,[0,1]]  
mean\_class1 = sumClass1/countClass1   
mean\_class2 = sumClass2/countClass2  
mean\_class1\_label = np.append(mean\_class1,1)  
mean\_class2\_label = np.append(mean\_class2,2)  
mean\_sample = np.array([mean\_class1,mean\_class2])  
for data in test\_data\_unlabelled:  
 if computeVectorEuclideanDistance(data,mean\_class1)> computeVectorEuclideanDistance(data,mean\_class2):  
 estimate\_labels.append(2)  
 else:  
 estimate\_labels.append(1)  
  
errorrate = computeErrorRate(estimate\_labels,test\_labels)  
train\_data\_plot = test\_data[:,[0,1]]  
plotDecBoundaries(test\_data[:,[0,1]],test\_labels,mean\_sample)