# Density of data points

import warnings

warnings.filterwarnings("ignore")

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import sklearn.preprocessing

import random

import keras

from keras.models import Sequential

from keras.layers import Dense

POINT\_N = 1000

DIM\_N = 2

data0 = pd.read\_csv("./class\_splash\_3\_w100\_PCA.csv", sep=',')

print("Info of the data:", data0.info())



data0 = data0.to\_numpy(copy=True)

scaler = sklearn.preprocessing.MinMaxScaler((-1.0, 1.0))

data0 = scaler.fit\_transform(data0)

data = np.empty([POINT\_N, DIM\_N])

for i in range(POINT\_N):

 data[i] = data0[random.randrange(0, data0.shape[0])]

plt.figure(figsize=(8, 8))

for n in range(len(data)):

 plt.scatter(data[n][0], data[n][1], c='blue', marker='o')

plt.show()



# ---------------------------------------------------------------------------------------------

model = Sequential()

model.add(Dense(30, input\_dim=DIM\_N, activation='relu'))

model.add(Dense(2, activation='softmax'))

opt = keras.optimizers.RMSprop(learning\_rate=.001, momentum=0.9)

model.compile(loss='mse', optimizer=opt, metrics=['accuracy'])

eps = 0.0001

err = 1.0e+99

data1 = np.empty([POINT\_N, DIM\_N])

DD = int(2.5 \* data1.shape[0])

iter = 0

while iter < 1000:

 iter += 1

 for i in range(POINT\_N):

 data1[i] = data0[random.randrange(0, data0.shape[0])]

 data = data1[:, 0:DIM\_N]

 data = np.concatenate((data, np.zeros((data.shape[0], 1))), axis=1)

 data\_noise = np.random.rand(DD, DIM\_N) \* 2.0 - 1.0

 data\_noise = np.concatenate((data\_noise, np.ones((DD, 1))), axis=1)

 data = np.concatenate([data, data\_noise], axis=0)

 np.random.shuffle(data)

 x = data[:, 0:DIM\_N]

 y = data[:, DIM\_N].astype(int)

 y = keras.utils.to\_categorical(y, num\_classes=2)

 if iter == 1:

 plt.figure(figsize=(8, 8))

 for n in range(len(x)):

 plt.scatter(x[n][0], x[n][1], c=('blue' if y[n][0] == 0 else 'red'), marker='o')

 plt.show()



 hist = model.fit(x, y, batch\_size=100, epochs=1, verbose=0)

 if abs(hist.history['loss'][0] - err) < eps and iter > 100:

 break

 err = hist.history['loss'][0]

 print('\*\*\*\*\* ', iter, ' - ', err)

# -----------------------------------------------------------------------------------------------

xx = np.empty([3\*POINT\_N, DIM\_N])

for i in range(3\*POINT\_N):

 xx[i] = data0[random.randrange(0, data0.shape[0])]

pred = model.predict(xx)

res = np.argmax(pred, axis=1)

plt.figure(figsize=(8, 8))

for n in range(len(xx)):

 plt.scatter(xx[n][0], xx[n][1], c=('blue' if res[n] == 0 else 'red'), marker='o')

plt.show()

a = 1.0 - np.sum(res)/len(xx)

print("\n", "Accuracy of the dataset", a)



