Київський національний університет імені Тараса Шевченка
Факультет радіофізики, електроніки та комп’ютерних систем
Кафедра нанофізики та наноелектроніки

Звіт з лабораторної роботи 3
З курсу «комп’ютерна фізика»

Студента 2 курсу магістратури
кафедра НФНЕ
Кухельного Кирила

київ - 2017

Вступ

Машинне навчання – це спрощена версія процесу навчання, яке відбувається з людиною. Як правило, в машинному навчанні наявний певний набір прикладів, спостережень, реакцій до цих спостережень. Задача полягає у тому, щоб сконструювати такі моделі, які будуть максимально ефективно описувати наявні дані і робити достовірні прогнози. Такими прогнозами можуть бути відповіді на питання: чи є котик на зображенні? Чи є сенс зараз купляти акції певної компанії? Це позитивний чи негативний відгук на товар в Інтернеті? Існує безліч сфер застосування машинного навчання. Така розповсюдженість стала наслідком розвитку Інтернету і накопичення величезної кількості даних, а також тим, що широкому загалу стали доступні великі обчислювальні потужності.

Мета :

практично ознайомитися з модельними штучними нейронними мережами, оцінити характеристики ефективності мереж та отримати навики вибору структури мереж для різних типів задач розпізнавання та аналізу даних.

Експериментальна частина

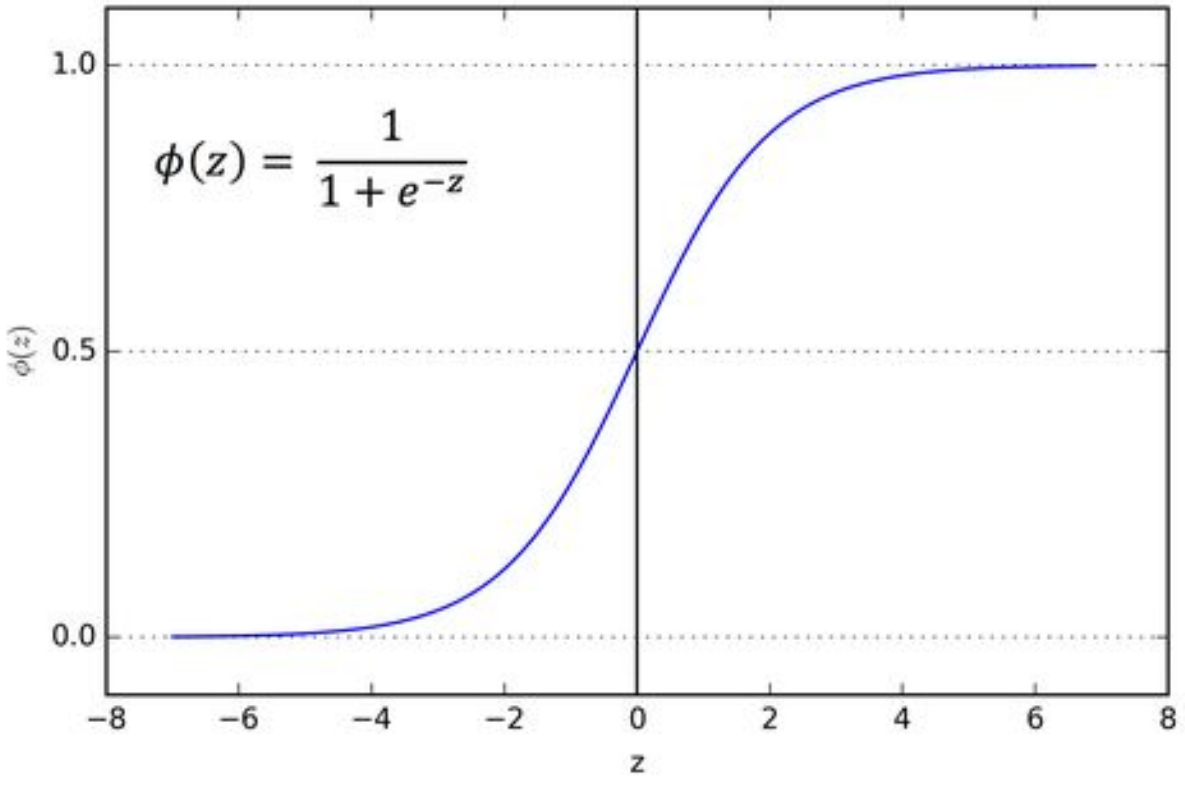
Було отримано два датасети:

1. 3ABD_03_HFHE_17_Training_set.txt
2. 3ABD_03_HFHE_17_Testing_set.txt

В датасеті 3ABD_03_HFHE_17_Testing_set.txt вхідні данні та результати, тобто
“y x1 x2 x3 x4 x5 … x63 x64”, а в тестовому набір данних які треба обробити вже навченою нейромережею.

Побудуємо модель на 64 входи, 64 нейрони в першому шарі 64 в другому шарі 1 нейрон в третьому. Для навчання виберемо 3000 епох. Розмір батча 64 елементи.Функцію активації вибираємо sigmoid. так як вона краще всього зможе опрацювати данні.

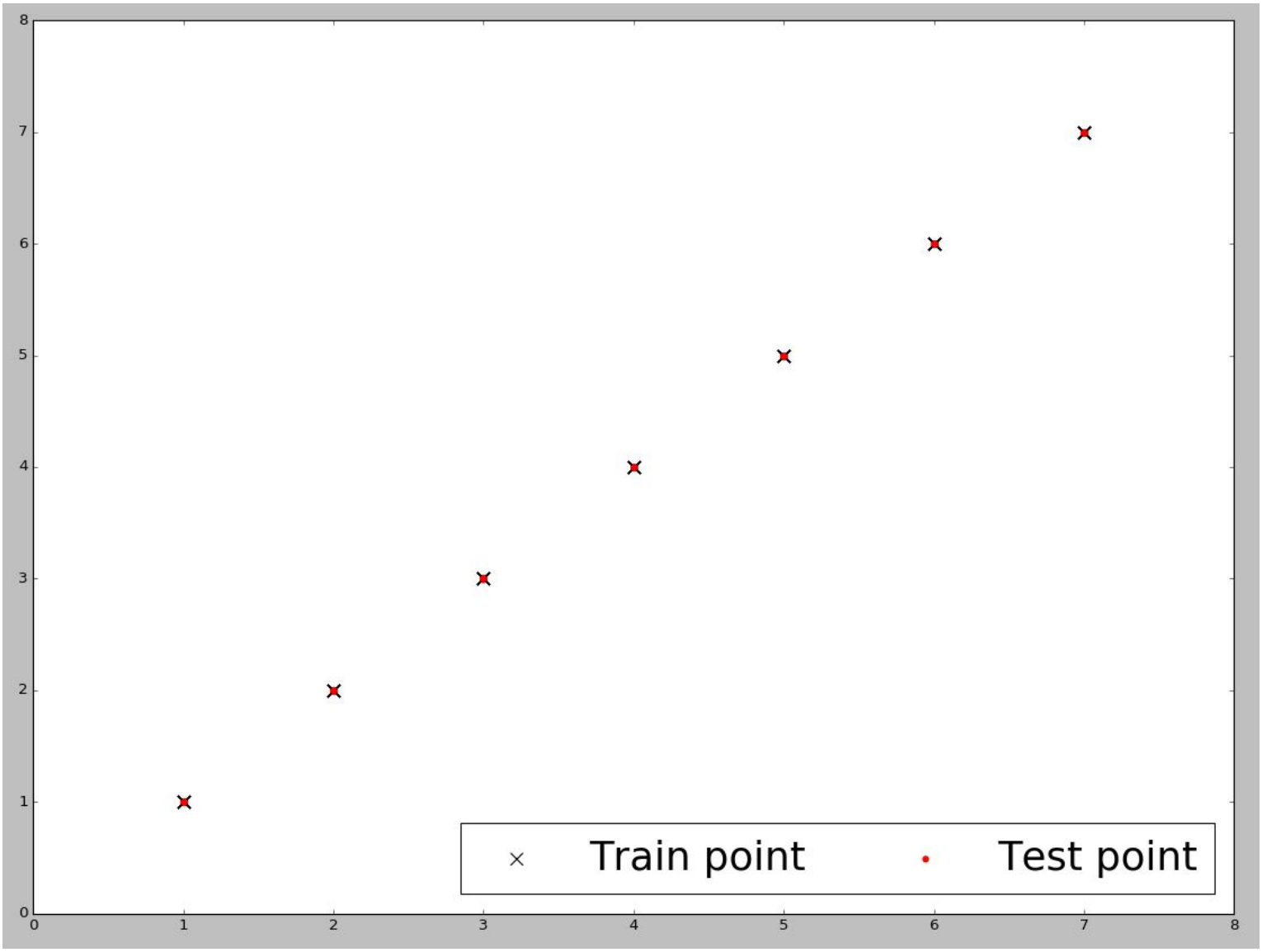
$$f(x) = sigmoid(x) = \frac{1}{1+e^{-x}}$$



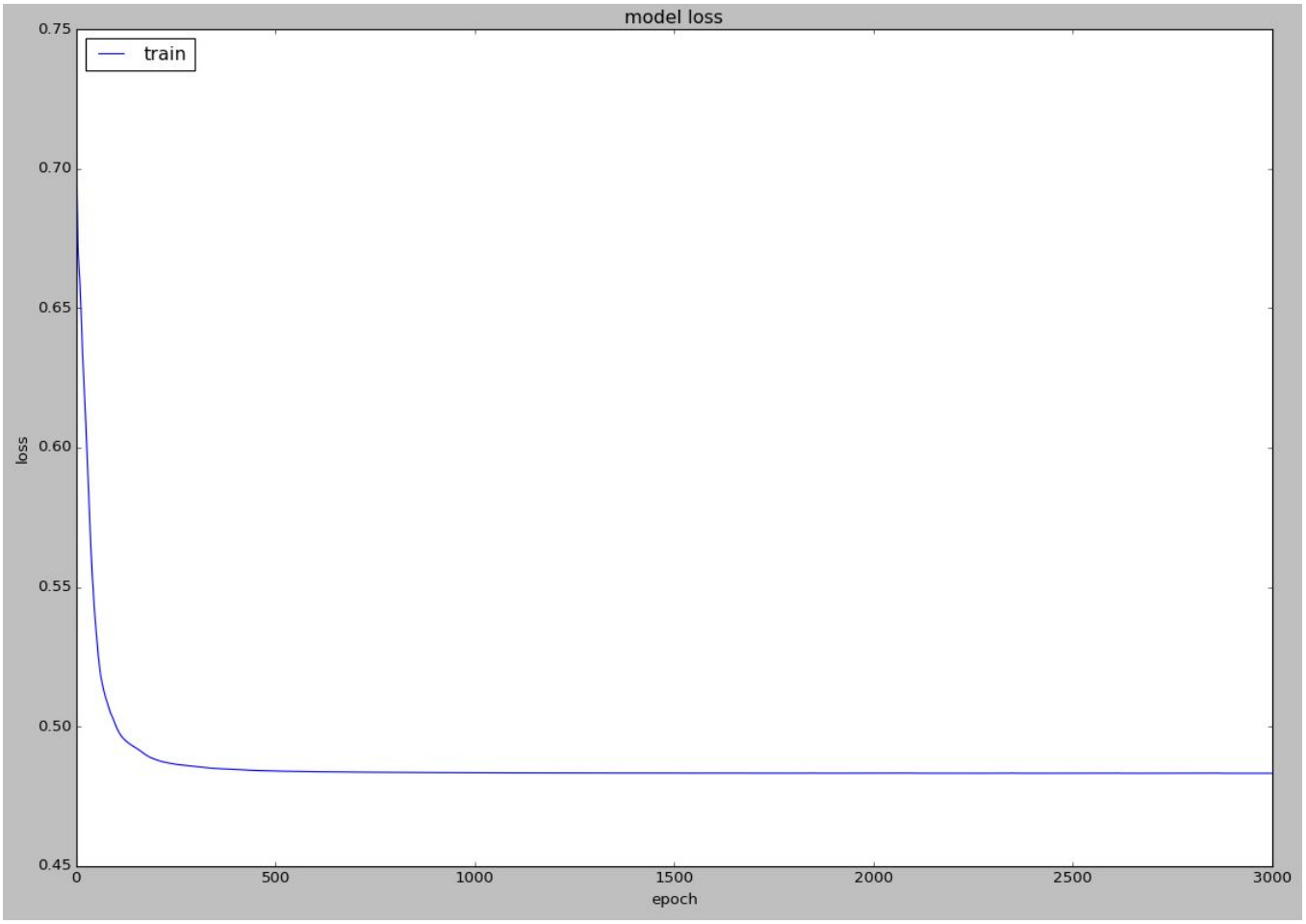
Dataset, 64 входи, 2 шари, (64-64-1), 3000 епох				
Функція активації	Помилка	Точність	Час, сек	Данні
sigmoid	loss: 0.4832	acc: 0.1429	4.9	1.0 >> 1.0 1.0 >> 0.982 1.0 >> 0.988 1.0 >> 1.0 1.0 >> 1.007 1.0 >> 1.042 1.0 >> 1.0 1.0 >> 1.016 2.0 >> 2.0 2.0 >> 1.974 2.0 >> 2.0 2.0 >> 1.986 2.0 >> 2.024 2.0 >> 1.971 2.0 >> 2.0 2.0 >> 2.013 3.0 >> 3.0 3.0 >> 2.994 3.0 >> 3.0 3.0 >> 3.0 3.0 >> 3.01 3.0 >> 3.0 3.0 >> 3.002 3.0 >> 3.0 4.0 >> 4.0 4.0 >> 3.974 4.0 >> 3.962 4.0 >> 4.029 4.0 >> 4.0 4.0 >> 4.0 4.0 >> 4.019 4.0 >> 4.0 5.0 >> 5.041 5.0 >> 5.0 5.0 >> 5.0 5.0 >> 4.965 5.0 >> 5.0 5.0 >> 5.0 5.0 >> 4.981 5.0 >> 4.981 6.0 >> 6.009 6.0 >> 6.0 6.0 >> 6.0 6.0 >> 6.0 6.0 >> 6.0 6.0 >> 6.0 6.0 >> 6.0 6.0 >> 6.0 7.0 >> 7.0 7.0 >> 7.0 7.0 >> 7.0 7.0 >> 7.0 7.0 >> 7.0 7.0 >> 7.0 7.0 >> 7.0 7.0 >> 7.0

Враховуючи невелику кількість данних, точність моделі доволі гарна. Тож маючи натреновану модель, використаємо її для тестової вибірки данних.

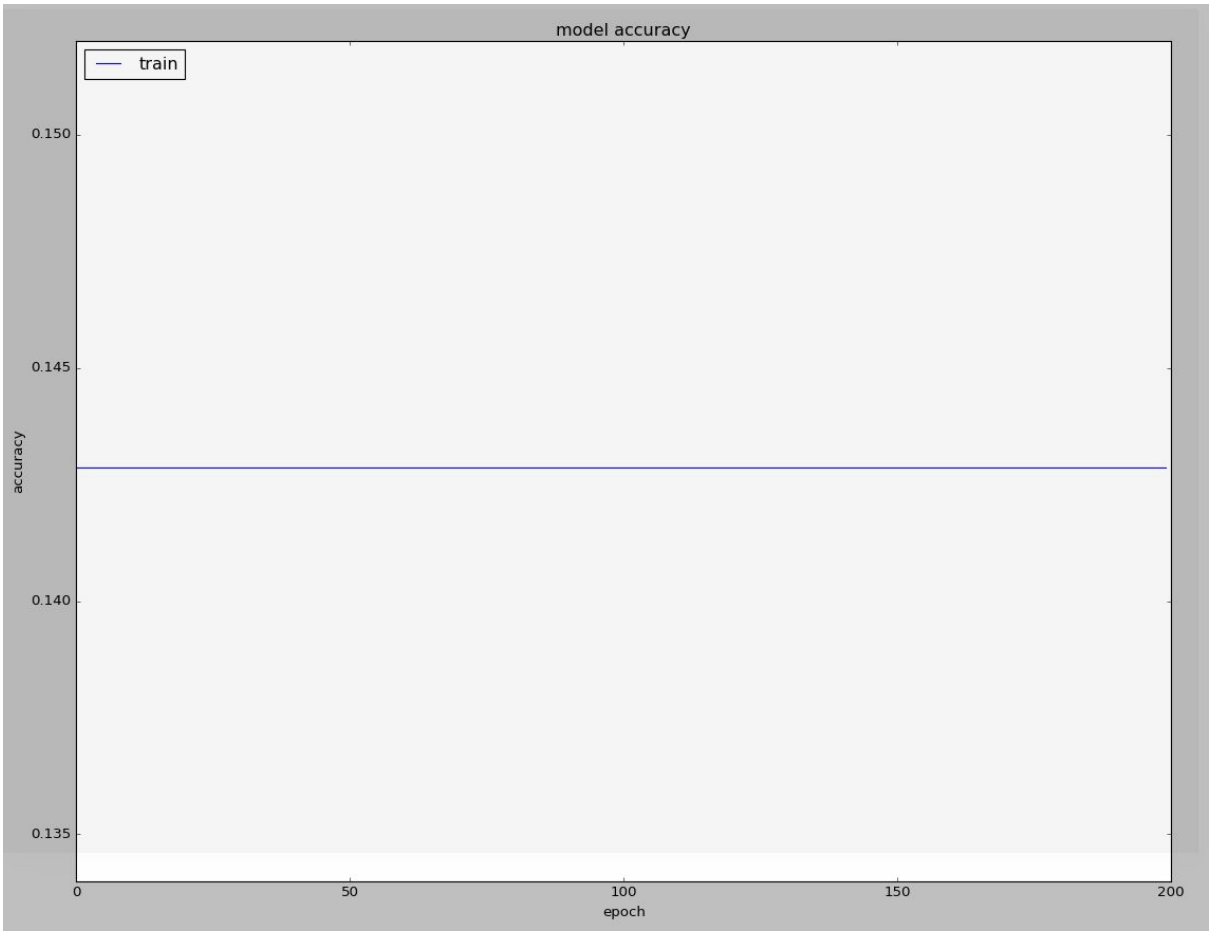
>> 1.001 >> 1.0 >> 1.0 >> 1.0 >> 1.0 >> 1.0 >> 1.0 >> 0.999 >> 2.0 >> 1.999 >> 2.001 >> 2.0 >> 2.0 >> 2.0 >> 2.0 >> 2.0 >> 2.998>> 3.0 >> 3.0 >> 3.0 >> 3.0 >> 3.0 >> 3.0 >> 3.0 >> 4.0 >> 4.003 >> 4.001 >> 3.999 >> 3.998 >> 4.0 >> 4.0 >> 4.0 >> 5.0 >> 5.0 >> 5.0 >> 5.0 >> 5.0 >> 5.0 >> 5.0 >> 5.0 >> 6.0 >> 6.0 >> 6.0>> 6.0 >> 5.999 >> 6.001 >> 6.0 >> 6.0 >> 7.0 >> 7.0 >> 7.0 >> 7.0 >> 7.0 >> 7.0 >> 7.0



Таким чином, після візуалізації данних можна бачити що хрестиками позначені точки з тренувальної вибірки, а кружечками з тестової. Як бачимо розкид мінімальний. Точність моделі гарна. Був доданий ще один шар з 64 нейронів, так як двошарова модель давала значний розкид. що було неприйнятним.



Графік залежності втрат під час навчання нейромережі. Можна бачити що втрати нормалізуються в районі 1000 епохи. Що каже про надмірність кількості епох. Можна було взяти менше. На наступному графіку залежність точності моделівід кількості епох. Лишається стабільною протягом всього навчання. Пояснюється тим що розмір батча дорівнював всьому стеку данних, для прискорення навчання. Через це не відбувалось перенавчання мережі.



Висновок:

Вибрна модель гарно описує датасет. Точність моделі видно на графіку. Для конструювання нейромережі було вибрано sigmoid функцію. Час навчання для данної архітектури 4.9 сек.
loss: 0.4833 - acc: 0.1429.

Код програми наведено нижче.

```
import numpy as np
import os
os.environ['TF_CPP_MIN_LOG_LEVEL']='2'
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.layers import Embedding
from keras.layers import LSTM
from keras.layers import Convolution1D, MaxPooling1D
from keras.datasets import imdb
from keras.callbacks import ModelCheckpoint
from keras.utils import np_utils
from keras.utils import plot_model
```

```
import time
import matplotlib as mpl
import matplotlib.pyplot as plt

batch_size = 64
NEURONS = 64
shape_x = 64
epochs = 3000
norma = 7

def training_set():
    with open("3ABD_03_HFHE_17_Training_set.txt") as fp:
        data = fp.read()
    data = data.split("\n")
    X_train = []
    y_train = []
    for d in data[1:]:
        row = d.split("\t")
        y_train.append([float(el) for el in row[0]])
        X_train.append([float(el) for el in row[1:]])
    return (np.array(X_train), np.array(y_train)/norma )

def test_set():
    with open("3ABD_03_HFHE_17_Testing_set.txt") as fp:
        data = fp.read()
    data = data.split("\n")
    X_test = []
    for d in data[1:]:
        row = d.split("\t")
        X_test.append([float(el) for el in row])
    return (np.array(X_train) )

X_train, y_train = training_set()
X_test = test_set()

"""
Activation:
  - relu
  - elu
  - selu
  - softmax
  + sigmoid
  + hard_sigmoid
  - linear
  - tanh
"""

act = "sigmoid"
TIMES = time.time()
model = Sequential() #модель така що кожен шар має лише 1 наступний шар і навпаки
model.add(Dense(output_dim=NEURONS, input_dim=shape_x, activation=act))
#model.add(Dense(output_dim=2048, activation=act))
#model.add(Dense(output_dim=512, activation=act))
model.add(Dense(output_dim=64, activation=act))
model.add(Dense(output_dim=1, activation=act))
#model.add(Dropout(4) )
model.summary() #Print model info
model.compile(loss="binary_crossentropy",
              optimizer="adam",
              metrics=["accuracy"],
              )
```

```
print("Train ...")

history = model.fit(X_train, y_train, batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    #validation_data=(X_test, y_test)
                    )

print(history.history.keys())
# summarize history for accuracy
plt.plot(history.history['acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train'], loc='upper left')
plt.show()

def predict(x_tr, ind):
    prediction = model.predict(np.array(x_tr[ind]).reshape(1, shape_x))
    return round(prediction[0][0]*norma, 3)

print("\n\n")
print("NEURONS >> ", NEURONS, ",Activation >> ", act, ",epochs >> ", epochs)
print("Learning time >> ", round(time.time() - TIMES, 3), "сек")
print("\n\ny_train ")
for i in range(len(X_train)):
    prediction = model.predict(np.array(X_train[i]).reshape(1, shape_x))
    print(y_train[i][0]*norma, " >> ", round(prediction[0][0]*norma, 3) )

print("\n\ny_test ")
y_test = []
for i in range(len(X_test)):
    prediction = model.predict(np.array(X_test[i]).reshape(1, shape_x))
    y_test.append(prediction)
    print(" >> ", round(prediction[0][0]*norma, 3) )

def plot_1D(X, Y1, Y2):
    fig = plt.figure()
    ax = fig.add_subplot(111)
    for i in range(len(X)):
        lo = ax.scatter(X[i], Y1[i], marker='x', s=100.0, color="k")
        li = ax.scatter(X[i], Y2[i], marker='o', color="red")

    plt.legend((lo, li),
           ('Train point', 'Test point'),
           scatterpoints=1,
           loc='lower right',
           ncol=30,
           fontsize=32)
    plt.show()
y_tr = [el[0]*norma for el in y_train]
y_ts = [el[0]*norma for el in y_test]
plot_1D(y_tr, y_tr, y_ts)
```