МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по практической работе № 5

по дисциплине «Искусственные нейронные сети»

Студент гр. 8383 _____ Костарев К.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Постановка задачи.

Необходимо в зависимости от варианта сгенерировать датасет и сохранить его в формате csv.

Построить модель, которая будет содержать в себе автокодировщик и регрессионную модель.

В качестве результата представить исходный код, сгенерированные данные в формате csv, кодированные и декодированные данные в формате csv, результат регрессии в формате csv (что должно быть и что выдает модель), и сами 3 модели в формате h5.

Вариант 3, целевая функция 1.

X ∈ N(0,10)
e ∈ N(0,0.3)

Признак	1	2	3	4	5	6	7
Формула	X^2+X+e	|X|+e	sin(X-PI\4)+e	lg(|X|)+e	-X^3+e	-X/4+e	-X+e

Выполнение работы.

Была написана программа gen_data.py для генерации данных:

```
import numpy as np


def f1(x, e):
    return x**2 + x + e


def f2(x, e):
    return np.abs(x) + e


def f3(x, e):
    return np.sin(x - np.pi / 4) + e


def f4(x, e):
    return np.log(abs(x)) + e


def f5(x, e):
    return -np.power(x, 3) + e
```

```
def f6(x, e):
    return -x/4 + e


def f7(x, e):
    return -x + e


def gen_data(num_data=1000):
    X = np.random.normal(loc=0, scale=10, size=num_data)
    E = np.random.normal(loc=0, scale=0.3, size=num_data)
    data_train = np.array([[f2(X[i], E[i]), f3(X[i], E[i]), f4(X[i], E[i]),
    f5(X[i], E[i]), f6(X[i], E[i]), f7(X[i], E[i])] for i in range(num_data)])
    data_labels = np.reshape(np.array([f1(X[i], E[i]) for i in range(num_data)]),
(num_data, 1))
    data = np.hstack((data_train, data_labels))
    return data


train = gen_data(1000)
validation = gen_data(200)

np.savetxt("train.csv", train, delimiter=";")
np.savetxt("validation.csv", validation, delimiter=";")
```

Далее в файле main.py сгенерированные данные загружаются:

```
train = np.genfromtxt("train.csv", delimiter=";")
validation = np.genfromtxt("validation.csv", delimiter=";")

train_data = np.reshape(train[:,:6], (len(train), 6))
train_labels = np.reshape(train[:,6], (len(train), 1))

validation_data = np.reshape(validation[:,:6], (len(validation), 6))
validation_labels = np.reshape(validation[:,6], (len(validation), 1))
```

Далее входной вектор 6-ой размерности преобразуется в размер из 4.

```
encoding_dim = 4
layer_input_main = Input(shape=(6,), name="input_main")
```

Далее были настроены слои для кодировки:

```
layer_encoded = Dense(encoding_dim * 6, activation="relu")(layer_input_main)
layer_encoded = Dense(encoding_dim * 3, activation="relu")(layer_encoded)
layer_encoded_output = Dense(encoding_dim, name="output_encoded")(layer_encoded)
```

Также были настроены слои для декодировки:

```
layer_input_decoded = Input(shape=(encoding_dim,), name="input_decoding")
layer_decoded = Dense(encoding_dim * 3, activation="relu")(layer_input_decoded)
layer_decoded = Dense(encoding_dim * 6, activation="relu")(layer_decoded)
```

```
layer_decoded_output = Dense(6, name="output_decoded")(layer_decoded)
```

И регрессии:

```
layer_input_regression = Input(shape=(encoding_dim,), name="input_regression")
layer_regression = Dense(encoding_dim * 6, activation="relu")(layer_input_regression)
layer_regression = Dense(encoding_dim * 4, activation="relu")(layer_regression)
layer_regression_output = Dense(1, name="output_regression")(layer_regression)
```

Затем были созданы три модели соответственно для каждой операции. Также была создана модель model_main, объединяющая все остальные, для которой была проведена тренировка. После тренировки был произведен быстрый тест, на одном случайном тестовом наборе данных. В конце концов через каждую из 3 моделей были пропущены тестовые данные и все результаты и сами модели были сохранены в файлы.

```
test_index = np.random.randint(0, len(validation_data) - 1)
test_data = np.reshape(validation_data[test_index,:], (1, 6))
test_label = validation_labels[test_index,:]

print("Test data:", test_data)
print("Test label:", test_label)

encoded_data = encoder.predict(test_data)
print("Encoded data:", encoded_data)

regression_data = regression.predict(encoded_data)
print("Regression prediction:", regression_data)

decoded_data = decoder.predict(encoded_data)
print("Decoder prediction:", decoded_data)

encoded_data = encoder.predict(validation_data)
regression_data = regression.predict(encoded_data)
decoded_data = decoder.predict(encoded_data)

np.savetxt("encoded_data.csv", encoded_data, delimiter=";")
np.savetxt("decoded_data.csv", decoded_data, delimiter=";")
np.savetxt("regression_data.csv", np.hstack((regression_data, validation_labels)), delimiter=";")

encoder.save("model_encoder.h5")
decoder.save("model_decoder.h5")
regression.save("model_regression.h5")
```