Практическое задание №5

Варинат №2 Цель регресси 5

Условие:

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

Построить модель, которая будет содержать в себе автокодировщик и регрессионную модель. Схематично это должно выглядеть следующим образом:

Обучить модель и разбить обученную модель на 3: Модель кодирования данных (Входные данные -> Закодированные данные), модель декодирования данных (Закодированные данные -> Декодированные данные), и регрессионную модель (Входные данные -> Результат регрессии).

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 2

X ∈ N(-5,10)
e ∈ N(0,0.3)

Признак	1	2	3	4	5	6	7
Формула	-X^3+e	ln(|X|)+e	sin(3X)+e	exp(X)+e	X+4+e	-X+sqrt(|X|)+e	X+e

Выполнение:

Была написана генерация тренировочных и тестовых данных, а также реализована их запись в файл:

```
import numpy as np
#v2_5

def f1(x, e):
    return pow(-x, 3)+e


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


def f3(x, e):
    return np.sin(3*x)+e


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


def f5(x, e):
    return x+4+e


def f6(x, e):
    return -x+np.sqrt(abs(x))+e


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

m_x = 0
x_e = 0
s_x = 10
s_e = 0.3
train_size = 1000
test_size = 300
X = np.random.normal(m_x, s_x, train_size)
```

```
E = np.random.normal(m_x, s_x, train_size)
train_data = np.array([[f1(X[i], E[i]), f2(X[i], E[i]), f3(X[i], E[i]),
f4(X[i], E[i]), f6(X[i], E[i]), f7(X[i], E[i])] for i in range(train_size)])
train_labels = np.array([f5(X[i], E[i]) for i in range(train_size)])
train_labels = np.reshape(train_labels, (train_size, 1))
train_data = np.hstack((train_data, train_labels))

X = np.random.normal(m_x, s_x, test_size)
E = np.random.normal(m_x, s_x, test_size)
test_data = np.array([[f1(X[i], E[i]), f2(X[i], E[i]), f3(X[i], E[i]),
f4(X[i], E[i]), f6(X[i], E[i]), f7(X[i], E[i])] for i in range(test_size)])
test_labels = np.array([f5(X[i], E[i]) for i in range(test_size)])
test_labels = np.reshape(test_labels, (test_size, 1))
test_data = np.hstack((test_data, test_labels))

np.savetxt("train_data.csv", train_data, delimiter=",")
np.savetxt("test_data.csv", test_data, delimiter=",")
```

Считывание из файла производится следующим образом:

```
train_dataset = pandas.read_csv("train_data.csv", header=None)
train_dataset = train_dataset.values
train_data = train_dataset[:,0:6].astype(float)
train_labels = train_dataset[:,6].astype(float)

test_dataset = pandas.read_csv("test_data.csv", header=None)
test_dataset = test_dataset.values
test_data = test_dataset[:,0:6].astype(float)
test_labels = test_dataset[:,6].astype(float)
```

Также производится нормировка данных, это позволяет избежать появления nan в работе модели и делает ее поведение корректным.

```
mean = train_data.mean(axis=0)
train_data -= mean
std = train_data.std(axis=0)
train_data /= std
```

Была написана архитектура модели:

```
#encoder
output_e1 = Dense(64, activation='relu')(inputs)
output_e2 = Dense(32, activation='relu')(output_e1)
encoder = Dense(4, name="encoder")(output_e2)

#decoder
output_d1 = Dense(32, activation="relu")(encoder)
output_d2 = Dense(64, activation="relu")(output_d1)
decoder = Dense(6, name="decoder")(output_d2)

#regression
output_r1 = Dense(8, activation="relu")(encoder)
output_r2 = Dense(64, activation="relu")(output_r1)
regression = Dense(1, name="regression")(output_r2)
```

```
model = Model(inputs=inputs, outputs=[decoder, regression])
model.compile(optimizer="adam", loss='mse', metrics=['mae'])
```

Модель была обучена, и разбита на три отдельные модели: model_encoder, model_decoder, model_regression

```
model.fit(train_data, [train_data, train_labels], epochs=200, batch_size=10)


model_encoder = Model(inputs=inputs, outputs=encoder)
encoder_predict = model_encoder.predict(test_data)
np.savetxt("encoder_test.csv", encoder_predict, delimiter=",")
model.save("model_encoder.h5")


model_decoder = Model(inputs=inputs, outputs=decoder)
decoder_predict = model_decoder.predict(test_data)
np.savetxt("decoder_test.csv", decoder_predict, delimiter=",")
model.save("model_decoder.h5")

model_regression = Model(inputs=inputs, outputs=regression)
regression_predict = model_regression.predict(test_data)
np.savetxt("regression_test.csv", regression_predict, delimiter=",")
model.save("model_regression.h5")
```

Все три модели были сохранены. Также через модели пропущены тестовые данные, результаты работы всех трех моделей были сохранены в csv файлы.

Сравнение части полученных данных приведено в таблицах. (на входе и выходе декодера нормированные значения):

Test_data	[-0.05742717 -1.62749495 -1.71682268 -0.03597105 -1.57508863 -0.77543579]
decoder	[-0.06925493 -1.6565641 -1.6023362 -0.03759114 -1.6043696 -0.7939152]

Test_lables	Regression
-6.36964169	-6.2076917
12.59654002	12.560927
11.57157251	11.543971
10.40990624	10.419944

Полученные данные разняться не сильно, что говорит о корректной работе декодера, энкодера и регрессии.