Практическое задание 5

Вариант 1, цель регрессии 2

Аверина Ольга, 8383

Задание:

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 1

$X \in N(3,10)$
$e \in N(0,0.3)$

Признак	1	2	3	4	5	6	7
Формула	X^2+e	sin(X/2)+e	cos(2x)+e	X-3+e	-X+e	|X|+e	(X^3)/4+e

Выполнение:

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

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


def f2(x, e):
    return np.sin(x/2) + e


def f3(x, e):
    return np.cos(2*x) + e


def f4(x, e):
    return x - 3 + e


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


def f6(x, e):
    return abs(x) + e


def f7(x, e):
    return (x**3)/4 + e


funcs = [f1, f3, f4, f5, f6, f7, f2]

train_size = 2500
test_size = 500

X_from = 3
X_to = 10
X_data_train = np.random.normal(X_from, X_to, train_size)
```

```
X_data_test = np.random.normal(X_from, X_to, test_size)

e_from = 0
e_to = 0.3
e_data_train = np.random.normal(e_from, e_to, train_size)
e_data_test = np.random.normal(e_from, e_to, test_size)

train_data = np.array([[funcs[j](X_data_train[i], e_data_train[i]) for j in
range(7)] for i in range(train_size)])
test_data = np.array([[funcs[j](X_data_test[i], e_data_test[i]) for j in
range(7)] for i in range(test_size)])

np.savetxt('train_data.csv', train_data, delimiter=' ', fmt='%1.5f')
np.savetxt('test_data.csv', test_data, delimiter=' ', fmt='%1.5f')
```

Считывание из файла производится следующим образом:

```
train_file = np.genfromtxt('train_data.csv', delimiter=' ')
test_file = np.genfromtxt('test_data.csv', delimiter=' ')
```

Также была произведена нормировка данных, чтобы избежать появления NaN в работе модели.

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

train_data = (train_data - mean) / std
test_data = (test_data - mean) / std
```

Была написана архитектура модели:

```
def func_encoder(input):
    e = Dense(64, activation='relu')(input)
    e = Dense(32, activation='relu')(e)
    e = Dense(4, activation='relu', name='encoder')(e)
    return e


def func_decoder(input):
    d = Dense(32, activation='relu', name='dc1')(input)
    d = Dense(64, activation='relu', name='dc2')(d)
    d = Dense(6, activation='relu', name='decoder')(d)
    return d


def func_regr(input):
    r = Dense(64, activation='relu')(input)
    r = Dense(128, activation='relu')(r)
    r = Dense(32, activation='relu')(r)
    r = Dense(16, activation='relu')(r)
    r = Dense(1, name='regression')(r)
    return r


input = Input(shape=(6,))
encoder = func_encoder(input)
decoder = func_decoder(encoder)
regression = func_regr(encoder)
```

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

```
model = Model(input, outputs=[regression, decoder])

model.compile(optimizer='adam', loss='mse', metrics=['mae'])
model.fit(train_data, [train_labels, train_data], epochs=100,
batch_size=25, validation_split=0.3)
model.evaluate(test_data, [test_labels, test_data])

encoder_model = Model(input, encoder)
decoder_model = Model(input, decoder)
regression_model = Model(input, regression)
```

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 таблице:

Test_lables	Regression
0.9262	0.92960256
1.94334	1.8794998
2.1394	2.10724
2.71059	2.4927306

Полученные данные различаются незначительно, что говорит о корректной работе декодера, энкодера и регрессии.