Практическое задание №5

Вариант №2

Цель регрессии: 2

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.

$X \in N(-5,10)$
$e \in N(0,0.3)$

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 функция `gen_data(size)`, генерирующая необходимый датасет.

```
def gen_data(size):
    data = np.empty([size, 6])
    labels = np.empty(size)
    for i in range(size):
        x = np.random.normal(-5, 10)
        e = np.random.normal(0, 0.3)
        data[i] = np.array([-1 * x ** 3, np.sin(3*x),
np.exp(x),x+4, -1 * x + np.sqrt(np.fabs(x)), x])
        data[i] += e
        labels[i] = np.log(np.fabs(x)) + e
    return data, labels
```

В соответствии со схемой на рис. 1 была создана модель ИНС в функциональном виде.

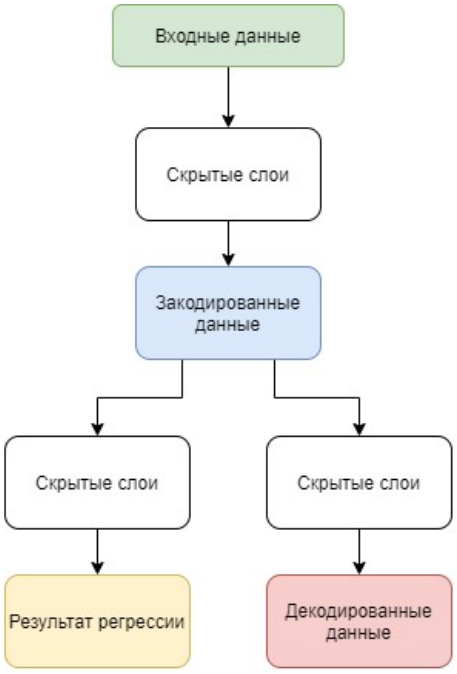

Рисунок 1 – Схема ИНС.

```
inputs = Input(shape=(6,))
encoder_1 = Dense(64, activation='relu')(inputs)
encoder_2 = Dense(32, activation='relu')(encoder_1)
encoder_3 = Dense(4, activation='relu',
name='encoder_output')(encoder_2)
decoder_1=Dense(32,activation='relu',name='decoder_1')(encod
er_3)
decoder_2=Dense(64,activation='relu',name='decoder_2')(decod
er_1)
decoder_3 = Dense(6, name='decoder_3')(decoder_2)
regr_1 = Dense(32, activation='relu')(encoder_3)
regr_2 = Dense(64, activation='relu')(regr_1)
regr_3 = Dense(45, activation='relu')(regr_2)
regression = Dense(1, name='predicted_output')(regr_3)
model = Model(inputs=inputs, outputs=[decoder_3, regression])
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

Далее общая модель была обучена, сохранена и разбита на 3 отдельные модели.

Регрессионная модель

```
regr_model = Model(inputs=inputs, outputs=regression)
```

Энкодер

```
encoder_model = Model(inputs=inputs, outputs=encoder_3)
```

Декодер

```
decoder_inputs = Input(shape=(4,))
dec_1 = model.get_layer('decoder_1')(decoder_inputs)
dec_2 = model.get_layer('decoder_2')(dec_1)
dec_3 = model.get_layer('decoder_3')(dec_2)
decoder_model = Model(decoder_inputs, dec_3)
```

Через эти отдельные модели были прогнаны тестовые данные. Модели были сохранены в файлах encoder_model.h5, regression_model.h5 и decoder_model.h5. Результат работы регрессионной модели был сохранен в файле regression.csv, результат декодировщика в файле decoded_test_n_data.csv, результат энкодера в файле encoded_test_n_data.csv.

Для корректной работы нейросети, данные, которые генерировались в начале, были подвергнуты нормировке. При отсутствии нормировки регрессионная модель выдает nan, этого можно избежать, используя в качестве

функции активации, например, sigmoid, однако была выбрана именно нормировка.

Сгенерированные данные хранятся в файлах train_data.csv, train_labels.csv, test_data.csv, test_labels.csv. Данные после нормировки хранятся в файлах train_data_n.csv, test_data_n.csv.

Сравним предсказании регрессии со значениями test_labels.

test_labels	regression
1.738944247397778975e+00	1.758089542388916016e+00
2.470397756632273012e+00	2.472320318222045898e+00
1.305220587032462021e+00	1.294270277023315430e+00
-4.658868655064308251e-01	-4.534713029861450195e-01

Полученные значение почти совпадают, что говорит о корректной работе регресионной модели.

Сравним декодированные тестовые нормированные данные и исходные тестовые нормированные данные.

test_data	decoded
-2.491738358213372861e-01	-2.402397245168685913e-01
5.263699290777846818e-01	5.040099024772644043e-01
-5.144466114335309592e-02	-6.796674430370330811e-02
-2.796615226471633742e-01	-2.983410060405731201e-01

Эти данные тоже во многом совпадают, что говорит о корректной работе энкодера и декодера.