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

ОТЧЕТ

по лабораторной работе 3

по дисциплине «Искусственные нейронные сети»

Тема: «Регрессионная модель изменения цен на дома в Бостоне»

Студентка гр. 7383 _____ Чемова К.А.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

Цель.

Реализовать предсказание медианной цены на дома в пригороде Бостона в середине 1970-х по таким данным, как уровень преступности, ставка местного имущественного налога и т.д.

Задачи.

- Ознакомиться с задачей регрессии
- Изучить отличие задачи регрессии от задачи классификации
- Создать модель
- Настроить параметры обучения
- Обучить и оценить модели
- Ознакомиться с перекрестной проверкой

Требования.

1. Объяснить различия задач классификации и регрессии;
2. Изучить влияние кол-ва нейронов на слое на результат обучения модели;
3. Выявить точку переобучения;
4. Применить перекрестную проверку по K блокам при различных K;
5. Построить графики ошибки и точности во время обучения для моделей, а также усредненные графики по всем моделям.

Ход работы.

Была построена нейронная сеть, код которой представлен в приложении А.

Сначала происходит нормализация данных, после чего применяется перекрестная проверка по k блокам.

После запуска программы были получены графики оценки средней абсолютной ошибки (MAE). Графики, полученные при перекрестной проверки, представлены на рис. 1-4. На рис. 5 усредненная оценка для всех моделей.

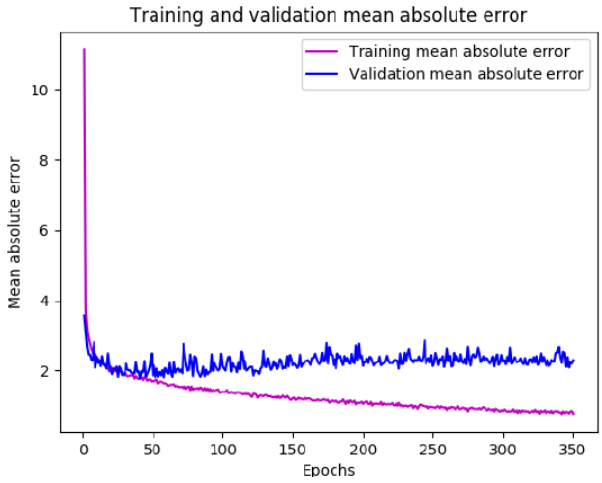

Рисунок 1 – График оценки MAE, k = 1

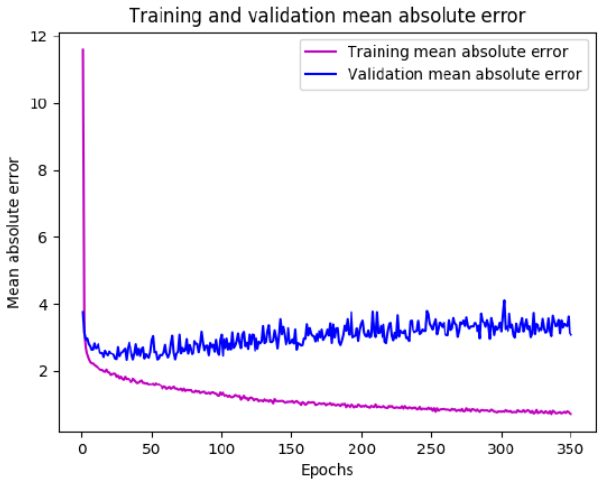

Рисунок 2 – График оценки MAE, k = 2

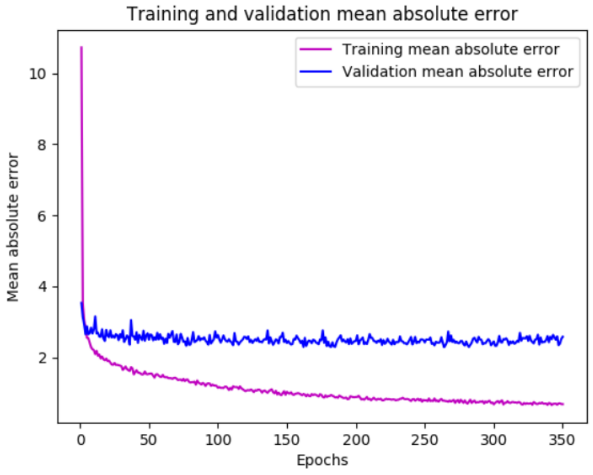

Рисунок 3 – График оценки MAE, k = 3

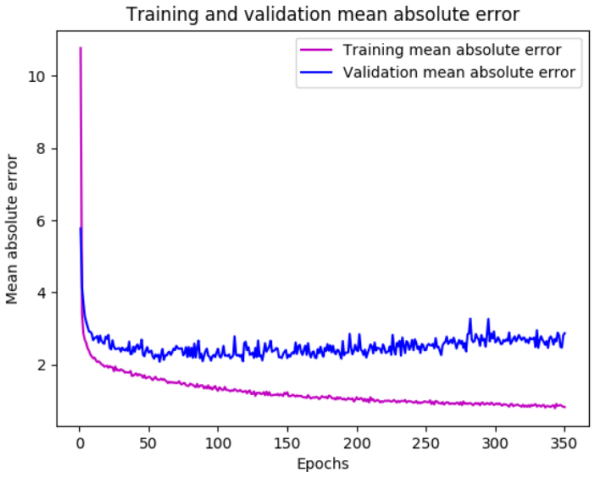

Рисунок 4 – График оценки MAE, k = 4

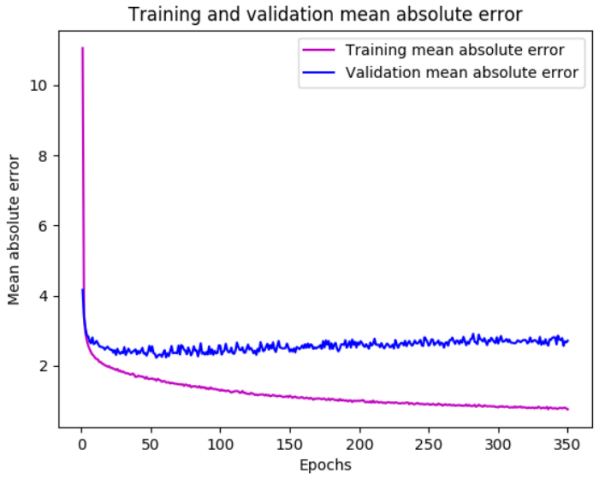

Рисунок 5 – График оценки средней MAE

Можно заметить, что при k = 2 переобучение наступает в точке 20, при k = 4 – в точке 40. Что свидетельствует о том, что перекрестная проверка по К блокам является один из средств снижения возможности переобучения..

Выводы.

В ходе выполнения лабораторной работы были выявлены различия задач классификации и регрессии: задача классификации – используется для нечисловых данных, сможет ли человек выплатить кредит, в результате получает множество значений, определяющих принадлежность к тому или иному классу, задача регрессии – в основном используется для числовых данных, когда нужно предсказать, вероятных заработок человека с данной профессией, в результате – единственное значение, выражение количества. Была выявлена точка переобучения: 40 эпох. Была применена перекрестная проверка по К блокам при различных К, что объясняет: при увеличении К переобучение должно стремиться к минимуму.

ПРИЛОЖЕНИЕ А

```
import numpy as np
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.datasets import boston_housing
import matplotlib.pyplot as plt

def build_model():
    model = Sequential()
    model.add(Dense(64, activation='relu', input_shape=(train_data.shape[1],)))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(1))
    model.compile(optimizer='rmsprop', loss='mse', metrics=['mae'])
    return model

def plot_show(epochs, mae, loss, val_mae, val_loss):
    print(len(mae), len(loss), len(val_mae), len(val_loss))
    plt.plot(epochs, loss, 'm', label='Training loss')
    plt.plot(epochs, val_loss, 'b', label='Validation loss')
    plt.title('Training and validation loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
    plt.clf()
    plt.plot(epochs, mae, 'm', label='Training mean absolute error')
    plt.plot(epochs, val_mae, 'b', label='Validation mean absolute error')
    plt.title('Training and validation mean absolute error')
    plt.xlabel('Epochs')
    plt.ylabel('Mean absolute error')
    plt.legend()
    plt.show()

(train_data, train_targets), (test_data, test_targets) = boston_housing.load_data()
print(train_data.shape)
print(test_data.shape)
print(test_targets)
mean = train_data.mean(axis=0)
train_data -= mean
std = train_data.std(axis=0)
train_data /= std
test_data -= mean
test_data /= std

k = 2
num_val_samples = len(train_data) // k
num_epochs = 350
all_scores = []
arr_loss = []
arr_mae = []
arr_val_loss = []
arr_v_al_mae = []
for i in range(k):
    print('processing fold #', i)
    val_data = train_data[i * num_val_samples: (i + 1) * num_val_samples]
    val_targets = train_targets[i * num_val_samples: (i + 1) * num_val_samples]
    partial_train_data = np.concatenate([train_data[:i * num_val_samples],
train_data[(i + 1) * num_val_samples:]],
                                        axis=0)
    partial_train_targets = np.concatenate(
```

```
        [train_targets[:i * num_val_samples], train_targets[(i + 1) *
num_val_samples:]], axis=0)
    model = build_model()
    history = model.fit(partial_train_data, partial_train_targets, epochs=num_epochs,
batch_size=1, validation_data = (val_data, val_targets))
    loss = history.history['loss']
    arr_loss.append(loss)
    mae = history.history['mae']
    arr_mae.append(mae)
    val_loss = history.history['val_loss']
    arr_val_loss.append(val_loss)
    v_al_mae = history.history['val_mae']
    arr_v_al_mae.append(v_al_mae)
    val_mse, val_mae = model.evaluate(val_data, val_targets, verbose=0)
    all_scores.append(val_mae)
    plot_show(range(1, num_epochs + 1), mae, loss, v_al_mae, val_loss)
plot_show(range(1, num_epochs + 1), np.mean(arr_mae, axis = 0), np.mean(arr_loss,
axis = 0), np.mean(arr_v_al_mae, axis = 0), np.mean(arr_val_loss, axis = 0))
print(np.mean(all_scores))
```