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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Искусственные нейронные сети»

Тема: Регрессионная модель изменения цен на дома в Бостоне

Студент гр. 8382 _____ Гордиенко А.М.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы.

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.

Данный набор содержит относительно немного образцов данных: всего 506, разбитых на 404 обучающих и 102 контрольных образца. И каждый признак во входных данных (например, уровень преступности) имеет свой масштаб. Например, некоторые признаки являются пропорциями и имеют значения между 0 и 1, другие — между 1 и 12 и т. д.

Порядок выполнения работы.

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 эпох 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.

Основные теоретические положения.

Набор данных присутствуют в составе Keras.

Листинг 1 – Подключение модулей

```
import numpy as np
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.datasets import boston_housing
(train_data, train_targets), (test_data, test_targets) = boston_housing.load_data()
print(train_data.shape)
print(test_data.shape)
print(test_targets)
```

404 обучающих и 102 контрольных образца, каждый с 13 числовыми признаками.

Цены в основном находятся в диапазоне от 10 000 до 50 000 долларов США.

Было бы проблематично передать в нейронную сеть значения, имеющие самые разные диапазоны. Сеть, конечно, сможет автоматически адаптироваться к таким разнородным данным, однако это усложнит обучение. На практике к таким данным принято применять нормализацию: для каждого признака во входных данных (столбца в матрице входных данных) из каждого значения вычитается среднее по этому признаку, и разность делится на стандартное отклонение, в результате признак центрируется по нулевому значению и имеет стандартное отклонение, равное единице. Такую нормализацию легко выполнить с помощью Numpy.

Листинг 2 – Обработка данных

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

Листинг 3 – Определение функции build_model()

```
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
```

Сеть заканчивается одномерным слоем, не имеющим функции активации (это линейный слой). Это типичная конфигурация для скалярной регрессии (целью которой является предсказание одного значения на непрерывной числовой прямой). Применение функции активации могло бы ограничить диапазон выходных значений: например, если в последнем слое применить функцию активации sigmoid, сеть обучилась бы предсказывать только значения из диапазона между 0 и 1.

В данном случае, с линейным последним слоем, сеть способна предсказывать значения из любого диапазона.

Обратите внимание на то, что сеть компилируется с функцией потерь mse — mean squared error (среднеквадратичная ошибка), вычисляющей квадрат разности между предсказанными и целевыми значениями. Эта функция широко используется в задачах регрессии. Также добавлен новый параметр на этапе обучения: mae — mean absolute error (средняя абсолютная ошибка). Это абсолютное значение разности между предсказанными и целевыми значениями. Например, значение MAE, равное 0,5, в этой задаче означает, что в среднем прогнозы отклоняются на 500 долларов США.

Чтобы оценить качество сети в ходе корректировки ее параметров (таких, как количество эпох обучения), можно разбить исходные данные на обучающий и проверочный наборы, как это делалось в предыдущих примерах. Однако так как у нас и без того небольшой набор данных, проверочный набор получился бы

слишком маленьким (скажем, что-нибудь около 100 образцов). Как следствие, оценки при проверке могут сильно меняться в зависимости от того, какие данные попадут в проверочный и обучающий наборы: оценки при проверке могут иметь слишком большой разброс. Это не позволит надежно оценить качество модели.

Хорошей практикой в таких ситуациях является применение перекрестной проверки по K блокам (K-fold cross-validation). Суть ее заключается в разделении доступных данных на K блоков (обычно K = 4 или 5), создании K идентичных моделей и обучении каждой на K—1 блоках с оценкой по оставшимся блокам. По полученным K оценкам вычисляется среднее значение, которое принимается как оценка модели. В коде такая проверка реализуется достаточно просто.

Листинг 4 – Оценка качества сети

```
k = 4
num_val_samples = len(train_data) // k
num_epochs = 100
all_scores = []
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()
            model.fit(partial_train_data, partial_train_targets,
epochs=num_epochs, batch_size=1, verbose=0)
       val_mse, val_mae = model.evaluate(val_data, val_targets,
verbose=0)
```

```
        all_scores.append(val_mae)
print(np.mean(all_scores))
```

Разные прогоны действительно показывают разные оценки, от 2,6 до 3,2. Средняя (3,0) выглядит более достоверно, чем любая из оценок отдельных прогонов, — в этом главная ценность перекрестной проверки по K блокам. В данном случае средняя ошибка составила 3000 долларов, что довольно много, если вспомнить, что цены колеблются в диапазоне от 10 000 до 50 000 долларов.

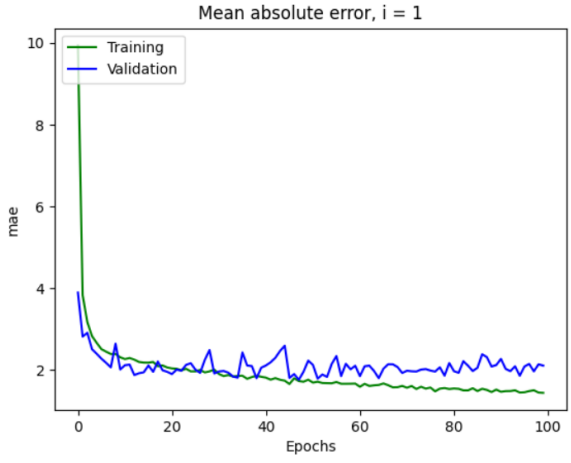
- Необходимо уменьшить или увеличить количество эпох обучения и проанализировать полученные результаты.

Ход работы.

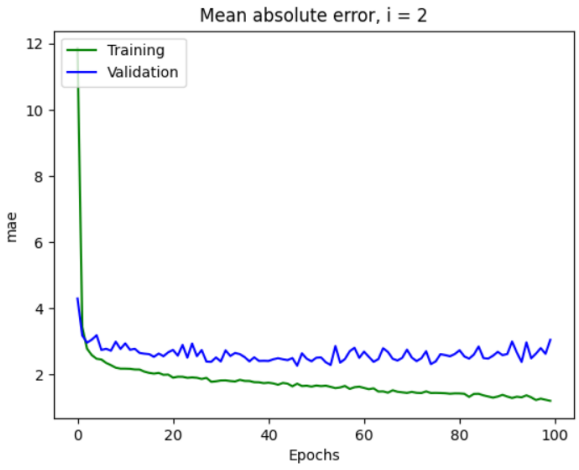
Задача классификации сводится к определению класса объекта по его характеристикам. Необходимо заметить, что в этой задаче множество классов, к которым может быть отнесен объект, заранее известно.

Задача регрессии, подобно задаче классификации, позволяет определить по известным характеристикам объекта значение некоторого его параметра. В отличие от задачи классификации значение параметра является не конечное множество классов, а множество действительных чисел.

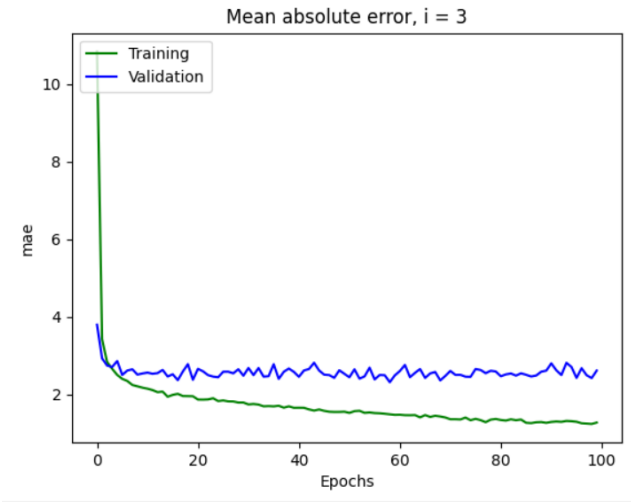
Посмотрим на результаты нейронной сети на данных по умолчанию – на 4 блоках и 100 эпохах. Графики представлены на рис. 1, 2.

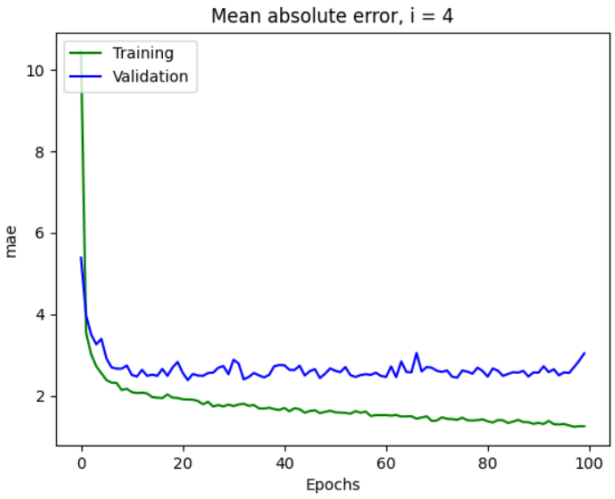


а



б





в г

Рисунок 1 – График оценки MAE для блока а – 1, б – 2, в – 3, г – 4.

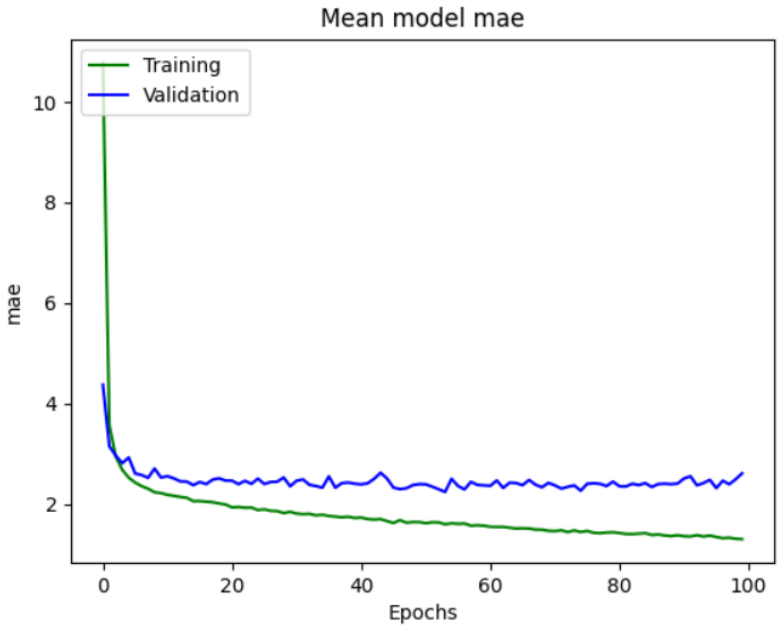


Рисунок 2 – График среднего значения MAE.

Заметим, что оценки MAE на тестовых данных начинают возрастать после 20 эпохи, значит следует убавить количество эпох по этого значения во избежание переобучения.

Рассмотрим модели с 20 эпохами на 2, 4, 6 и 8 блоках. Графики представлены на рис. 3.

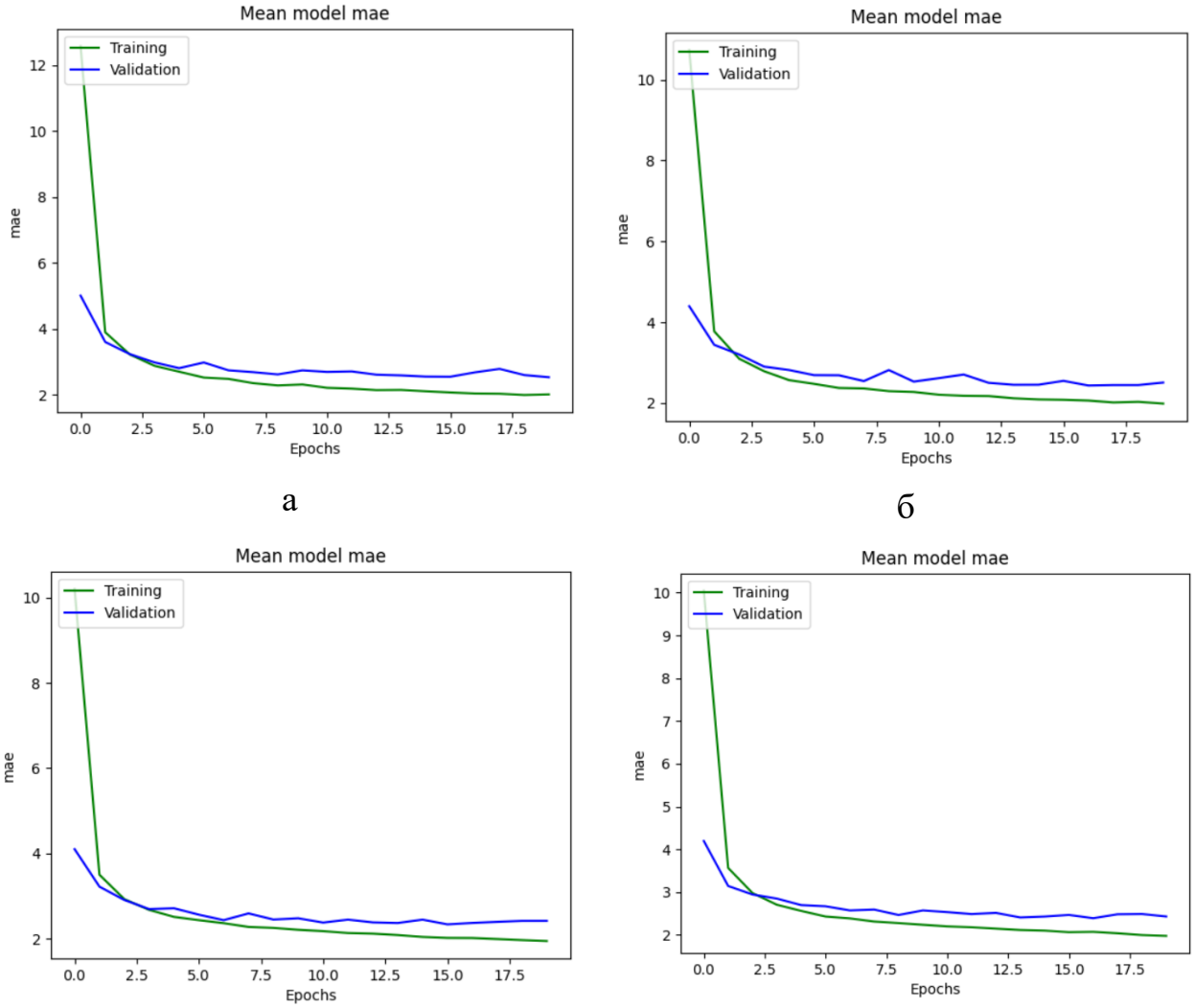


Рисунок 3 – Графики среднего значения MAE для модели с количеством блоков: а – 2, б – 4, в – 6, г – 8.

Из графиков видим, что наилучшей сходимостью и наименьшей средней ошибкой обладает модель с 6 блоками.

Выводы.

В ходе выполнения лабораторной работы была изучена задача регрессии и ее отличие от задачи классификации с помощью библиотеки Keras. Также было изучено влияние количества эпох и числа блоков на результат обучения сети.