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

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине «Искусственные нейронные сети»

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

Студент гр. 7383 _____ Ханова Ю. А.

Преподаватель _____ Жукова Н. А.

Санкт-Петербург

2020

Цели.

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.

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

Ход работы.

В задачи регрессии предсказывается некоторая характеристика объекта, значения которой не ограничены, тогда как в задаче классификации предсказывается принадлежность объекта к одному из заданных классов, причем набор значений ограничен.

Изначально была рассмотрена модель с перекрестной проверкой на 3 блоках и со 100 эпохами. На рис.1 представлены графики оценки MAE для каждого блока, на рис.2 представлен график среднего значения MAE.

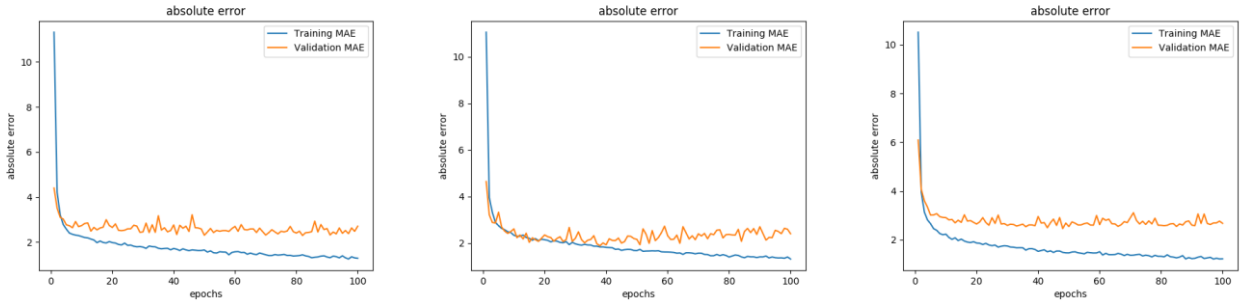

Рисунок 1 – Графики оценки MAE для каждого блока

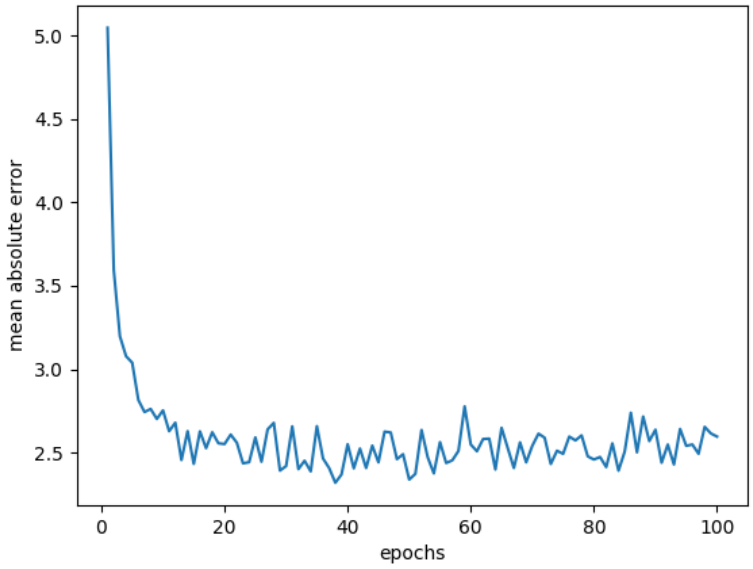

Рисунок 2 – График средних значений MAE

Как видно из рисунков примерно после 10 эпохи значение MAE на проверочных данных начинает возрастать, в то время как на тестовых данных продолжает уменьшаться – это свидетельствует о переобучении сети.

Рассмотрим влияние количества блоков перекрестной проверки на сеть при 10 эпохах. На рис.3-6 представлены графики средних значений MAE на 2, 4, 6 и 8 блоках:

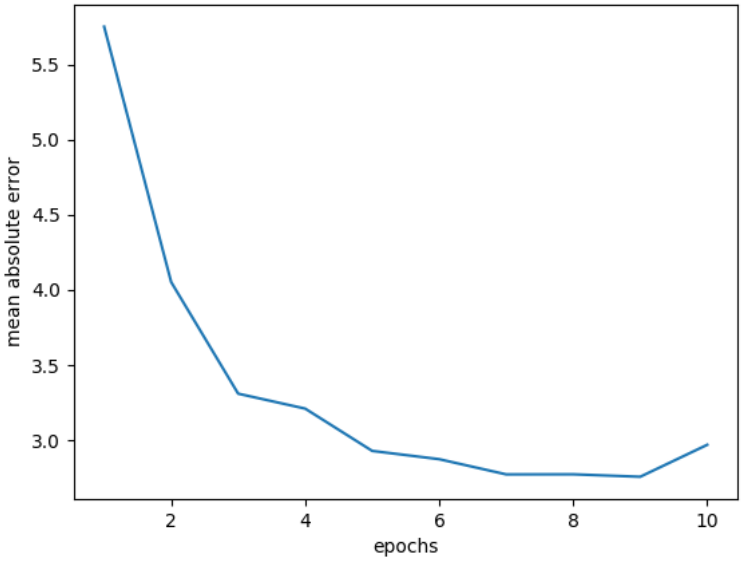

Рисунок 3 – Среднее значение на 2 блоках

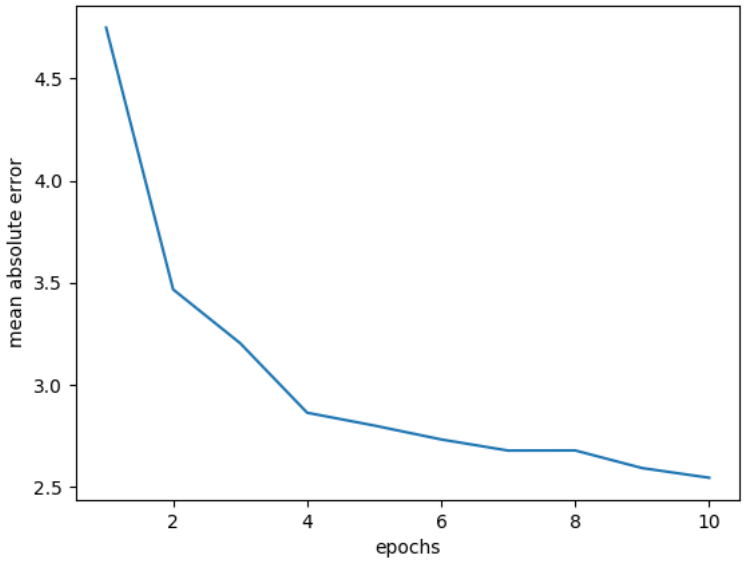

Рисунок 4 – Среднее значение на 4 блоках

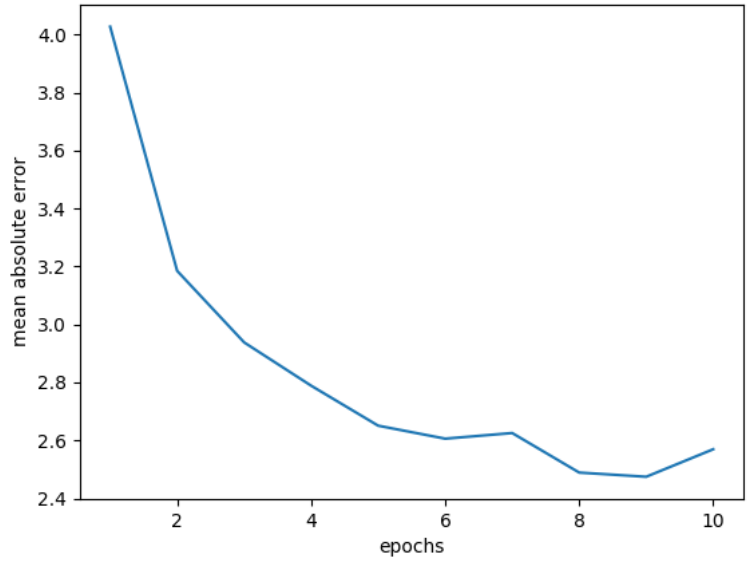

Рисунок 5 – Среднее значение на 6 блоках

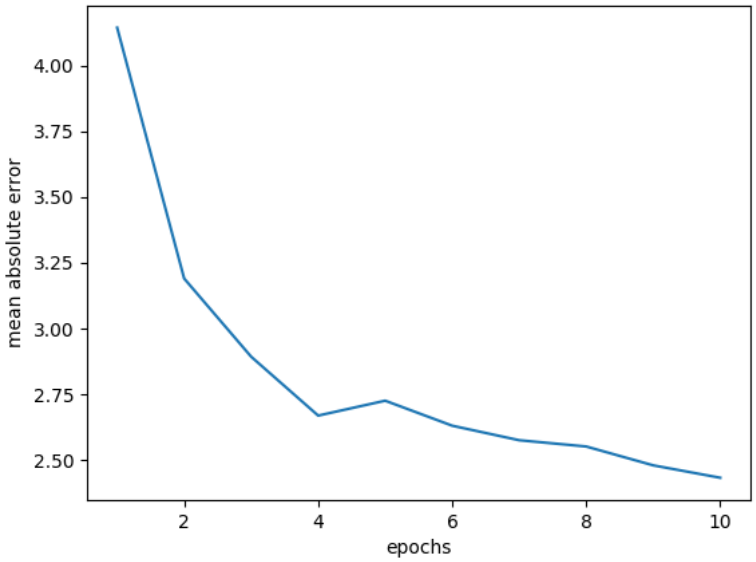

Рисунок 6 – Среднее значение на 8 блоках

Из графиков можем определить, что худший результат получен на модели с 2 блоками, а наилучшая работа сети наблюдается на 4 блоках перекрестной проверки.

Вывод.

В ходе выполнения данной работы была изучена задача регрессии и ее отличие от задачи классификации. Была изучена и проведена перекрестная проверка модели. Код программы представлен в приложении А.

Приложения

Приложение А

```
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential

from tensorflow.keras.datasets import boston_housing

import numpy as np
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


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

k = 8
num_val_samples = len(train_data) // k
num_epochs = 10
mae_histories = []
for i in range(k):
    print(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_target = 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_target, epochs=num_epochs, batch_size=1,
                        validation_data=(val_data,
val_targets))
    mae = history.history['mae']
    v_mae = history.history['val_mae']
    x = range(1, num_epochs + 1)
    mae_histories.append(v_mae)
    plt.figure(i + 1)
    plt.plot(x, mae, label='Training MAE')
    plt.plot(x, v_mae, label='Validation MAE')
    plt.title('absolute error')
    plt.ylabel('absolute error')
    plt.xlabel('epochs')
    plt.legend()

average_mae_history = [np.mean([x[i] for x in mae_histories])
for i in range(num_epochs)]
plt.figure(0)
plt.plot(range(1, num_epochs + 1), average_mae_history)
plt.xlabel('epochs')
plt.ylabel("mean absolute error")
figs = [plt.figure(n) for n in plt.get_fignums()]
for i in range(len(figs)):
        figs[i].savefig("5Graphics%d.png" %(i), format='png')
```