

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. 8383

Сырцова Е.А.

Преподаватель

Жангиров Т.Р.

Санкт-Петербург

2021

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.

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

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

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 4 блоках и со 100 эпохами. На рис.1 представлены графики оценки MAE для каждого блока, на рис.2 представлен график среднего значения MAE.

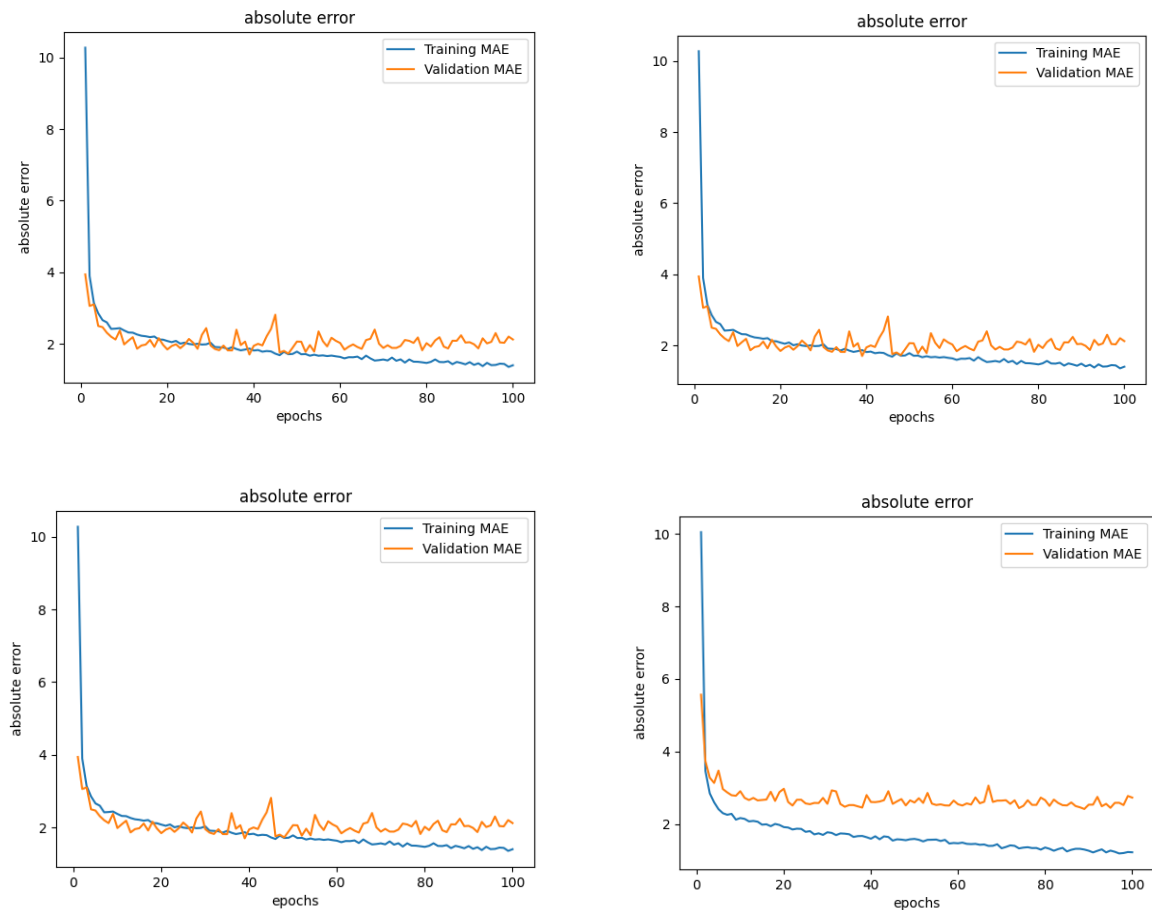


Рисунок 1 – Графики оценки MAE для каждого блока

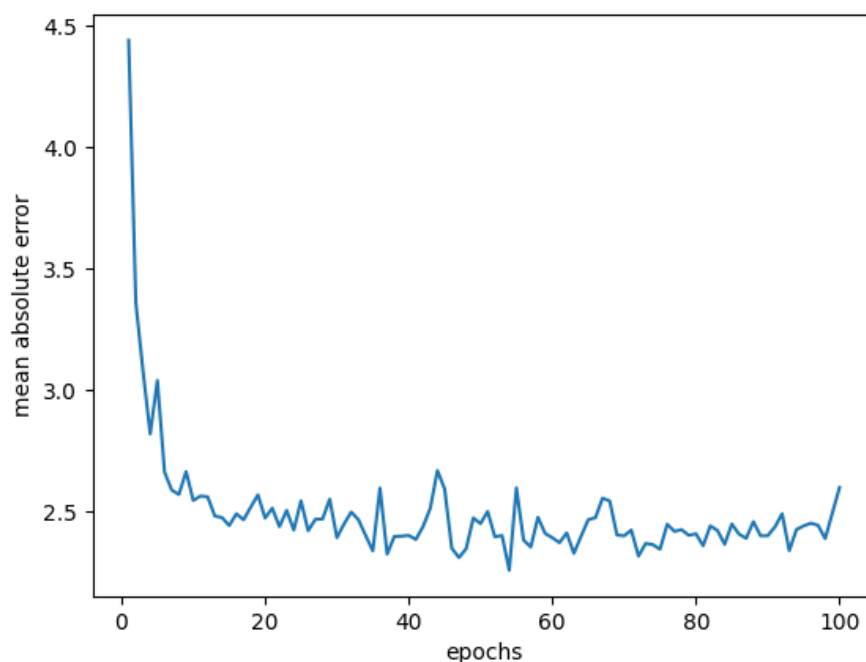


Рисунок 2 – График среднего значения MAE

Исходя из результатов нескольких запусков и анализа графиков оценки MAE можно сделать вывод, что примерно после десяти эпох значение MAE на проверочных данных начинает возрастать, в то время как на тестовых данных продолжает уменьшаться, что свидетельствует о переобучении сети. Значит следует убавить значение количества эпох до 10 во избежание переобучения 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 соответственно:

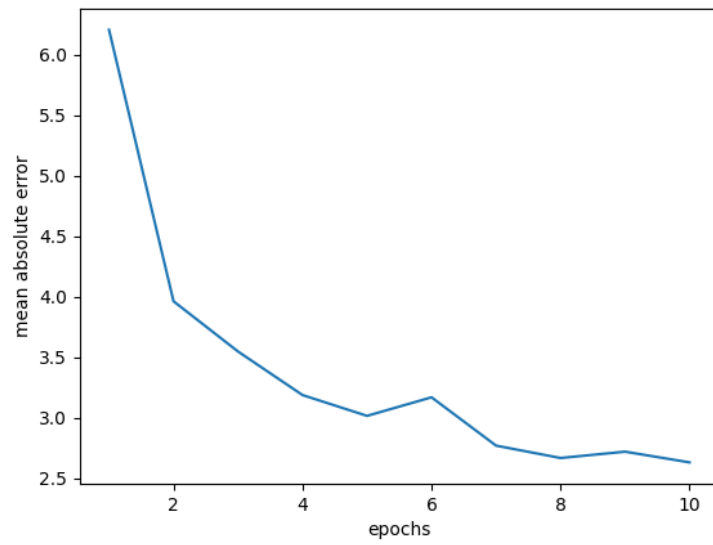


Рисунок 3 – График среднего значения MAE на 2 блоках

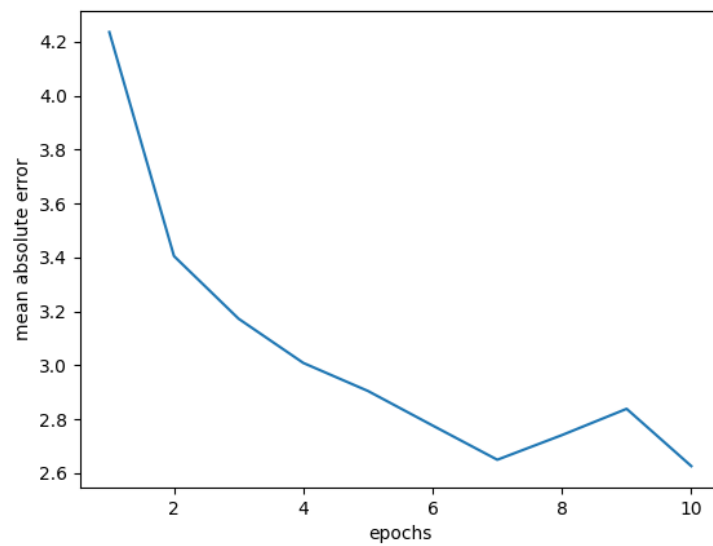


Рисунок 4 – График среднего значения MAE на 4 блоках

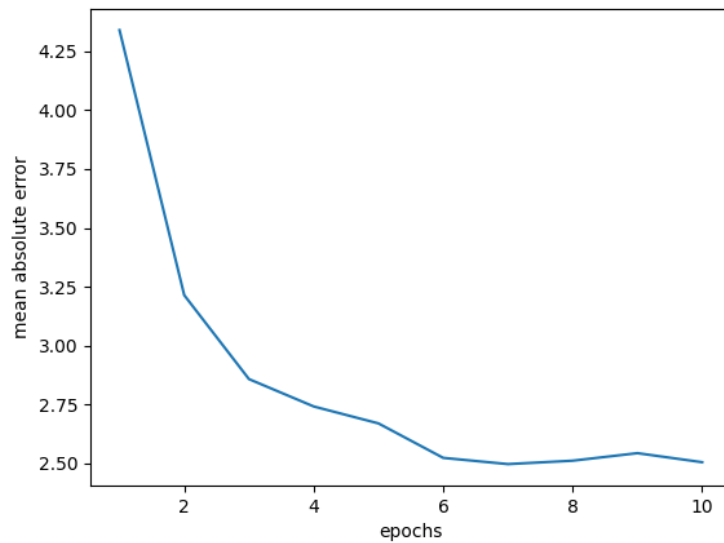


Рисунок 5 – График среднего значения MAE на 6 блоках

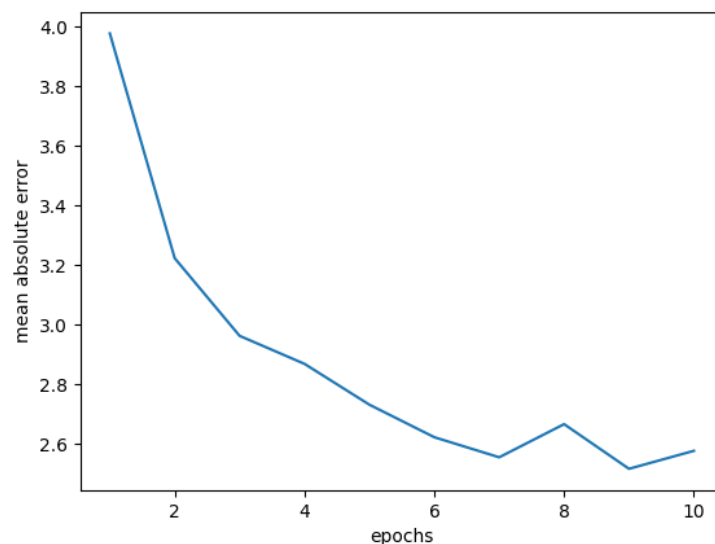


Рисунок 6 – Графики среднего значения MAE на 8 блоках

Из графиков можем определить, что наилучшей сходимостью и наименьшей средней ошибкой обладает модель на 6 блоках

Вывод

В процессе 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.

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

(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 = 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()
    history = model.fit(partial_train_data, partial_train_targets,
epochs=num_epochs, batch_size=1, validation_data=(val_data, val_targets))
    plt.figure(i + 1)
```

```

mae = history.history['mean_absolute_error']
v_mae = history.history['val_mean_absolute_error']
x = range(1, num_epochs + 1)
all_scores.append(v_mae)
print(np.mean(all_scores))
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 all_scores]) 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].show()
    figs[i].savefig("1_%d.png" % (i), format='png')

```