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. 7383 _____ Бергалиев М.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

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.

Порядок выполнения работы.

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

Ход работы.

0. Исходный код программы представлен в приложении.

1. В задаче классификации предсказывается принадлежность к одному из нескольких классов. Предсказанное значение является качественным, т.е. принадлежит ограниченному набору значений. В задаче регрессии же предсказывается количественная переменная, набор значений которой бесконечен.

2. Изучим влияние количества эпох на результат обучения. Точность будем оценивать с помощью средней абсолютной ошибки. Графики ошибок и точности показаны на рис. 1, 2.

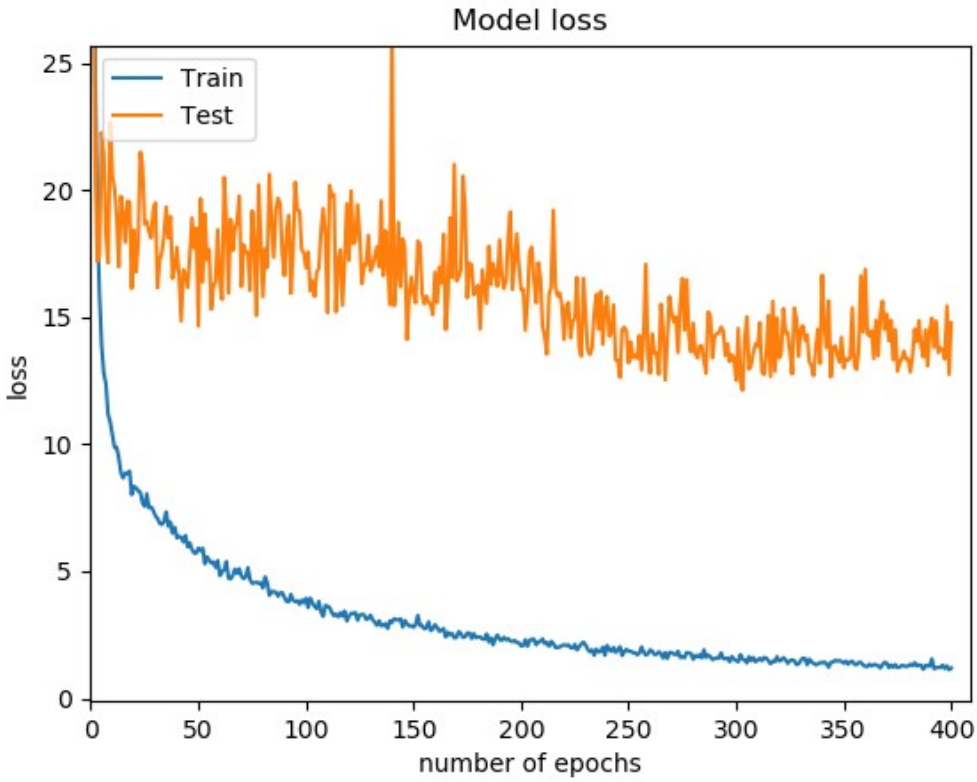

Рисунок 1 — Ошибки в зависимости от числа эпох

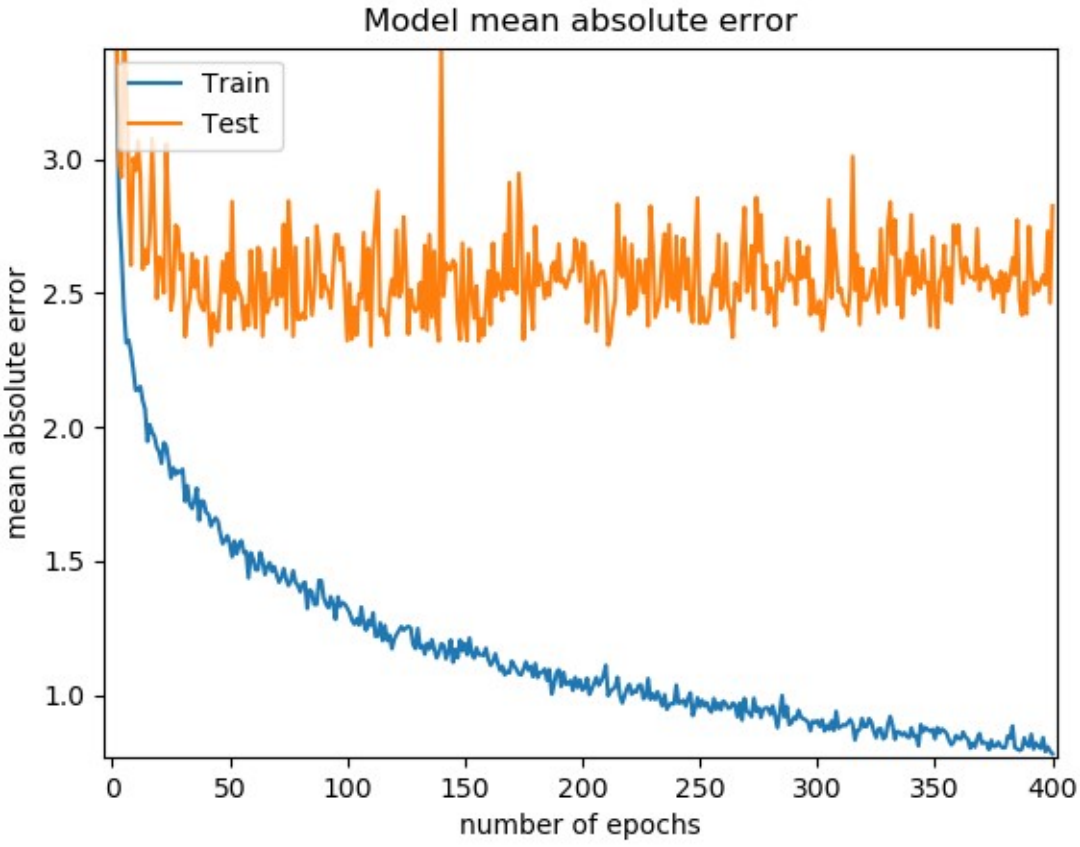

Рисунок 2 — Зависимость средней абсолютной ошибки от числа эпох

Как видно, после 50ти эпох ошибки и точность на проверочных данных перестают улучшаться.

3. Точка переобучения находится там, где ошибки и точность на проверочных данных перестают улучшаться, когда как на обучающих данных продолжают улучшаться. Судя по средней абсолютной ошибке, переобучение начинается после 50 эпох.

4. Проведем перекрестную проверку, меняя число блоков K. График средней точности в зависимости от числа блоков показан на рис. 3.

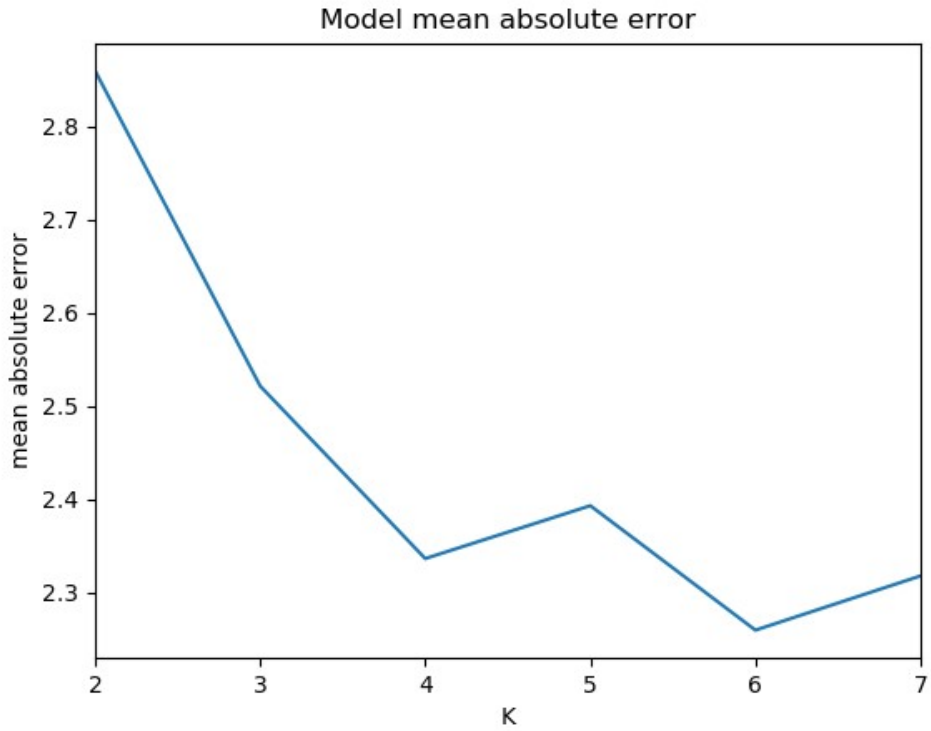

Рисунок 3 — Зависимость средней точности от числа блоков K при перекрестной проверке

Как видно, с ростом числа блоков точность увеличивается. Однако скорость увеличения точности уменьшается, поэтому при 4 блоках достигается оптимальное соотношение точности и затрат времени на перекрестную проверку.

Выводы: Было изучено влияние числа эпох на результат обучения в задаче регрессии, найдена точка переобучения. Переобучение возникает после 50 эпох. Была проведена перекрестная проверка модели, изучена зависимость результатов проверки от числа блоков.

ПРИЛОЖЕНИЕ

```
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
import matplotlib.pyplot as plt
import pylab

(train_data, train_targets), (test_data, test_targets) =
boston_housing.load_data()

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

test_data -= mean
test_data /= std

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 _test(epochs):
    loss = []
    mae = []
    v_loss = []
    v_mae = []
    model = build_model()
    history = model.fit(train_data, train_targets, epochs=epochs,
                         validation_data=(test_data, test_targets),
batch_size=1)
    loss = history.history['loss']
    mae = history.history['mae']
    v_loss = history.history['val_loss']
    v_mae = history.history['val_mae']
    x = range(1, epochs+1)

    plt.plot(x, loss)
```

```
    plt.plot(x, v_loss)
    plt.title('Model loss')
    plt.ylabel('loss')
    plt.xlabel('number of epochs')
    plt.xlim(x[19], x[-1])
    plt.ylim(min(loss[19:]), max(v_loss[19:]))
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()

    plt.plot(x, mae)
    plt.plot(x, v_mae)
    plt.title('Model mean absolute error')
    plt.ylabel('mean absolute error')
    plt.xlabel('number of epochs')
    plt.xlim(x[19], x[-1])
    plt.ylim(min(mae[19:]), max(v_mae[19:]))
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()


def cross_check(k_list):
    k = 4
    all_scores = []
    for k in k_list:
        num_val_samples = len(train_data) // k
        all_scores.append([])
        for i in range(0, 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 = best_model()
               model.fit(partial_train_data, partial_train_targets,
epochs=100, batch_size=1, verbose=0)
                        val_mse,  val_mae  =  model.evaluate(val_data,
val_targets, verbose=0)
            all_scores[-1].append(val_mae)
        print(np.mean(all_scores[-1]))
```

```
plt.plot(k_list, [np.mean(i) for i in all_scores])
plt.title('Model mean absolute error')
plt.ylabel('mean absolute error')
plt.xlabel('K')
plt.xlim(k_list[0], k_list[-1])
pylab.xticks(k_list)
plt.show()
```