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. 7382 _____ Государкин Я.С.

Преподаватель _____ Жангиров Т.Р

Санкт-Петербург

2020

Задание

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.

Теоретическая часть

Задача регрессии в нейронных сетях сводится к тому, чтобы имея выборку данных, которая состоит из:

- признаков (вектор независимых переменных)
- отклика (зависимой переменной)

с помощью целевой функции приближения (ф-я потерь, обычно для регрессии — MSE) как можно более точно предсказывать значения зависимых переменных (какое-то действительное число, например, предсказывая курс доллара к рублю это число 82.96) на неизвестном сети наборе данных.

Отличие от задачи классификации в том, что нет необходимости использовать нормирующие функции активации последнем слое, которые ограничивают область значений, поскольку в контексте задачи регрессии работа НС — выдать наиболее точный отклик (т.е когда ошибка будет стремиться к нулю)

Подготовка

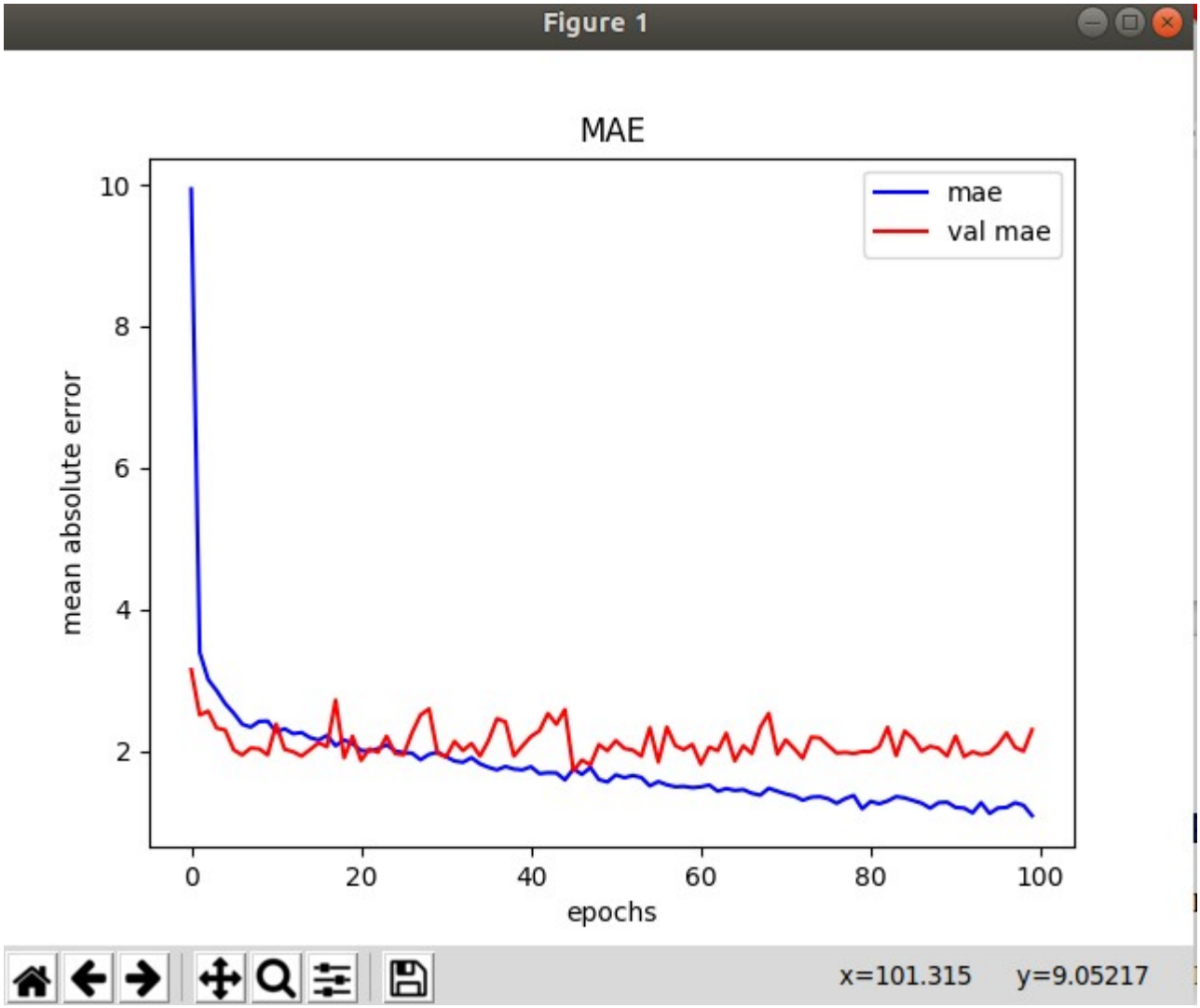
Для исследования были выбраны следующие настройки для обучения НС:

- **Эпохи (epochs)** : 100, 1000
- Кол-во перекрёстных прогонов: 2, 4, 8, 16
- **Размер тренировочного пакета данных (batch size):** 1
- **Слои (скрытые): 2** (64 нейрона)
- **Функция активации скрытых слоёв:** ReLU
- **Функции потерь:** mse
- **Метрика:** mae
- **Метод оптимизации:** ADAM: learning_rate=0.001, beta_1=0.9, beta_2=0.999

Эксперименты

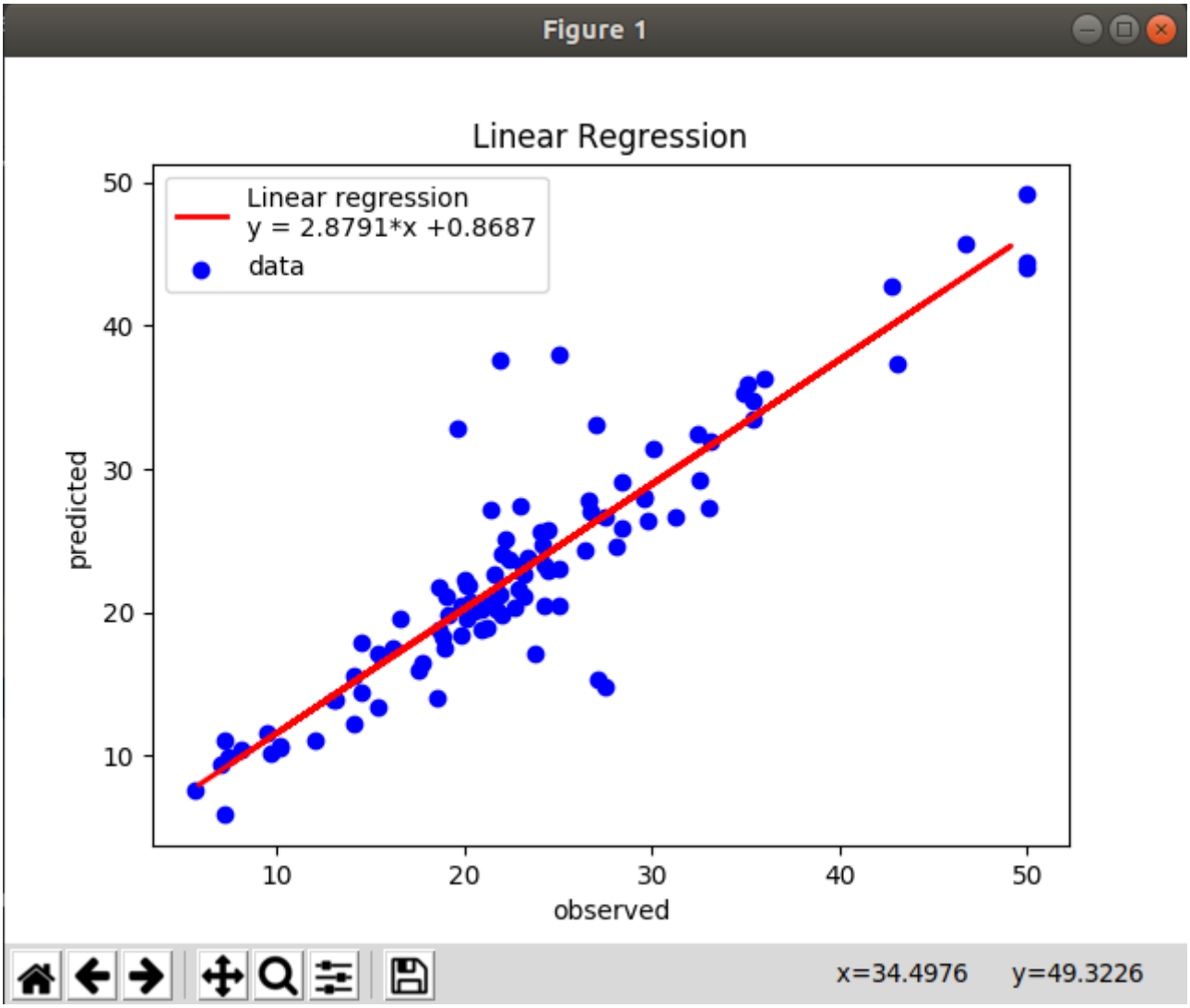
Конфигурация 1:

- Кол-во перекрёстных прогонов: 4
- Кол-во эпох: 100



Это конфигурация, предлагаемая в условии ЛР. Средняя абсолютная ошибка по 4 проходам — 2.237342, что в пересчёте на деньги означает ~ 2240 $

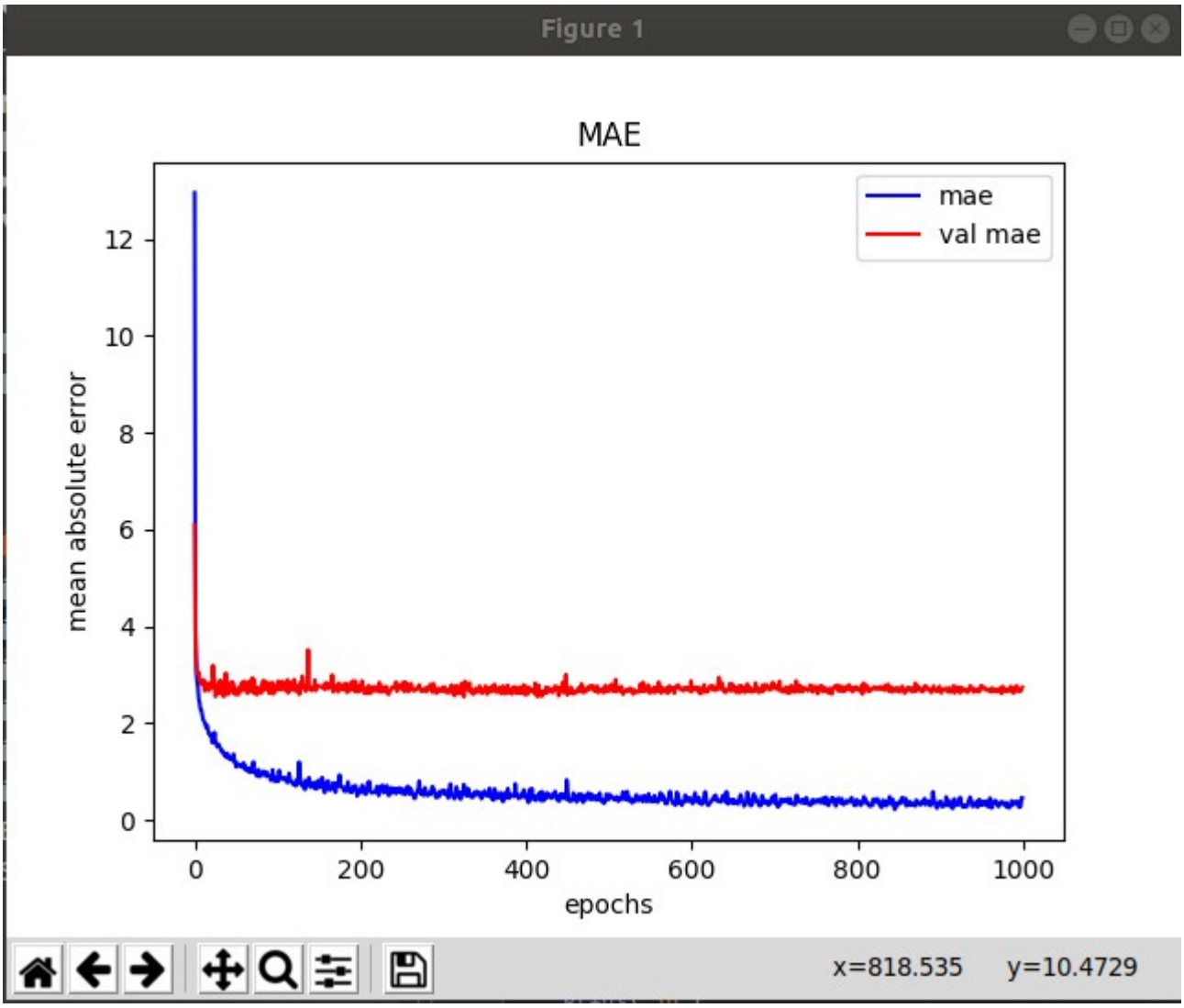
Выше — график абсолютной ошибки, ниже — график линейной регрессии.



В целом, прямая хорошо описывает данный датасет, есть небольшая дисперсия и девиации.

Конфигурация 2:

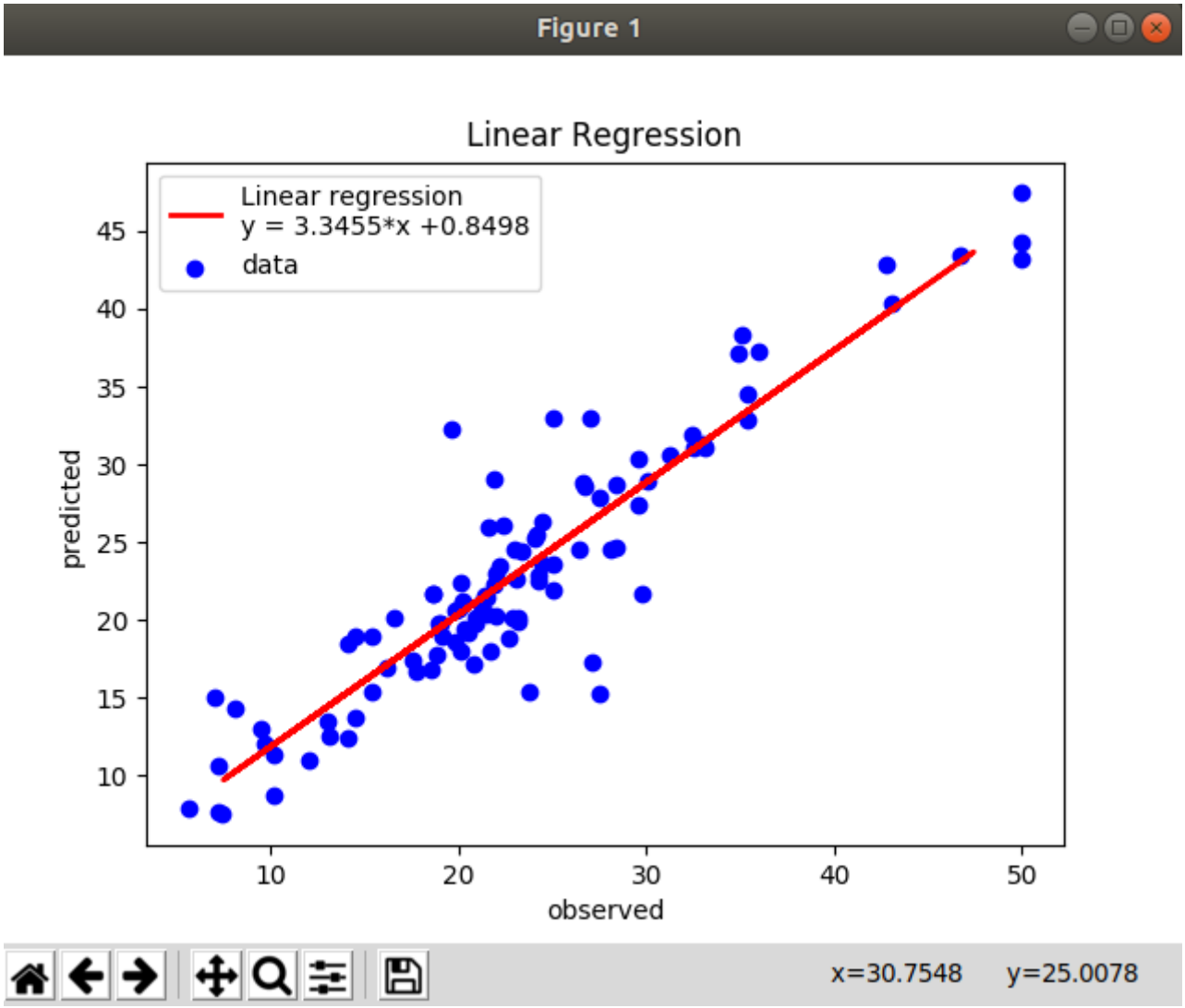
- Кол-во перекрёстных прогонов: 4
- Кол-во эпох: 1000



Увеличив количество эпох в 10 ничего кардинально не поменялось. Несколько видимых отличий:

- Появилась заметная дисперсия ошибки
- Точность в целом на обучающей выборке стала выше, однако явное переобучение налицо

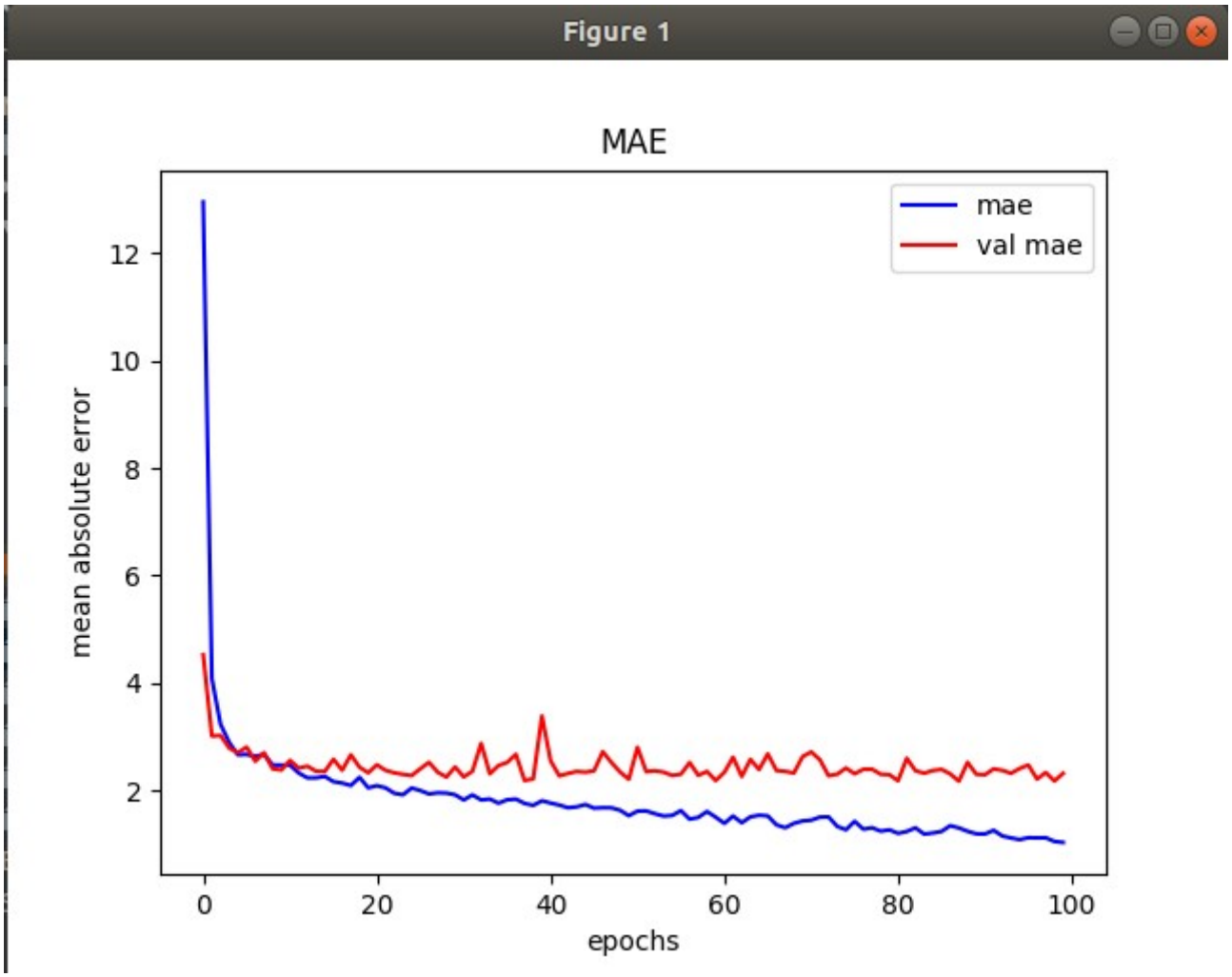
MAE — 2370 $, что выше чем в 1 конфигурации. Предположительно из-за увеличенной дисперсии.

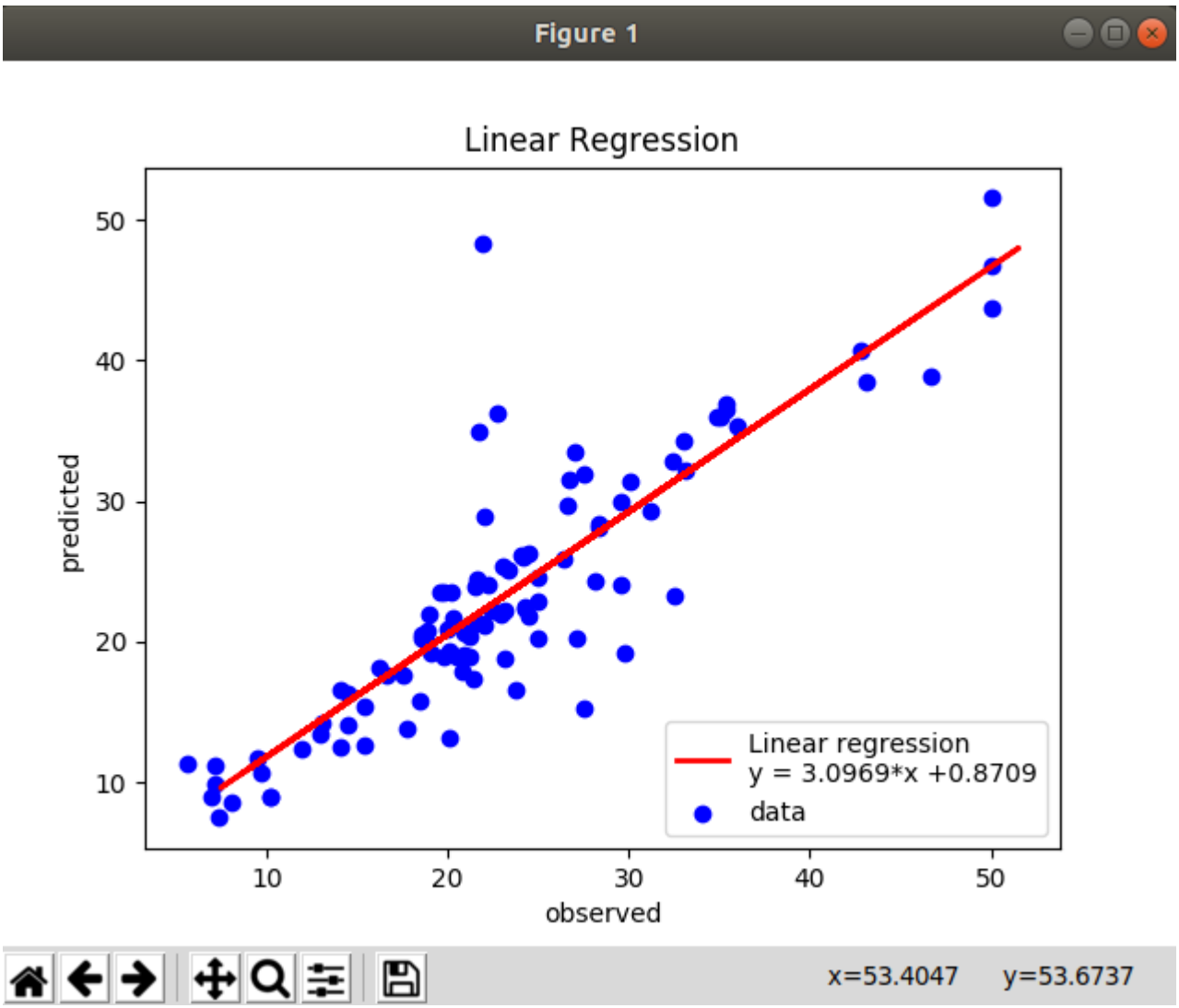


Как видно на скрине выше, дисперсия ошибки у большого количества точек — ощутимая.

Конфигурация 3:

- Кол-во перекрёстных прогонов: 2
- Кол-во эпох: 100





При таких сетапах MAE составила ~ 2800 $, что больше чем раньше. Скорее всего это потому что мало перекрёстных проходов, поэтому результат так отклонён.

Конфигурация 4:

- Кол-во перекрёстных прогонов: 8,16
- Кол-во эпох: 100

Если увеличить число прогонов до 8, то MAE составит ~ 2450 $.

Увеличив до 16 получим ~ 2340 $

Выводы.

В ходе лабораторной работы мы ознакомились с задачей регрессии, как она решается с помощью нейронных сетей, в чём особенности архитектуры сетей, которые используются для подобных задач.

Было выяснено, что перекрестные проверки дают хороший средний показатель абсолютной ошибки на небольших датасетах. Как повлияло увеличение кол-ва эпох сказать сложно. У конфига с 1000 эпох MAE ближе к среднему при 16 проверках, чем у конфига со 100 эпохами.

P.S По поводу словосочетания «точка переобучения». Не уверен, что такой термин существует (гугл тоже не нашёл ни в русскому сегменте ни в английском), как я понял, тут имеется ввиду точка баланса между bias^2 и дисперсией, где отклонение наименьшее. Если так, то в моём случае это:

k: ~ 4-5, epochs: ~ 20-30

ПРИЛОЖЕНИЕ А

```
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 sklearn.metrics, math
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt;
(train_data, train_targets), (test_data, test_targets) = boston_housing.load_data()
acc_sq = []
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 = 16
num_val_samples = len(train_data) // k
num_epochs = 100
all_scores = []
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='adam', loss='mse', metrics=['mae'])
    return model
def write_graphics(H, y_pred):
    regressor = LinearRegression()
    regressor.fit(test_targets.reshape(-1, 1), y_pred)
    y_fit = regressor.predict(y_pred)
    reg_intercept = round(regressor.intercept_[0], 4)
    reg_coef = round(regressor.coef_.flatten()[0], 4)
    reg_label = "y = " + str(reg_intercept) + "*x +" + str(reg_coef)
    print("\n")
    print("Mean absolute error (MAE):      %f" %
sklearn.metrics.mean_absolute_error(test_targets, y_pred))
    print("Mean squared error (MSE):       %f" %
sklearn.metrics.mean_squared_error(test_targets, y_pred))
    regressor = LinearRegression()
    regressor.fit(test_targets.reshape(-1,1), y_pred)
    y_fit = regressor.predict(y_pred)
    reg_intercept = round(regressor.intercept_[0],4)
    reg_coef = round(regressor.coef_.flatten()[0],4)
    reg_label = "y = " + str(reg_intercept) + "*x +" + str(reg_coef)
    plt.scatter(test_targets, y_pred, color='blue', label= 'data')
    plt.plot(y_pred, y_fit, color='red', linewidth=2, label = 'Linear regression\
n'+reg_label)
    plt.title('Linear Regression')
    plt.legend()
    plt.xlabel('observed')
    plt.ylabel('predicted')
    plt.show()
    plt.plot(H.history['mean_absolute_error'], color='blue', label='test')
    plt.title('MAE')
    plt.legend()
    plt.xlabel('epochs')
    plt.ylabel('mean absolute error')
    plt.show()
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()
    H = model.fit(partial_train_data, partial_train_targets, epochs=num_epochs,
batch_size=1, verbose=0)
    pred = model.predict(test_data)
    write_graphics(H, pred)
    val_mse, val_mae = model.evaluate(val_data, val_targets, verbose=0)
    all_scores.append(val_mae)
print(np.mean(all_scores))
```