## CSEE 5590 0001- Python and Deep Learning

## Fall 2018

## Python Lab Assignment 3

## Submitted On 11/9/2018

## Name: Alexandria Piatt and Alexander Larios

## Class IDs (respectively): 23 15

## TABLE OF CONTENTS

## 1. Author

## 2. Objective

## 3. Features

## 4. Configuration

5. Input/Output Screenshots

6. Implementation & Full Source Code

7. References

**AUTHORS**

This lab report corresponds to the third lab assignment in CS 5590 0001 Python and Deep-Learning. ALEXANDRIA PIATT (ID: 23) and ALEXANDER LARIOS (ID: 15) are both in their last year at UMKC, both are completing a B.S. in C.S. in December 2018. The leader of the course is SARIA GOUDARZVAND.

**OBJECTIVE**

The objective of this lab was to introduce the programmer to the Keras library and how to use TensorBoard and TensorFlow technologies. This is accomplished by completing two open ended tasks. The first is performing linear regression on a dataset of the programmers choice. The second task is to perform logistic regression on a dataset of the programmers choice.

**FEATURES**

Task 1:

***Perform linear regression on a dataset***

The programmer used the auto\_mpg dataset. There are about 400 datapoints in the set with the goal of estimating the mpg of various vehicles based on attributes. The attributes used in the lab are : cylinders, horsepower, weight acceleration, model year. The linear regression model was trained using the attributes to estimate the miles per gallon of gas. The training set was 80% of the data, the testing set was the remaining 20%.

Task 2:

***Perform logistic regression on a dataset***

The programmer used the Titanic dataset. There are about 1300 datapoints in the set with the goal of classifying passengers as either surviving or dying the disaster. The attributes used in the lab are : passenger class, sex, age, whether the passenger was with a family, parch, fare paid. The logistic regression model was trained using the attributes to classify each passenger as either dying or surviving.

**CONFIGURATION**

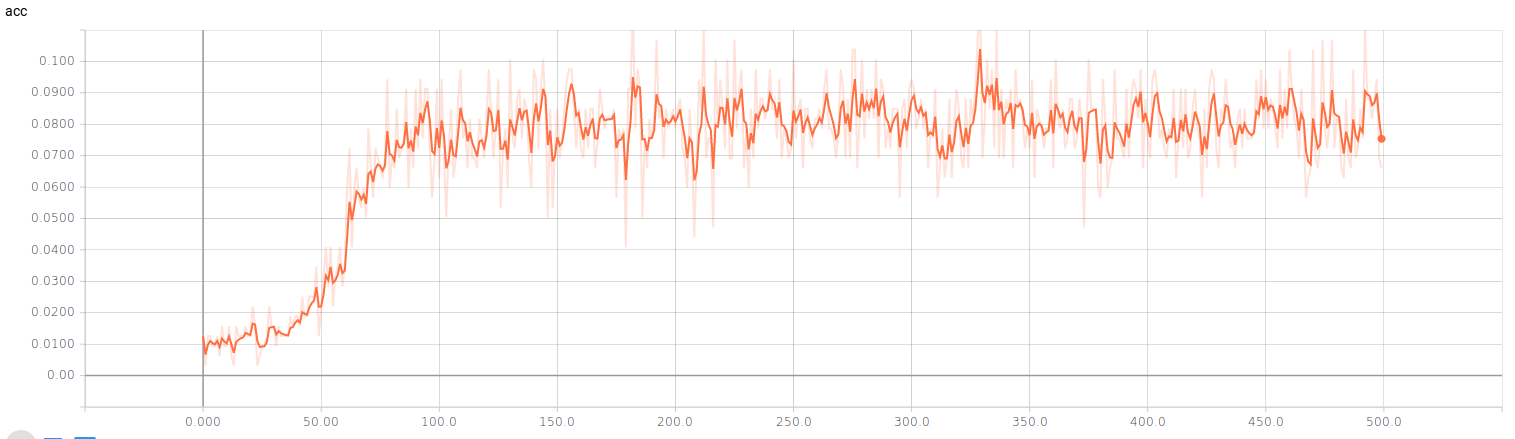
All of the code in this lab was written and built using PYCHARM IDE, in an Anaconda3 environment and using Python 3.6.

**INPUT/OUTPUT SCREENSHOTS**

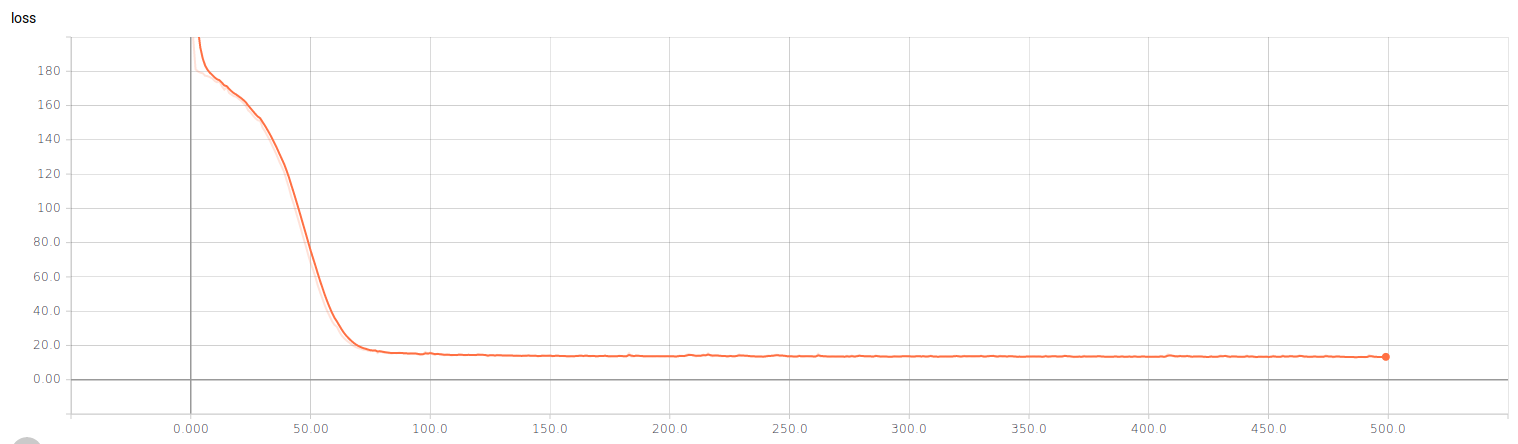
Task 1:

There is no required user input but the script reads in auto-mpg.csv and then trains the model and estimates the mpg of the remaining rows. Originally the programmer tried to use a 70% training and a 30% testing sets, but an 80%/20% provided for better accuracy. The best accuracy the programmer was able to obtain was a 9% accurate. The settings to obtain the 9% was using epochs = 6000, batch\_size = 10, loss = mean\_squared\_error, optimizer = “adam”. Other settings used were epochs = 3000, 500, 1000, optimizer = “sgd” and “adagrad”, batch\_size = 100, 35, activation = “relu”, “linear” and “sigmoid”. None of the above mentioned settings had better accuracy, with some combinations being as low as 0%. The programmer believes the problem with this model’s accuracy was the dataset it began with. There just was not enough data with attributes that were strongly correlated with the mpg. With more datapoints and more attributes, a better accuracy can be achieved.

TensorBoard accuracy output:



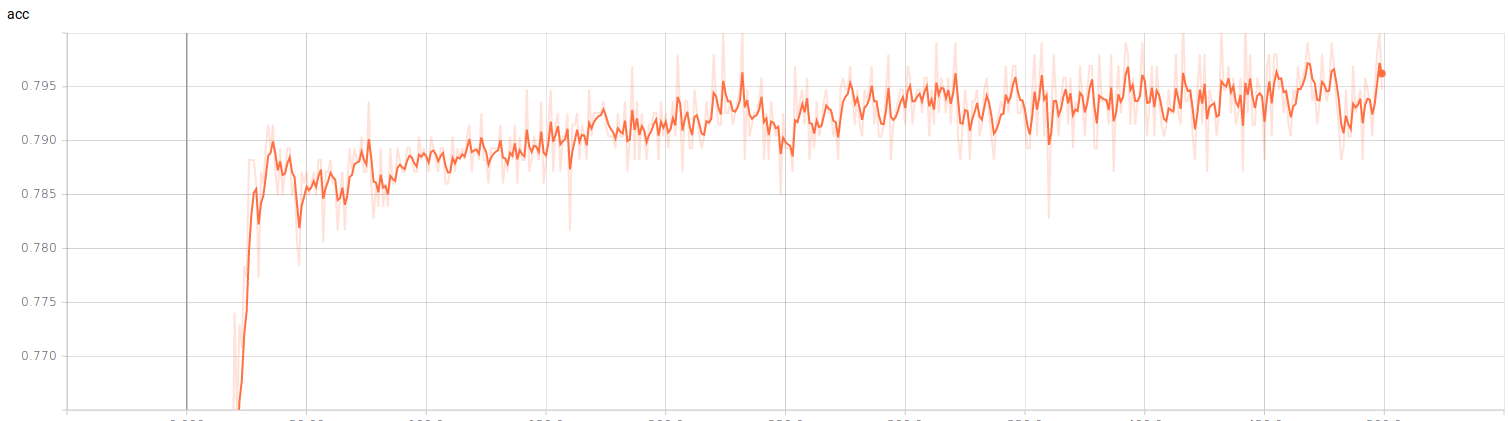
TensorBoard loss output:



Task 2:

Again, this task does not require any user input. The script reads in the Titanic dataset. The programmer used a 70%/30% training and testing set. The best accuracy the programmer found was a 80% accuracy rate. The options for this set was epochs=500, activation=”sigmoid”, optimizer=”adam”, loss=”binary\_crossentropy. Higher and lower epoch numbers had worse accuracy. Other activation options tested were “relu” and “tanh”, both resulting in worse accuracy, similar results with optimizer = “sgd”. Everything resulted in worse accuracy. The accuracy for this model was more likely caused by random chance in the actual disaster than any issues with the dataset. The dataset does not include where the passengers were at the time of the crash. That could have played a factor into who lived and who died.

TensorBoard accuracy output:

TensorBoard loss output:



**IMPLEMENTATION & FULL SOURCE CODE**

Task 1:

The programmer used the Keras library to implement a linear regression model. A small amount of dataclean up was required to be able to successfully train and test the model. The horsepower attribute had missing data so to fill in the missing parts, the average of the rest of the horsepower points was used. After the data was cleaned up the rest of the work was done using the Keras library.

**from** \_\_future\_\_ **import** print\_function

**from** time **import** time

**import** numpy **as** np

**import** pandas **as** pd

**from** keras.callbacks **import** TensorBoard

**from** keras.models **import** Sequential

**from** keras.layers **import** Dense

**from** sklearn.model\_selection **import** train\_test\_split

*# read in data*

auto\_data = pd.read\_csv(**'auto-mpg.csv'**)

*# clean up and fill in missing data points*

auto\_data.horsepower = auto\_data.horsepower.fillna(value=auto\_data.horsepower.mean())

*# create the tensorboard framework*

tensor\_board = TensorBoard(log\_dir=**"linear\_logs/{}"**.format(time()))

*# split up the data*

x\_train, x\_valid, y\_train, y\_valid = train\_test\_split(auto\_data.iloc[:,0:5], auto\_data.iloc[:,5],test\_size=0.2, random\_state=87)

np.random.seed(816)

*# create the model*

model = Sequential()

model.add(Dense(1, input\_dim=5, init=**'normal'**, activation=**"linear"**))

model.add(Dense(1, init=**'normal'**))

model.compile(loss=**'mean\_squared\_error'**, optimizer=**'adam'**, metrics=[**"accuracy"**])

*# train and test the model*

model.summary()

epochs = 6000

batch\_size =10

history = model.fit(x\_train, y\_train,

batch\_size=batch\_size,

epochs=epochs,

shuffle=**True**,

verbose=2,

validation\_data=(x\_valid, y\_valid), callbacks=[tensor\_board])

score = model.evaluate(x\_valid, y\_valid)

print(**"test accuracy"**, score[1])

Task 2:

The implementation for the logistic regression model was also done using the Keras library. Again there was data that needed cleaned up. The specific attributes missing data points were age and fare. The sex attribute was a text column, so using a label\_encoder the programmer transformed the column into a binary representation of the two sexes. After the data was cleaned up, the programmer used the Keras library to create the model.

**from** time **import** time

**import** pandas

**from** sklearn.preprocessing **import** LabelEncoder

**from** sklearn.model\_selection **import** train\_test\_split

**from** keras.models **import** Sequential

**from** keras.layers **import** Dense

**from** keras.callbacks **import** TensorBoard

titanic\_data = pandas.read\_excel(**'titanic3.xls'**, **'titanic3'**)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(titanic\_data.iloc[:,0:6], titanic\_data.iloc[:,6], test\_size=0.3, random\_state=87)

sex\_label = LabelEncoder()

*# clean up the training data*

filler\_age = x\_train.age.median()

x\_train.age = x\_train.age.fillna(value=filler\_age)

x\_train.sex = sex\_label.fit\_transform(x\_train.sex)

x\_train.fare = x\_train.fare.fillna(value=x\_train.fare.median())

*# clean up testing data*

filler\_age\_test = x\_test.age.median()

x\_test.age = x\_test.age.fillna(value=filler\_age\_test)

filler\_fare\_test = x\_test.fare.median()

x\_test.sex = sex\_label.fit\_transform(x\_test.sex)

*# create the tensorboard*

tb = TensorBoard(log\_dir=**"logs/{}"**.format(time()))

*# create the model*

kModel = Sequential()

kModel.add(Dense(1, input\_dim=6, activation=**'sigmoid'**))

kModel.compile(optimizer=**'adam'**, loss=**'binary\_crossentropy'**, metrics=[**"accuracy"**])

print(kModel.summary())

*# Train the model*

fit\_results = kModel.fit(x\_train, y\_train, batch\_size=10, nb\_epoch=500, validation\_data=(x\_test, y\_test), callbacks=[tb])

score=kModel.evaluate(x\_test, y\_test)

print(**"test accuracy"**, score[1])

**REFERENCES**

kaggle.com

keras.io