Newton vs Machine

Three-body problem

Large Scale Computing last Assignment – Joon Jung

I did want to try taking out entire one trajectory so we can actually set that aside as test set, but I wasn’t able to do that. I think I have a general sense of how to plot trajectory, actually trying it as of right now, but probably won’t be able to get visualization before class, so I am uploading my results, which should be good enough for the assignment.

#Importing all the required libraries

import tensorflow as tf

import keras

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import InputLayer,Conv2D, Conv3D,MaxPooling2D, Dense, Flatten, Dropout, BatchNormalization

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.callbacks import TensorBoard

import matplotlib.pyplot as plt

from tensorflow.keras.callbacks import EarlyStopping,ReduceLROnPlateau

import numpy as np

import pandas as pd

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

train\_X\_initial = pd.read\_csv('~/joon/train\_X.csv',header=None)

train\_Y\_initial = pd.read\_csv('~/joon/train\_Y.csv',header=None)

train\_X, test\_X = train\_test\_split(train\_X\_initial, test\_size=0.01, random\_state=14, shuffle=True)

train\_Y, test\_Y = train\_test\_split(train\_Y\_initial, test\_size=0.01, random\_state=14, shuffle=True)

#Setting a seed for analysis.

tf.keras.utils.set\_random\_seed(14)

tf.config.experimental.enable\_op\_determinism()

model = tf.keras.Sequential([

    InputLayer(input\_shape = (3,)),

    Dense(128, activation='relu'),

    Dense(128, activation='relu'),

    Dense(128, activation='relu'),

    Dense(128, activation='relu'),

    Dense(128, activation='relu'),

    Dense(128, activation='relu'),

    Dense(128, activation='relu'),

    Dense(128, activation='relu'),

    Dense(128, activation='relu'),

    Dense(128, activation='relu'),

    Dense(4, activation='linear')

])

model.compile(optimizer=tf.keras.optimizers.Adam(), loss=tf.keras.losses.MeanAbsoluteError(), metrics=['MeanAbsoluteError', 'accuracy'])

train\_model = model.fit(train\_X, train\_Y, epochs=1000, batch\_size = 5000, verbose=1, validation\_data=(test\_X, test\_Y))

# Plotting the train and test loss

plt.plot(train\_model.history['val\_mean\_absolute\_error'], label='Val\_MAE', color='red', linestyle='dashed')

plt.plot(train\_model.history['mean\_absolute\_error'], label='Train\_MAE', color='blue')

plt.yscale('log')

plt.legend()

plt.title('MAE')

plt.savefig('TrainTest.png')

plt.clf()

# Plotting the train and test accuracy

plt.plot(train\_model.history['val\_accuracy'], label='Val\_accuracy', color='red', linestyle='dashed')

plt.plot(train\_model.history['accuracy'], label='Train\_accuracy', color='blue')

plt.yscale('log')

plt.legend()

plt.title('Accuracy')

plt.savefig('TrainTestaccuracy.png')

plt.clf()

# Calculating the final test accuracy

predicted\_Y = model.predict(test\_X)

# predicted\_classes = np.argmax(predicted\_classes, axis=1)

from sklearn.metrics import r2\_score

score = r2\_score(test\_Y, predicted\_Y)

print("The final test accuracy of the model is {}%".format(round(score, 2) \* 100))

# Fitting the model

history\_model\_class = model\_class.fit(train\_images, train\_labels, batch\_size = 32, verbose = 1, epochs = 20, validation\_data=(test\_images, test\_labels))

# Plotting the accuracies

dict\_hist = train\_model.history

list\_ep = [i for i in range(1, 21)]

plt.figure(figsize = (8, 8))

plt.plot(list\_ep, dict\_hist['accuracy'], ls = '--', label = 'accuracy')

plt.plot(list\_ep, dict\_hist['val\_accuracy'], ls = '--', label = 'val\_accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epochs')

plt.legend()

plt.show()

# used to import it to my local machine

scp [jjung5@bridges2.psc.edu](mailto:jjung5@bridges2.psc.edu):/jet/home/jjung5/joon/TrainTest.png /Users/joonjung/Desktop/CMU/LargeScale

scp [jjung5@bridges2.psc.edu](mailto:jjung5@bridges2.psc.edu):/jet/home/jjung5/joon/TrainTestaccuracy.png /Users/joonjung/Desktop/CMU/LargeScale

A graph of a graph

Description automatically generated

Y scale is log scale.

A graph of a curve

Description automatically generated

A screen shot of a computer program

Description automatically generated

I used the sklearn.metrics r2\_score to calculate the last actual test accuracy on test data set.