AI Final assignment

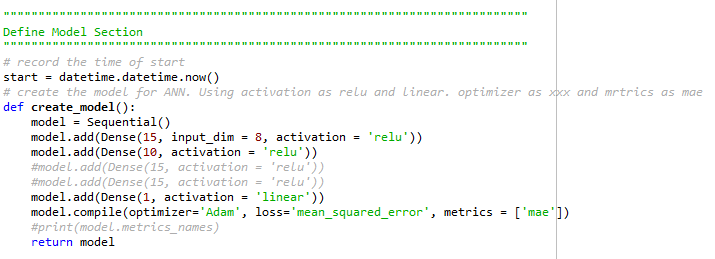
Yuke Luo

2020.4.14

**Objective**

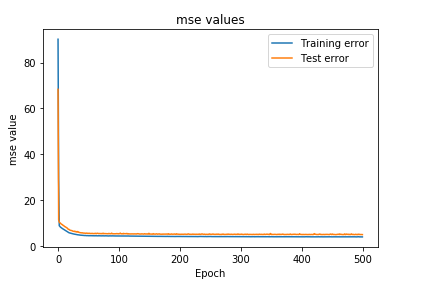
I am going to use Artificial Neural Network (ANN) to find a model that is the best one to predict the age of abalone from physical measurements according to the dataset including sex, length, diameter, height, etc. The last field of the csv file, which is “Rings (+1.5 gives the age in years)” is the field to make the prediction from the previous fields. To find the best model, we need to find the model with lowest mean squared error and lowest mae. Therefore, the objective for this assignment is to find the most fit value for the best model. Dataset is from <https://archive.ics.uci.edu/ml/datasets/Abalone>. Data Set Characteristics: Multivariate. Attribute Characteristics: Categorical, Integer, Real. The model for this assignment will be ANN regressor model.

**Final ANN model, in code**



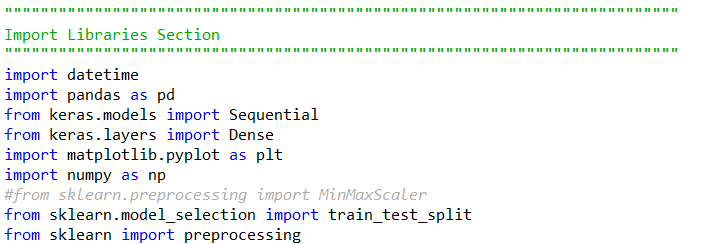
**Final model and training algorithm, in words**

I have changed several variables during the modeling setting, such the nodes for the hidden layers (the first number in Dense function); number of layers; number of epochs and batch size. I didn’t use k = 5 in this model because in my case, it’s unnecessary. For the final model, I decide to use 1 input layer and 1 hidden layer and 1 output layer because in this way, the model will not waste too much time, meanwhile, the error rate is not influenced severely. In the input layer, I chose 15 as its node, and 10 for the hidden layer, 1 for output layer since this is an ANN regressor model. After trying all possible activations and optimizers, I decided to use ‘relu’ for input and hidden layer’s activation and ‘linear’ for output layer. I used ‘adam’ for the optimizer and ‘mae’ for matrics since we mainly want to look at mse and mae for ANN regressor.

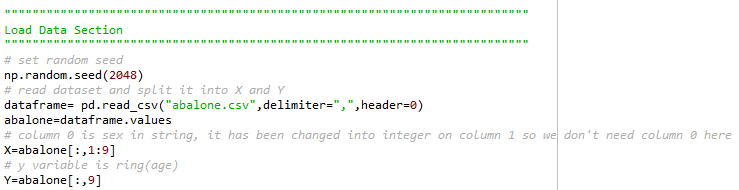


**An explanation of the input variables and any preprocessing steps you took.**

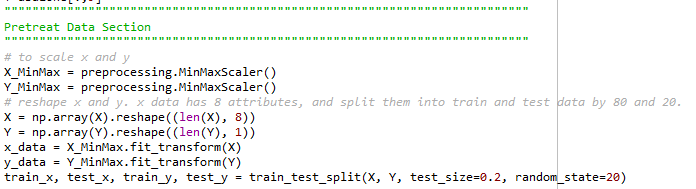
The first thing to do is the most important thing – importing packages. Without this step, most functions will not work in the code.



Next is to read the table in order to access the data. Because the dataset is in the csv file, we must use delimiter=”,” to split it into different fields. Use header=0 to exclude the title row, which tells us the information of all fields. Because I checked the csv file and didn’t see any missing value and null value in the dataset, I didn’t use any code to omit NAs. In addition, I manually rearranged the dataset in excel by changing sex in strings to integers (originally “F”, “M”, “I”, now 1,2,3) for convenience data processing.



Then, it’s time to set our training data and test data. I put 80% of the data as training data, and the rest 20% as test data because I think this is an appropriate ratio to split training and test data. I tried to scale and reshape x and y because the original data is so noisy that makes both training and testing mse and mae extremely high. I wanted to see if scale and reshape could help solve the problem. As a result, they did not.



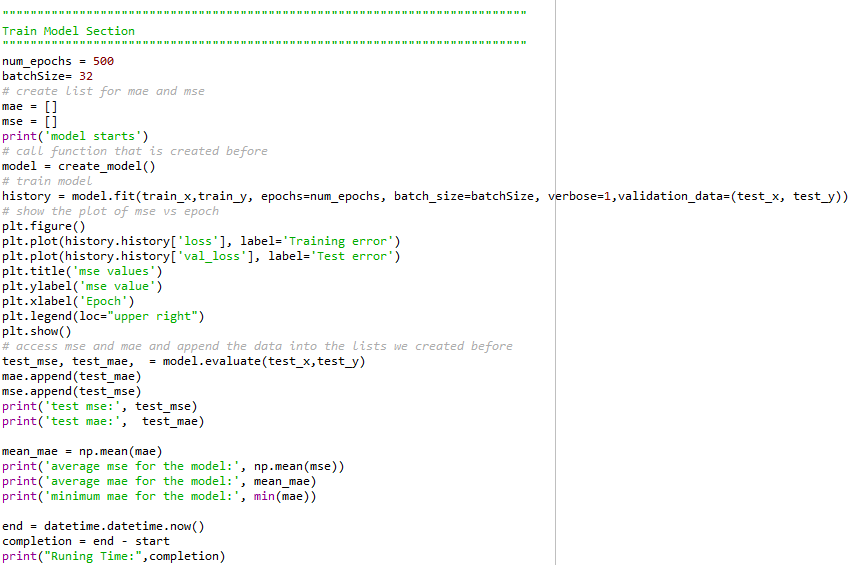
**An explanation of your metrics and justification for your choice. An explanation of your method to validate the model.**

Because the dataset size is not very large (approximately 4200 rows), the batch size doesn’t need to be too large. I will decrease the batch size and increase epoch number to see if the error rate will change. As known, larger batch size will speed up the training process, but not converge as fast. I first tried to change the number of the batch size, starting from 24, and kept other factors in constant. I chose 24 because the dataset is not huge but not small, either, if the batch size is too small, it will take a long time to run the model. On the other hand, if the batch size is too large, it will be meaningless to split such a small dataset into too many pieces. I also calculated the average number of mae and mse to see the error rate. I will increase the batch size by 8 each time, because increasing by 8 is not a too large or too small interval for batch size. As soon as the error rate stop to decrease, or the running time start to increase, I will choose that batch size and start to adjust other factors. For number of epochs, I decide to start with 300, and increase by 50 each time. For hidden layers and nodes, I decide to start with simple ones which contains only three layers and (15,15,1) for the nodes. I chose 15s for both input layer and hidden layer because it’s a small number and is still qualified to be as the number of nodes. After adjusting several times, I decided to keep num\_epochs as 500 and batchsize as 32 to maintain the short running. In this way, the only thing I need to change is the number of layers and number of the first hidden layers. For the changing of layer, I tried 3 layers at first, I choose 10, 15, 50, 100 for testing the quality of the layers only because they are some typical numbers used for the nodes. As seen, I calculated average value for mse and mae for convenience. I didn’t take the number of averages of mse as a reference number because I think it’s uncessary to do that, but it will be printed in the code just for additional information. The highlight parts are the choice I made during experiments for each changes of variables. Also, for most of the time the changes of results are not huge, in this case, I will simply pick the ones with shorter running time. According to my testing and records:

Average mse: 5.25

Average mae: 1.60

Running time: 02:08



For the metrics, I use “print(model.metrics\_names)” code when checking the results of names of different matrics. This is a very useful code during my experiments because I used two different computer for the assignment, and each time when I changes to another computer, I found out the variables names are different, on one computer, I need to use ‘mae’, on another, I need to use ‘mean\_absolute\_error’. When I use only mse for metrics, the output I am given will only have loss and mse instead of having the mae, which is asked in the assignment. Therefore, I use ‘mae’ as matrix to check both level of accuracy and mean squared error for the model. The code in this section is one of the justifications I did during examinations.

For the activations of input layer and hidden layers, I used “relu” as usual. For activation of output layer, I tried 'softmax', 'softplus', 'softsign', 'relu', 'tanh', 'sigmoid', 'hard\_sigmoid', and 'linear'. The result among all these activations, linear has the best mae and loss, so I picked it as output activation. For the optimizer, I keep it as “adam” and didn’t change it because I think this might be the best optimizer for this model. In the cross-validation, I choose to plot training mse and test mse. There are three values given by model.evaluate(), which are loss, accuracy and mean squared error, I assigned each value a name and print mae and mse to see their values. After the validation, I calculated the running time by subtracting end time to start time.

**Conclusion and discussion:**

In this dataset with 4178 records with 9 attributes, because of the noisy data, my results are given in some inappropriate numbers and my judgement for optimal number of units may be influenced as well. However, I’ve tried my best to justify the model to make my model as optimal as possible. Therefore, for the optimal number of units, I decided to keep number of epochs as 500 (the number of repetitions during training) and batch-size as 32 to maintain the short running. For layers, I picked 3 layers with 15, 10, 1 node in each layer. For the activations of input layer and hidden layers, I used “relu” as usual. For activation of output layer, ‘linear’ will be the best choice. When choosing the optimizer, I keep it as “adam” and didn’t change it because I think this might be the best optimizer for this model. In the cross-validation, I choose to plot training mse and test mse because I have ‘mae’ as my metrics. In this model, I’m using supervised training method to train the model since it allows you to collect data or produce a data output from the previous experience in the regression model. In data splitting section, I put 80% of the data as training data, and the rest 20% as test data because I think this is an appropriate ratio to split training and test data. I didn’t contain learning rate or momentum in my ANN model. Even though learning rate and momentum may be helpful to speed up the training, the model for now is fast enough. For further work, the noise in dataset must be adjusted, either in excel or within the model during data preprocessing. Also, if time allows, it will be a good idea to set k = 10 to make a better training for the model.

**Code in words:**

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

Import Libraries Section

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

import datetime

import pandas as pd

from keras.models import Sequential

from keras.layers import Dense

import matplotlib.pyplot as plt

import numpy as np

#from sklearn.preprocessing import MinMaxScaler

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

from sklearn import preprocessing

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

Load Data Section

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

# set random seed

np.random.seed(2048)

# read dataset and split it into X and Y

dataframe= pd.read\_csv("abalone.csv",delimiter=",",header=0)

abalone=dataframe.values

# column 0 is sex in string, it has been changed into integer on column 1 so we don't need column 0 here

X=abalone[:,1:9]

# y variable is ring(age)

Y=abalone[:,9]

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

Pretreat Data Section

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

# to scale x and y

X\_MinMax = preprocessing.MinMaxScaler()

Y\_MinMax = preprocessing.MinMaxScaler()

# reshape x and y. x data has 8 attributes, and split them into train and test data by 80 and 20.

X = np.array(X).reshape((len(X), 8))

Y = np.array(Y).reshape((len(Y), 1))

x\_data = X\_MinMax.fit\_transform(X)

y\_data = Y\_MinMax.fit\_transform(Y)

train\_x, test\_x, train\_y, test\_y = train\_test\_split(X, Y, test\_size=0.2, random\_state=20)

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

Define Model Section

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

# record the time of start

start = datetime.datetime.now()

# create the model for ANN. Using activation as relu and linear. optimizer as xxx and mrtrics as mae

def create\_model():

model = Sequential()

model.add(Dense(15, input\_dim = 8, activation = 'relu'))

model.add(Dense(10, activation = 'relu'))

#model.add(Dense(15, activation = 'relu'))

#model.add(Dense(15, activation = 'relu'))

model.add(Dense(1, activation = 'linear'))

model.compile(optimizer='Adam', loss='mean\_squared\_error', metrics = ['mae'])

#print(model.metrics\_names)

return model

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

Train Model Section

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

num\_epochs = 500

batchSize= 32

# create list for mae and mse

mae = []

mse = []

print('model starts')

# call function that is created before

model = create\_model()

# train model with training data, and then test it with test data

history = model.fit(train\_x,train\_y, epochs=num\_epochs, batch\_size=batchSize, verbose=1,validation\_data=(test\_x, test\_y))

# show the plot of mse vs epoch

plt.figure()

# plot all data with mse values on y axis and epoch on x axis to see mse changing with epoch increases

plt.plot(history.history['loss'], label='Training error')

plt.plot(history.history['val\_loss'], label='Test error')

plt.title('mse values')

plt.ylabel('mse value')

plt.xlabel('Epoch')

plt.legend(loc="upper right")

plt.show()

# access mse and mae and append the data into the lists we created before

test\_mse, test\_mae, = model.evaluate(test\_x,test\_y)

mae.append(test\_mae)

mse.append(test\_mse)

print('test mse:', test\_mse)

print('test mae:', test\_mae)

# np.mean is used to calculate the mean value of the list

mean\_mae = np.mean(mae)

print('average mse for the model:', np.mean(mse))

print('average mae for the model:', mean\_mae)

print('minimum mae for the model:', min(mae))

# record the end time

end = datetime.datetime.now()

# end time minus start time will be the time duration of the whole training process

completion = end - start

print("Runing Time:",completion)