***NEURAL TRAINING MODEL FOR LEFT WHEEL***

import pandas as pd

import numpy as np

file\_ = pd.read\_csv("C:/Users/SJ/Desktop/Book4.csv")

file\_=file\_.drop('TIME',axis=1)

file\_.head()

x = file\_["US"]

y1=file\_["IR1"]

y2=file\_["IR2"]

y1.head(),y2.head(), x.head()

x = file\_["US"]

y1=file\_["IR1"]

y2=file\_["IR2"]

y1.head(),y2.head(), x.head()

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

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y1,test\_size=0.2,random\_state=10)

x\_train = np.array(x\_train)

x\_train=np.reshape(x\_train,(-1,1))

x\_test = np.array(x\_test)

x\_test = np.reshape(x\_test,(-1,1))

y\_test = np.array(y\_test)

y\_train = np.array(y\_train)

x\_train=x\_train/1000

x\_test=x\_test/1000

y\_train=y\_train/1000

y\_test=y\_test/1000

x\_train,x\_test

addSqlcol = lambda x: np.concatenate((x ,np.ones((x.shape[0], 1))),axis=1)

A = addSqlcol(x\_train)

Y = y\_train

A.shape

def sigmoid (x):

y = 1/(1 + np.exp(-x))

return y

def sigmoid\_derivative(x):

return x \* (1 - x)

ni = 2

nh = 4

no = 1

weights1 = (2\*np.random.random((nh,ni)) - 1)\*0.01/2

weights2 = (2\*np.random.random((no,nh)) - 1)\*0.01/2

eta = 0.1

for epochs in range(5000):

for s in range(A.shape[0]):

## forward propagation

hidden = sigmoid(np.dot(weights1, A[s])).reshape(-1, 1)

y = sigmoid(np.dot(weights2, hidden)).reshape(-1,1)

## backward propagation

delta\_weights2 = np.dot(((Y[s] - y) \* sigmoid\_derivative(y)),hidden.reshape(-1, 1).T)

delta\_weights1 = np.dot(((Y[s]-y) \* sigmoid\_derivative(y)\*weights2.T \* sigmoid\_derivative(hidden)), A[s].reshape(1,2))

## update weights

weights1 = weights1 + eta\*delta\_weights1

weights2 = weights2 + eta\*delta\_weights2

error = 0

for s in range(A.shape[0]):

hidden = sigmoid(np.dot(weights1, A[s])).reshape(-1, 1)

y = sigmoid(np.dot(weights2, hidden)).reshape((-1,1))

#print('x:',A[s,:-1],' y:',y)

error += (Y[s] - y).dot(Y[s] - y)

#print(Y[s], y)

print('error:',error/9454)

print(weights1)

print(weights2)

#Accuracy Of Left Wheel Velocity

acc=100-((error/9454)\*1000)

float(acc)

***#FOR CHECKING OUTPUT***

S = int(input("Distance-"))

S = [S/1000,1]

S = np.array(S)

hiddeni = sigmoid(np.dot(weights1, S)).reshape(-1, 1)

VL = sigmoid(np.dot(weights2, hiddeni)).reshape(-1,1)

VL=VL\*1000

VL=int(VL)

print("VL-",VL)