Project: **Machine Learning Programming Project 4 Part B**
Team Members: **Debit Paudel, Kushal Dahal**
Github Link: [https://github.com/debit7/Backpropagation_neural_network_learning (https://github.com/debit7/Backpropagation_neural_network_learning)](https://github.com/debit7/Backpropagation_neural_network_learning)

To practice the Backpropagation(Neural Network Training) Algorithm, we conducted feedforward, backpropagate and learn for each training sets. We have implemented a dynamic programming skill to conduct this project. We have built three major functions to practice feed_inputs_forward(),back_propagation(), Learn() for the project. Pandas and math are the libraries we have used. Initially we have built a dataframe and provided the learning rate as 0.5 and also the initial weights and values. Two dataframes are created to store and update the values of inputs and weights for initial inputs,hidden layer and output. Hidden layers,Errors and ouptut are provided in the form of list for dynamic programming.

In [1]:

```
import pandas as pd
import math as math
weights_df=pd.DataFrame([['b0',1,1,0],['l1',1,-1,0],['l2',0.5,2,0],['b1',0,0,1],[

values_df=pd.DataFrame([['b0',1],['l1',0],['l2',1],['h1',0],['h2',0],['b1',1],['y
initial_inputs=['b0','l1','l2']
Hidden_Layer=['b1','h1','h2']
Errors=['Ey','Eh1','Eh2']
Output=['y']
Learning_rate=0.5
```

find_values() helps to extract the value of the corresponding inputs.

In [2]:

```
def find_values(df,x1):
    df.set_index("x", inplace = True)
    val=df.loc[x1]['values']
    df.reset_index(inplace=True)
    return val
```

update_values() function helps to update the updated value for inputs.

In [3]:

```
def update_values(df,x1,new_val):
    df.loc[df.x == x1,"values"] = new_val
```

find_distance() function extract the distance between the two nodes from the weights_df dataframe.

In [4]:

```
def find_distance(df,x1,x2):
    df.set_index("x", inplace = True)
    val=df.loc[x1][x2]
    df.reset_index(inplace=True)
    return val
```

update_weights() function updates the updated distance values in the weights_df dataframe.

In [5]:

```
def update_weights(df,x1,x2,new_val):
    df.loc[df.x == x1,x2] = new_val
```

step() function performs the calculation for the output of each unit.

In [6]:

```
def step(x):
    return round(1/(1+(math.exp(x*-1))),3)
```

Calculate_error() function calculates the error of the final output after input feed forwarding.

In [7]:

```
def Calculate_error(target,value):
    return round((1/2)*(target-value)**2,3)
```

feed_inputs_forward() function propagates the input forward through the network. It inputs the instances and calculate output for each units. This function receives the inputs, weights dataframe, values dataframe, layer and the output. It calculates the input values for hidden layers, output and updates the values in the particular dataframe respectively.

In [8]:

```
def feed_inputs_forward(inputs,weights_df,values_df,layer,output):
    s=find_values(values_df,inputs[1])*find_distance(weights_df,inputs[1],layer[1
    find_values(values_df,inputs[2])*find_distance(weights_df,inputs[2],layer[1])
    find_values(values_df,inputs[0])
    update_values(values_df,layer[1],step(s))
    s=find_values(values_df,inputs[2])*find_distance(weights_df,inputs[2],layer[2
    find_values(values_df,inputs[1])*find_distance(weights_df,inputs[2],layer[1])
    find_values(values_df,inputs[0])
    update_values(values_df,layer[2],step(s))
    s=find_values(values_df,Hidden_Layer[1])*find_distance(weights_df,Hidden_Laye
    find_values(values_df,Hidden_Layer[2])*find_distance(weights_df,Hidden_Layer[
    find_values(values_df,Hidden_Layer[0])
    # print(s)
    update_values(values_df,output[0],step(s))
    print('Error y:',Calculate_error(1,find_values(values_df,output[0])))
    pass
```

back_propagation() computes and propagates the error backwards. For each output unit and hidden units, it calculates the error and updates in the dataframes respectively.

In [9]:

```
def back_propagation(values_df,output,target,layer,Errors):
    Ey=find_values(values_df,output[0])*(1-find_values(values_df,output[0]))*(tar
    Ey=round(Ey,3)
    update_values(values_df,Errors[0],Ey)
    c=0
    for ly in layer[1:]:
        c+=1
        z=find_values(values_df,ly)*\
        (1-find_values(values_df,ly))*\
        (find_distance(weights_df,ly,output[0])*\
        Ey)
        z=round(z,3)
        update_values(values_df,Errors[c],z)

    pass
```

Learn() function updates the each network weights proportionally with the help of error measure for each unit and the outputs of the units. The weights_df dataframe is updated in this function.

In [10]:

```
def Learn(Learning_rate,layer,output,values_df,Errors,weights_df,inputs):
    #update weights from hidden layer to output
    for lys in layer:
            z=find_distance(weights_df,lys,output[0])+\
                Learning_rate*\
                find_values(values_df,Errors[0])*\
                find_values(values_df,lys)
            update_weights(weights_df,lys,output[0],round(z,3))
    #update hidden layer from initial input to hidden layer
    i=0
    for lyr in layer[1:]:
        i+=1
        for inpts in inputs:
                z=find_distance(weights_df,inpts,lyr)+\
                    Learning_rate*\
                    find_values(values_df,Errors[i])*\
                    find_values(values_df,inpts)
                update_weights(weights_df,inpts,lyr,round(z,3))

    pass
```

Below we have called the function to complete the process for a single instance.We followed the steps of feed forwarding the inputs, backpropagating the errors and updating the weights(Learn) using feed_inputs_forward(), back_propagation() and Learn() functions.

In [11]:

```
feed_inputs_forward(initial_inputs,weights_df,values_df,Hidden_Layer,Output)
back_propagation(values_df,Output,1,Hidden_Layer,Errors)
Learn(Learning_rate,Hidden_Layer,Output,values_df,Errors,weights_df,initial_input
```

```
Error y: 0.024
```

```
In [12]: print(values_df)
```

```
     x values
0   b0      1
1   l1      0
2   l2      1
3   h1  0.818
4   h2  0.953
5   b1      1
6    y  0.781
7   Ey  0.037
8  Eh1  0.008
9  Eh2 -0.002
```

```
In [13]: print(weights_df)
```

```
    x     h1     h2      y
0  b0  1.004  0.999  0.000
1  l1  1.000 -1.000  0.000
2  l2  0.504  1.999  0.000
3  b1  0.000  0.000  1.018
4  h1  0.000  0.000  1.515
5  h2  0.000  0.000 -0.982
```

We have also printed the values and the final weights and found out that all the values and weights exactly matches the tutorial steps, values and weights.

```
In [ ]:
```