Project: **Machine Learning Programming Project 4 Part A**
Team Members: **Debit Paudel, Kushal Dahal**
Github Link: **https://github.com/debit7/Truth_Table_Learning**

While working on the Project 4 Part A, we initially created a dataset named 'instances.text'. This dataset contains the information target values of boolean gates and the inputs. Initially we read this dataset using open() of python and appended on 'lst' list. The learning rate is initially given the value '0.20' and weights are provided accordingly in a list.

In [9]:
```
weights = [0.01,0.08, 0.08]
learning_rate=0.20
with open('instances.text') as f:
        lst = []
        for ele in f:
            line = ele.replace('\n','').split(',')
            lst.append(line)
print(lst)
```

```
[['0', '0', '0', 'AND'], ['0', '1', '0', 'AND'], ['1', '0', '0', 'AND'], ['1', '1', '1',
'AND'], ['0', '0', '0', 'OR'], ['0', '1', '1', 'OR'], ['1', '0', '1', 'OR'], ['1', '1',
'1', 'OR'], ['0', '0', '1', 'NAND'], ['0', '1', '1', 'NAND'], ['1', '0', '1', 'NAND'],
['1', '1', '0', 'NAND'], ['0', '0', '1', 'NOR'], ['0', '1', '0', 'NOR'], ['1', '0', '0',
'NOR'], ['1', '1', '0', 'NOR']]
```

Below we have created step function to return 1 if the passed parameter 'val' is greater than 0.If the passed parameter 'val' is less than 0, then it returns 0. The predict function activates the perceptron and returns the activated value calling the step() function.

In [2]:
```
def step_function(val):
    return 1 if val > 0 else 0
def predict(row,weights):
    activation=weights[0]
    for i in range(len(row)-2):
        activation+=weights[i+1]* int(row[i])
    return step_function(activation)
```

Here, we will separate the dataset according to input gate provided by the interpreter.

In [3]:
```
def dataset(whole_dataset,Gate):
    dataset=[]
    for sets in whole_dataset:
        if sets[3]==Gate:
            dataset.append(sets)
    return dataset
```

For the training, we have build a training function where we input the data related to the particular gate and dataframe that provides inputs and targets. In the row[-2], we have the target value. When the target value is unequal to the predicted value, we update the weights according to the delta rule. For the next input values, we consider the new updated weights to predict until all predictions are not matched with target values.

In [4]:
```
def Train(Gate,Data):
```

```
    dataset=[]
    for sets in Data:
        if sets[3]==Gate:
            dataset.append(sets)
    correct=0
    print("Initial weights",weights)

    epoch=0
    while True:
        print('**************************************************************************
        epoch+=1
        print("epoch:",epoch)
        correct=0
        for row in dataset:
            print('\n')
            print('inputs:',row[0],row[1])
            print('weights:',weights)
            prediction = predict(row, weights)
            if int(row[-2])==prediction:
                strrr="Correct"
            else:
                strrr="Incorrect"
            print("y=%d, t=%d =>%s" % ( prediction,int(row[-2]),strrr))
            if int(row[-2])!=prediction:
                weights[0]=round(learning_rate*(int(row[-2])-prediction)+weights[0],3)
                weights[1]=round(learning_rate*(int(row[-2])-prediction)*int(row[0])+we
                weights[2]=round(learning_rate*(int(row[-2])-prediction)*int(row[1])+we
                # print('new weights',weights)
            else:
                correct+=1
        print('\n')
        print("total corrects:",correct)
        if correct==4:

            break
    print('final weights',weights)
```

After writing a dynamic code, we called a Train() function below to explore the updated weights and epoch for 'AND','OR','NAND' and 'NOR' gates. We set the initial weights as [0.01,0.08, 0.08] and run the code for all the gates. We found out that there were 6 epoches for 'AND' gate, 4 epoches for 'OR' gate, 6 epoches for 'NOR' gate and 4 epoches for 'NAND' gate to match all the predicted values with the targets.

In [5]:
```
Train('AND',lst)
```

```
Initial weights [0.01, 0.08, 0.08]
**************************************************************************
epoch: 1


inputs: 0 0
weights: [0.01, 0.08, 0.08]
y=1, t=0 =>Incorrect


inputs: 0 1
weights: [-0.19, 0.08, 0.08]
y=0, t=0 =>Correct
```

```
inputs: 1 0
weights: [-0.19, 0.08, 0.08]
y=0, t=0 =>Correct


inputs: 1 1
weights: [-0.19, 0.08, 0.08]
y=0, t=1 =>Incorrect


total corrects: 2
**********************************************************************
epoch: 2


inputs: 0 0
weights: [0.01, 0.28, 0.28]
y=1, t=0 =>Incorrect


inputs: 0 1
weights: [-0.19, 0.28, 0.28]
y=1, t=0 =>Incorrect


inputs: 1 0
weights: [-0.39, 0.28, 0.08]
y=0, t=0 =>Correct


inputs: 1 1
weights: [-0.39, 0.28, 0.08]
y=0, t=1 =>Incorrect


total corrects: 1
**********************************************************************
epoch: 3


inputs: 0 0
weights: [-0.19, 0.48, 0.28]
y=0, t=0 =>Correct


inputs: 0 1
weights: [-0.19, 0.48, 0.28]
y=1, t=0 =>Incorrect


inputs: 1 0
weights: [-0.39, 0.48, 0.08]
y=1, t=0 =>Incorrect


inputs: 1 1
weights: [-0.59, 0.28, 0.08]
y=0, t=1 =>Incorrect


total corrects: 1
**********************************************************************
epoch: 4
```

```
inputs: 0 0
weights: [-0.39, 0.48, 0.28]
y=0, t=0 =>Correct


inputs: 0 1
weights: [-0.39, 0.48, 0.28]
y=0, t=0 =>Correct


inputs: 1 0
weights: [-0.39, 0.48, 0.28]
y=1, t=0 =>Incorrect


inputs: 1 1
weights: [-0.59, 0.28, 0.28]
y=0, t=1 =>Incorrect


total corrects: 2
********************************************************************************
epoch: 5


inputs: 0 0
weights: [-0.39, 0.48, 0.48]
y=0, t=0 =>Correct


inputs: 0 1
weights: [-0.39, 0.48, 0.48]
y=1, t=0 =>Incorrect


inputs: 1 0
weights: [-0.59, 0.48, 0.28]
y=0, t=0 =>Correct


inputs: 1 1
weights: [-0.59, 0.48, 0.28]
y=1, t=1 =>Correct


total corrects: 3
********************************************************************************
epoch: 6


inputs: 0 0
weights: [-0.59, 0.48, 0.28]
y=0, t=0 =>Correct


inputs: 0 1
weights: [-0.59, 0.48, 0.28]
y=0, t=0 =>Correct


inputs: 1 0
weights: [-0.59, 0.48, 0.28]
y=0, t=0 =>Correct
```

```
inputs: 1 1
weights: [-0.59, 0.48, 0.28]
y=1, t=1 =>Correct


total corrects: 4
final weights [-0.59, 0.48, 0.28]
```

In [6]:

```
weights = [0.01,0.08, 0.08]
Train('OR',lst)
```

```
Initial weights [0.01, 0.08, 0.08]
**************************************************************************
epoch: 1


inputs: 0 0
weights: [0.01, 0.08, 0.08]
y=1, t=0 =>Incorrect


inputs: 0 1
weights: [-0.19, 0.08, 0.08]
y=0, t=1 =>Incorrect


inputs: 1 0
weights: [0.01, 0.08, 0.28]
y=1, t=1 =>Correct


inputs: 1 1
weights: [0.01, 0.08, 0.28]
y=1, t=1 =>Correct


total corrects: 2
**************************************************************************
epoch: 2


inputs: 0 0
weights: [0.01, 0.08, 0.28]
y=1, t=0 =>Incorrect


inputs: 0 1
weights: [-0.19, 0.08, 0.28]
y=1, t=1 =>Correct


inputs: 1 0
weights: [-0.19, 0.08, 0.28]
y=0, t=1 =>Incorrect


inputs: 1 1
weights: [0.01, 0.28, 0.28]
y=1, t=1 =>Correct


total corrects: 2
```

```
**********************************************************************
epoch: 3


inputs: 0 0
weights: [0.01, 0.28, 0.28]
y=1, t=0 =>Incorrect


inputs: 0 1
weights: [-0.19, 0.28, 0.28]
y=1, t=1 =>Correct


inputs: 1 0
weights: [-0.19, 0.28, 0.28]
y=1, t=1 =>Correct


inputs: 1 1
weights: [-0.19, 0.28, 0.28]
y=1, t=1 =>Correct


total corrects: 3
**********************************************************************
epoch: 4


inputs: 0 0
weights: [-0.19, 0.28, 0.28]
y=0, t=0 =>Correct


inputs: 0 1
weights: [-0.19, 0.28, 0.28]
y=1, t=1 =>Correct


inputs: 1 0
weights: [-0.19, 0.28, 0.28]
y=1, t=1 =>Correct


inputs: 1 1
weights: [-0.19, 0.28, 0.28]
y=1, t=1 =>Correct


total corrects: 4
final weights [-0.19, 0.28, 0.28]
```

In [7]:

```
weights = [0.01,0.08, 0.08]
Train('NAND',lst)
```

```
Initial weights [0.01, 0.08, 0.08]
**********************************************************************
epoch: 1


inputs: 0 0
weights: [0.01, 0.08, 0.08]
y=1, t=1 =>Correct
```

```
inputs: 0 1
weights: [0.01, 0.08, 0.08]
y=1, t=1 =>Correct


inputs: 1 0
weights: [0.01, 0.08, 0.08]
y=1, t=1 =>Correct


inputs: 1 1
weights: [0.01, 0.08, 0.08]
y=1, t=0 =>Incorrect


total corrects: 3
**************************************************************************
epoch: 2


inputs: 0 0
weights: [-0.19, -0.12, -0.12]
y=0, t=1 =>Incorrect


inputs: 0 1
weights: [0.01, -0.12, -0.12]
y=0, t=1 =>Incorrect


inputs: 1 0
weights: [0.21, -0.12, 0.08]
y=1, t=1 =>Correct


inputs: 1 1
weights: [0.21, -0.12, 0.08]
y=1, t=0 =>Incorrect


total corrects: 1
**************************************************************************
epoch: 3


inputs: 0 0
weights: [0.01, -0.32, -0.12]
y=1, t=1 =>Correct


inputs: 0 1
weights: [0.01, -0.32, -0.12]
y=0, t=1 =>Incorrect


inputs: 1 0
weights: [0.21, -0.32, 0.08]
y=0, t=1 =>Incorrect


inputs: 1 1
weights: [0.41, -0.12, 0.08]
y=1, t=0 =>Incorrect
```

```
total corrects: 1
********************************************************************************
epoch: 4


inputs: 0 0
weights: [0.21, -0.32, -0.12]
y=1, t=1 =>Correct


inputs: 0 1
weights: [0.21, -0.32, -0.12]
y=1, t=1 =>Correct


inputs: 1 0
weights: [0.21, -0.32, -0.12]
y=0, t=1 =>Incorrect


inputs: 1 1
weights: [0.41, -0.12, -0.12]
y=1, t=0 =>Incorrect


total corrects: 2
********************************************************************************
epoch: 5


inputs: 0 0
weights: [0.21, -0.32, -0.32]
y=1, t=1 =>Correct


inputs: 0 1
weights: [0.21, -0.32, -0.32]
y=0, t=1 =>Incorrect


inputs: 1 0
weights: [0.41, -0.32, -0.12]
y=1, t=1 =>Correct


inputs: 1 1
weights: [0.41, -0.32, -0.12]
y=0, t=0 =>Correct


total corrects: 3
********************************************************************************
epoch: 6


inputs: 0 0
weights: [0.41, -0.32, -0.12]
y=1, t=1 =>Correct


inputs: 0 1
weights: [0.41, -0.32, -0.12]
y=1, t=1 =>Correct
```

```
inputs: 1 0
weights: [0.41, -0.32, -0.12]
y=1, t=1 =>Correct


inputs: 1 1
weights: [0.41, -0.32, -0.12]
y=0, t=0 =>Correct


total corrects: 4
final weights [0.41, -0.32, -0.12]
```

In [8]:

```
weights = [0.01,0.08, 0.08]
Train('NOR',lst)
```

```
Initial weights [0.01, 0.08, 0.08]
**************************************************************************
epoch: 1


inputs: 0 0
weights: [0.01, 0.08, 0.08]
y=1, t=1 =>Correct


inputs: 0 1
weights: [0.01, 0.08, 0.08]
y=1, t=0 =>Incorrect


inputs: 1 0
weights: [-0.19, 0.08, -0.12]
y=0, t=0 =>Correct


inputs: 1 1
weights: [-0.19, 0.08, -0.12]
y=0, t=0 =>Correct


total corrects: 3
**************************************************************************
epoch: 2


inputs: 0 0
weights: [-0.19, 0.08, -0.12]
y=0, t=1 =>Incorrect


inputs: 0 1
weights: [0.01, 0.08, -0.12]
y=0, t=0 =>Correct


inputs: 1 0
weights: [0.01, 0.08, -0.12]
y=1, t=0 =>Incorrect


inputs: 1 1
weights: [-0.19, -0.12, -0.12]
```

```
y=0, t=0 =>Correct


total corrects: 2
********************************************************************************
epoch: 3


inputs: 0 0
weights: [-0.19, -0.12, -0.12]
y=0, t=1 =>Incorrect


inputs: 0 1
weights: [0.01, -0.12, -0.12]
y=0, t=0 =>Correct


inputs: 1 0
weights: [0.01, -0.12, -0.12]
y=0, t=0 =>Correct


inputs: 1 1
weights: [0.01, -0.12, -0.12]
y=0, t=0 =>Correct


total corrects: 3
********************************************************************************
epoch: 4


inputs: 0 0
weights: [0.01, -0.12, -0.12]
y=1, t=1 =>Correct


inputs: 0 1
weights: [0.01, -0.12, -0.12]
y=0, t=0 =>Correct


inputs: 1 0
weights: [0.01, -0.12, -0.12]
y=0, t=0 =>Correct


inputs: 1 1
weights: [0.01, -0.12, -0.12]
y=0, t=0 =>Correct


total corrects: 4
final weights [0.01, -0.12, -0.12]
```

In [ ]: