Single Perceptron for n bit data where 60% is train data and 40% is test data

Generating n bit of data

In [1]:

```
n = int(input('Enter Number of bits : '))
count = 0
i = n
string = 'bit_'
total_number = 2 ** n
```

In [2]:

```
value = list()
dictionary = dict()

while i >= 1 :
    key = string + str(i)
    d = 2 ** count

    while len(value) != total_number:
        for j in range(d):
            value.append(0)
        for j in range(d):
            value.append(1)

    dictionary[key] = value
    value = list()
    count = count + 1
    i = i - 1
```

In [3]:

```
# dictionary
```

In [4]:

```
#list(dictionary.items())
```

In [5]:

```
l = list(dictionary.items())
# l
```

In [6]:

```
reversed_dictionary = dict()
i = n-1
while i >= 0:
    reversed_dictionary[l[i][0]] = l[i][1]
    i = i - 1
# reversed_dictionary
```

In [7]:

```
dictionary = reversed_dictionary
# dictionary
```

In [8]:

```
output = dictionary['bit_1']
```

```
# output
```

In [9]:

```
import pandas as pd

df = pd.DataFrame(data=dictionary)
df
```

Out[9]:

	bit_1	bit_2	bit_3	bit_4	bit_5	bit_6
0	0	0	0	0	0	0
1	0	0	0	0	0	1
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	0	0	1	0	0
...	...	...	...	...	...	...
59	1	1	1	0	1	1
60	1	1	1	1	0	0
61	1	1	1	1	0	1
62	1	1	1	1	1	0
63	1	1	1	1	1	1

64 rows × 6 columns

In [81]:

```
df['Output'] = output
df
```

Out[81]:

	bit_1	bit_2	bit_3	bit_4	bit_5	bit_6	Output
0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0
2	0	0	0	0	1	0	0
3	0	0	0	0	1	1	0
4	0	0	0	1	0	0	0
...	...	...	...	...	...	...	...
59	1	1	1	0	1	1	1
60	1	1	1	1	0	0	1
61	1	1	1	1	0	1	1
62	1	1	1	1	1	0	1
63	1	1	1	1	1	1	1

64 rows × 7 columns

In [82]:

```
df = df.drop('Output',axis=1)
df
```

Out[82]:

	bit_1	bit_2	bit_3	bit_4	bit_5	bit_6

	bit_1	bit_2	bit_3	bit_4	bit_5	bit_6
0	0	0	0	0	0	0
1	0	0	0	0	0	1
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	0	0	1	0	0
...	...	...	...	...	...	...
59	1	1	1	0	1	1
60	1	1	1	1	0	0
61	1	1	1	1	0	1
62	1	1	1	1	1	0
63	1	1	1	1	1	1

64 rows × 6 columns

Generating n number of random weights , a fixed threshold value and a fixed learning rate

In [83]:

```
n
```

Out[83]:

```
6
```

In [84]:

```
import numpy as np
```

In [85]:

```
np.random.seed(42)
weights = np.random.rand(n)
weights
```

Out[85]:

```
array([0.37454012, 0.95071431, 0.73199394, 0.59865848, 0.15601864,
       0.15599452])
```

In [86]:

```
threshold = 0.5
threshold
```

Out[86]:

```
0.5
```

In [87]:

```
learning_rate = 0.4
learning_rate
```

Out[87]:

```
0.4
```

Train Test

In [88]:

```
train_percentage = 60
```

```
test_percentage = 100 - train_percentage

print('Train Percentage :',train_percentage)
print('Test Percentage :',test_percentage)
```

```
Train Percentage : 60
Test Percentage : 40
```

In [89]:

```
import math

no_of_train_data = math.ceil(( total_number * train_percentage ) / 100)
no_of_test_data = total_number - no_of_train_data

print('No of Train Data :',no_of_train_data)
print('No of Test Data :',no_of_test_data)
```

```
No of Train Data : 39
No of Test Data : 25
```

Adjusting Weights

In [90]:

```
total_number
```

Out[90]:

```
64
```

In [91]:

```
# df.columns
# df.columns[0]
# df[df.columns[0]]
df[df.columns[0]]
```

Out[91]:

```
0      0
1      0
2      0
3      0
4      0
      ..
59     1
60     1
61     1
62     1
63     1
Name: bit_1, Length: 64, dtype: int64
```

In [92]:

```
counter = 0

while counter!=no_of_train_data :

    for i in range(no_of_train_data):
        summation = 0

        # Take the first row in training data as input and
        # multiply all input data with corresponding weights and take the summation of it
        for j in range(n):
            summation = summation + df.iloc[i,j] * weights[j]

        if summation >= threshold :
            predicted_output = 1

        else:
            predicted_output = 0

        if predicted_output == output[i]:
            counter = counter + 1

        # weight updation happens if doesn't match
        else:
            difference = output[i] - predicted_output
            for j in range(n):
                weights[j] = weights[j] + learning_rate * difference * df.iloc[i,j]
            counter = 0
            break
```

In [93]:

```
weights
```

Out[93]:

```
array([ 1.17454012,  0.15071431, -0.06800606, -0.20134152, -0.24398136,
       -0.24400548])
```

In [94]:

```
# proof that these weights are valid for train data

for i in range(no_of_train_data) :
    summation = 0

    for j in range(n):
        summation = summation + df[df.columns[j]][i] * weights[j]

    if summation >= threshold :
        predicted_output = 1
    else:
        predicted_output = 0

    print(predicted_output)
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
1
```

```
1
1
1
1
1
1
```

In [95]:

```
right = 0
wrong = 0

for i in range(no_of_train_data,total_number) :
    summation = 0

    for j in range(n):
        summation = summation + df[df.columns[j]][i] * weights[j]

    if summation >= threshold :
        predicted_output = 1

    else:
        predicted_output = 0

    if predicted_output == output[i]:
        right = right + 1

    else:
        wrong = wrong + 1

accuracy = ( right * 100 ) / no_of_test_data

print('No of Test Data :',no_of_test_data)
print('Right :',right)
print('Wrong :',wrong)
print('Accuracy :',accuracy)
```

```
No of Test Data : 25
Right : 23
Wrong : 2
Accuracy : 92.0
```