Single Perceptron for n bit data where 80% is train data and 20% is test data and Groupwise learning

Generating n bit of data

In [2]:

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

In [3]:

```
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 [4]:

```
# dictionary
```

In [5]:

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

In [6]:

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

In [7]:

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

In [8]:

```
dictionary = reversed_dictionary
#dictionary
```

In [9]:

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

```
#output
```

In [10]:

```
import pandas as pd

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

Out[10]:

	bit_1	bit_2	bit_3	bit_4	bit_5	bit_6	bit_7	bit_8	bit_9	bit_10
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	0	0	1	1
4	0	0	0	0	0	0	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...
1019	1	1	1	1	1	1	1	0	1	1
1020	1	1	1	1	1	1	1	1	0	0
1021	1	1	1	1	1	1	1	1	0	1
1022	1	1	1	1	1	1	1	1	1	0
1023	1	1	1	1	1	1	1	1	1	1

1024 rows × 10 columns

In [229]:

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

Out[229]:

	bit_1	bit_2	bit_3	bit_4	bit_5	bit_6	bit_7	bit_8	bit_9	bit_10	Output
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0
2	0	0	0	0	0	0	0	0	1	0	0
3	0	0	0	0	0	0	0	0	1	1	0
4	0	0	0	0	0	0	0	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...
1019	1	1	1	1	1	1	1	0	1	1	1
1020	1	1	1	1	1	1	1	1	0	0	1
1021	1	1	1	1	1	1	1	1	0	1	1
1022	1	1	1	1	1	1	1	1	1	0	1
1023	1	1	1	1	1	1	1	1	1	1	1

1024 rows × 11 columns

In [230]:

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

Out[230]:

	bit_1	bit_2	bit_3	bit_4	bit_5	bit_6	bit_7	bit_8	bit_9	bit_10

	bit_1	bit_2	bit_3	bit_4	bit_5	bit_6	bit_7	bit_8	bit_9	bit_10
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	0	0	1	1
4	0	0	0	0	0	0	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...
1019	1	1	1	1	1	1	1	0	1	1
1020	1	1	1	1	1	1	1	1	0	0
1021	1	1	1	1	1	1	1	1	0	1
1022	1	1	1	1	1	1	1	1	1	0
1023	1	1	1	1	1	1	1	1	1	1

1024 rows × 10 columns

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

In [231]:

```
n
```

Out[231]:

```
10
```

In [232]:

```
import numpy as np
```

In [233]:

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

Out[233]:

```
array([0.37454012, 0.95071431, 0.73199394, 0.59865848, 0.15601864,
       0.15599452, 0.05808361, 0.86617615, 0.60111501, 0.70807258])
```

In [234]:

```
threshold = 0.5
threshold
```

Out[234]:

```
0.5
```

In [235]:

```
learning_rate = 0.4
learning_rate
```

Out[235]:

```
0.4
```

Train Test

In [236]:

```
train_percentage = 80
```

```
test_percentage = 100 - train_percentage

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

```
Train Percentage : 80
Test Percentage : 20
```

In [237]:

```
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 : 820
No of Test Data : 204
```

Adjusting Weights

In [238]:

```
total_number
```

Out[238]:

```
1024
```

In [11]:

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

In [240]:

```
# training 0 class and getting weights

counter = 0

while counter!=total_number/2 :

    for i in range(int(total_number/2)):
        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]:
            counter = counter + 1

        else:
            difference = output[i] - predicted_output
            for j in range(n):
                weights[j] = weights[j] + learning_rate * difference * df[df.columns[j]]
[i]
            counter = 0
            break
```

In [241]:

```
weights
```

Out[241]:

```
array([ 0.37454012, -0.24928569, -0.06800606, -0.20134152, -0.24398136,
       -0.24400548, -0.34191639, -0.33382385, -0.19888499, -0.09192742])
```

In [242]:

```
while True:
    # training 1 class and getting weights
    counter = 0

    while counter!=no_of_train_data - total_number/2 :

        for i in range(int(total_number/2),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

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

            else:
                difference = output[i] - predicted_output
                for j in range(n):
                    weights[j] = weights[j] + learning_rate * difference * df[df.columns
[j]][i]
                counter = 0
                break


    # weights gained after training class 1 are applied on the 0 class again
    right = 0
    wrong = 0

    for i in range(int(total_number/2)) :
        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

    # if weights gained after training class 1 works on all 0 class then break the loop
    if wrong == 0:
        break
    # otherwise train the 0 class with new weights
    else:
        counter = 0

        while counter!=total_number/2 :

            for i in range(int(total_number/2)):
                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]:
                    counter = counter + 1

                else:
                    difference = output[i] - predicted_output
                    for j in range(n):
                        weights[j] = weights[j] + learning_rate * difference * df[df.col
umns[j]][i]
                    counter = 0
                    break
```

In [243]:

```
weights
```

Out[243]:

```
array([ 1.57454012, -0.24928569, -0.06800606, -0.20134152, -0.24398136,
       -0.24400548,  0.05808361,  0.06617615, -0.19888499, -0.09192742])
```

In [1]:

```
# # 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)
```

In [245]:

```
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 : 204
Right : 188
Wrong : 16
Accuracy : 92.15686274509804
```