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(-1)
        for j in range(d):
            value.append(1)

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

In [3]:

```
dictionary
```

Out[3]:

```
{'bit_4': [-1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1],
 'bit_3': [-1, -1, 1, 1, -1, -1, 1, 1, -1, -1, 1, 1, -1, -1, 1, 1],
 'bit_2': [-1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, 1, 1, 1, 1],
 'bit_1': [-1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1]}
```

In [4]:

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

Out[4]:

```
[('bit_4', [-1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1]),
 ('bit_3', [-1, -1, 1, 1, -1, -1, 1, 1, -1, -1, 1, 1, -1, -1, 1, 1]),
 ('bit_2', [-1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, 1, 1, 1, 1]),
 ('bit_1', [-1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1])]
```

In [5]:

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

Out[5]:

```
[('bit_4', [-1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1]),
 ('bit_3', [-1, -1, 1, 1, -1, -1, 1, 1, -1, -1, 1, 1, -1, -1, 1, 1]),
 ('bit_2', [-1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, 1, 1, 1, 1]),
 ('bit_1', [-1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1])]
```

In [6]:

```
reversed_dictionary = dict()
i = n-1
```

```
while i >= 0:
    reversed_dictionary[l[i][0]] = l[i][1]
    i = i - 1
reversed_dictionary
```

Out[6]:

```
{'bit_1': [-1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1],
 'bit_2': [-1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, 1, 1, 1, 1],
 'bit_3': [-1, -1, 1, 1, -1, -1, 1, 1, -1, -1, 1, 1, -1, -1, 1, 1],
 'bit_4': [-1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1]}
```

In [7]:

```
dictionary = reversed_dictionary
dictionary
```

Out[7]:

```
{'bit_1': [-1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1],
 'bit_2': [-1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, 1, 1, 1, 1],
 'bit_3': [-1, -1, 1, 1, -1, -1, 1, 1, -1, -1, 1, 1, -1, -1, 1, 1],
 'bit_4': [-1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1]}
```

In [8]:

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

Out[8]:

```
[-1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1]
```

In [9]:

```
import pandas as pd

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

Out[9]:

	bit_1	bit_2	bit_3	bit_4
0	-1	-1	-1	-1
1	-1	-1	-1	1
2	-1	-1	1	-1
3	-1	-1	1	1
4	-1	1	-1	-1
5	-1	1	-1	1
6	-1	1	1	-1
7	-1	1	1	1
8	1	-1	-1	-1
9	1	-1	-1	1
10	1	-1	1	-1
11	1	-1	1	1
12	1	1	-1	-1
13	1	1	-1	1
14	1	1	1	-1
15	1	1	1	1

In [27]:

```
# saving the dataframe
```

```
# saving the dataframe
df.to_csv('file1.csv')
```

Train Test

In [10]:

```
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 [11]:

```
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 : 10
No of Test Data : 6
```

Weight Adjusting

In [12]:

```
df.shape
```

Out[12]:

```
(16, 4)
```

In [13]:

```
m = df.shape[1]
m
```

Out[13]:

```
4
```

In [14]:

```
import numpy as np
w = np.zeros((m,m))
w
```

Out[14]:

```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

In [15]:

```
summation = 0

for i in range(m):
    for j in range(m):
        if i != j :
            for pattern in range(no_of_train_data):
                summation = summation + df.iloc[pattern,i] * df.iloc[pattern,j]
```

```
            w[i,j] = summation
            summation = 0
```

In [16]:

```
w
```

Out[16]:

```
array([[ 0., -2., -2.,  0.],
       [-2.,  0.,  2.,  0.],
       [-2.,  2.,  0.,  0.],
       [ 0.,  0.,  0.,  0.]])
```

Pattern matching by taking test data

In [17]:

```
new_pattern = list()
temp = list()
sum = 0
count = 1
flag = True

for pattern in range(no_of_train_data,total_number):
    for j in range(m):
        new_pattern.append(df.iloc[pattern,j])
    print('New Pattern :',new_pattern)

    while flag:
        for row in range(m):
            for j in range(m):
                sum = sum + w[row,j] * new_pattern[j]

            if sum > 0 :
                new_pattern[row] = 1
            elif sum < 0 :
                new_pattern[row] = -1

            print('At Neuron ',row,':',new_pattern)

            if len(temp) == 0 :
                temp = new_pattern.copy()
            else:
                if temp == new_pattern:
                    count = count + 1
                else:
                    count = 1
                temp = new_pattern.copy()

            if count >= 5:
                flag = False
                break

            sum = 0

    print('Converged pattern of the test pattern :',new_pattern)

    for p in range(no_of_train_data):
        if new_pattern == list(df.iloc[p]):
            print('Cluster with',p)
            print('-------------------------------------------------------')
            break

    new_pattern = list()
    temp = list()
    sum = 0
    count = 1
    flag = True
```

New Pattern : [1, -1, 1, -1]

```
At Neuron  0 : [1, -1, 1, -1]
At Neuron  1 : [1, -1, 1, -1]
At Neuron  2 : [1, -1, -1, -1]
At Neuron  3 : [1, -1, -1, -1]
At Neuron  0 : [1, -1, -1, -1]
At Neuron  1 : [1, -1, -1, -1]
At Neuron  2 : [1, -1, -1, -1]
Converged pattern of the test pattern : [1, -1, -1, -1]
Cluster with 8
-------------------------------------------------------
New Pattern : [1, -1, 1, 1]
At Neuron  0 : [1, -1, 1, 1]
At Neuron  1 : [1, -1, 1, 1]
At Neuron  2 : [1, -1, -1, 1]
At Neuron  3 : [1, -1, -1, 1]
At Neuron  0 : [1, -1, -1, 1]
At Neuron  1 : [1, -1, -1, 1]
At Neuron  2 : [1, -1, -1, 1]
Converged pattern of the test pattern : [1, -1, -1, 1]
Cluster with 9
-------------------------------------------------------
New Pattern : [1, 1, -1, -1]
At Neuron  0 : [1, 1, -1, -1]
At Neuron  1 : [1, -1, -1, -1]
At Neuron  2 : [1, -1, -1, -1]
At Neuron  3 : [1, -1, -1, -1]
At Neuron  0 : [1, -1, -1, -1]
At Neuron  1 : [1, -1, -1, -1]
Converged pattern of the test pattern : [1, -1, -1, -1]
Cluster with 8
-------------------------------------------------------
New Pattern : [1, 1, -1, 1]
At Neuron  0 : [1, 1, -1, 1]
At Neuron  1 : [1, -1, -1, 1]
At Neuron  2 : [1, -1, -1, 1]
At Neuron  3 : [1, -1, -1, 1]
At Neuron  0 : [1, -1, -1, 1]
At Neuron  1 : [1, -1, -1, 1]
Converged pattern of the test pattern : [1, -1, -1, 1]
Cluster with 9
-------------------------------------------------------
New Pattern : [1, 1, 1, -1]
At Neuron  0 : [-1, 1, 1, -1]
At Neuron  1 : [-1, 1, 1, -1]
At Neuron  2 : [-1, 1, 1, -1]
At Neuron  3 : [-1, 1, 1, -1]
At Neuron  0 : [-1, 1, 1, -1]
Converged pattern of the test pattern : [-1, 1, 1, -1]
Cluster with 6
-------------------------------------------------------
New Pattern : [1, 1, 1, 1]
At Neuron  0 : [-1, 1, 1, 1]
At Neuron  1 : [-1, 1, 1, 1]
At Neuron  2 : [-1, 1, 1, 1]
At Neuron  3 : [-1, 1, 1, 1]
At Neuron  0 : [-1, 1, 1, 1]
Converged pattern of the test pattern : [-1, 1, 1, 1]
Cluster with 7
-------------------------------------------------------
```

----------Just Clearing my doubts----------

In [18]:

```
t = [1,2,2]
m = [1,2,3]
t
```

Out[18]:

```
[1, 2, 2]
```

[1, 2, 2]

In [19]:

```
if t == m:
    print('hi')
```

In [20]:

```
df
```

Out[20]:

	bit_1	bit_2	bit_3	bit_4
0	-1	-1	-1	-1
1	-1	-1	-1	1
2	-1	-1	1	-1
3	-1	-1	1	1
4	-1	1	-1	-1
5	-1	1	-1	1
6	-1	1	1	-1
7	-1	1	1	1
8	1	-1	-1	-1
9	1	-1	-1	1
10	1	-1	1	-1
11	1	-1	1	1
12	1	1	-1	-1
13	1	1	-1	1
14	1	1	1	-1
15	1	1	1	1

In [21]:

```
m = [-1,-1,-1]
```

In [22]:

```
df.iloc[0]
```

Out[22]:

```
bit_1   -1
bit_2   -1
bit_3   -1
bit_4   -1
Name: 0, dtype: int64
```

In [23]:

```
list(df.iloc[0])
```

Out[23]:

```
[-1, -1, -1, -1]
```

In [24]:

```
if m == list(df.iloc[0]):
    print('hi')
```

In [25]:

```
m = [1,2,3]
t = []
t = m.copy()
print(m)
print(t)
```

```
[1, 2, 3]
[1, 2, 3]
```

In [26]:

```
m[0] = 10
print(m)
print(t)
```

```
[10, 2, 3]
[1, 2, 3]
```