Experiment No. 5

Aim: Implementation of Expectation-Maximization Algorithm.

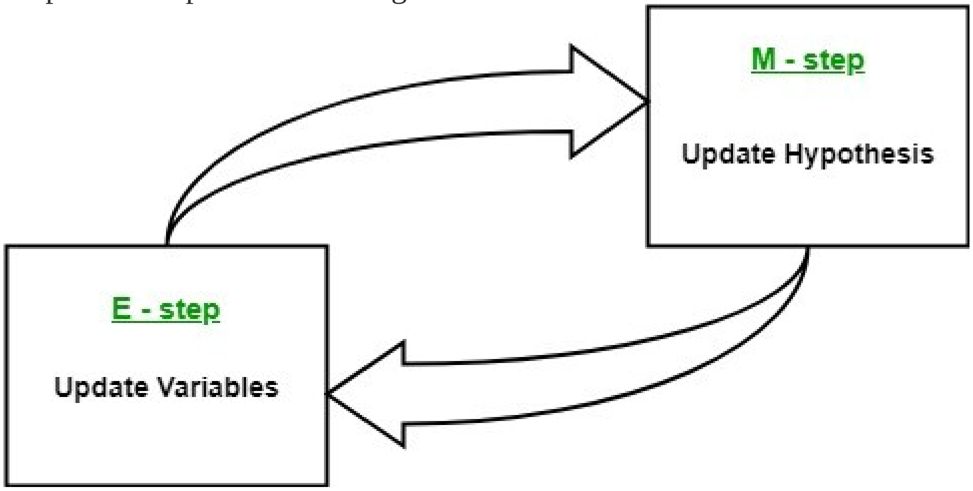
Theory:

In the real-world applications of machine learning, it is very common that there are many relevant features available for learning but only a small subset of them are observable. So, for the variables which are sometimes observable and sometimes not, then we can use the instances when that variable is visible is observed for the purpose of learning and then predict its value in the instances when it is not observable.

On the other hand, ***Expectation-Maximization algorithm*** can be used for the latent variables (variables that are not directly observable and are actually inferred from the values of the other observed variables) too in order to predict their values with the condition that the general form of probability distribution governing those latent variables is known to us. This algorithm is actually at the base of many unsupervised clustering algorithms in the field of machine learning. It was explained, proposed and given its name in a paper published in 1977 by Arthur Dempster, Nan Laird, and Donald Rubin. It is used to find the *local maximum likelihood parameters* of a statistical model in the cases where latent variables are involved and the data is missing or incomplete.

Algorithm:

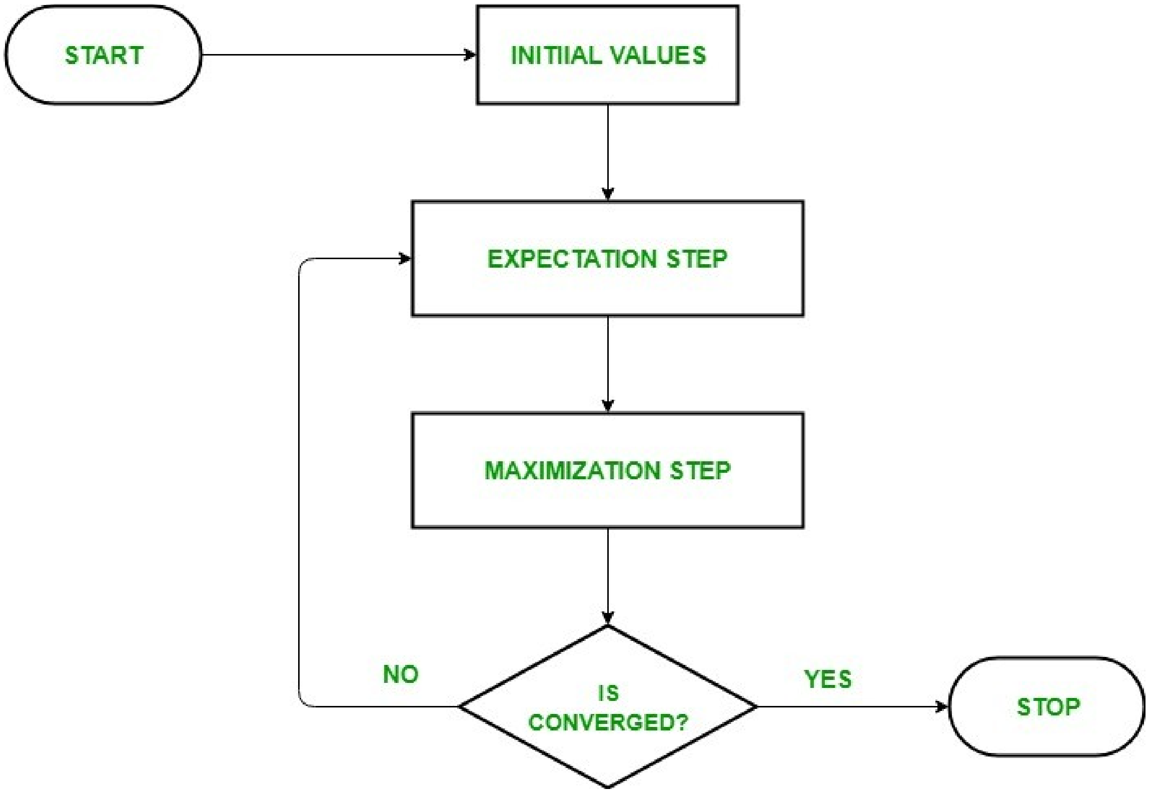
1. Given a set of incomplete data, consider a set of starting parameters.
2. **Expectation step (E – step):** Using the observed available data of the dataset, estimate (guess) the values of the missing data.
3. **Maximization step (M – step):** Complete data generated after the expectation (E) step is used in order to update the parameters.
4. Repeat step 2 and step 3 until convergence.



The essence of Expectation-Maximization algorithm is to use the available observed data of the dataset to estimate the missing data and then using that data to update the values of the parameters.

- Initially, a set of initial values of the parameters are considered. A set of incomplete observed data is given to the system with the assumption that the observed data comes from a specific model.
- The next step is known as "Expectation" – step or *E-step*. In this step, we use the observed data in order to estimate or guess the values of the missing or incomplete data. It is basically used to update the variables.
- The next step is known as "Maximization"-step or *M-step*. In this step, we use the complete data generated in the preceding "Expectation" – step in order to update the values of the parameters. It is basically used to update the hypothesis.
- Now, in the fourth step, it is checked whether the values are converging or not, if yes, then stop otherwise repeat *step-2* and *step-3* i.e. "Expectation" – step and "Maximization" – step until the convergence occurs.

Flow chart for EM algorithm –



Usage of EM algorithm –

- It can be used to fill the missing data in a sample.
- It can be used as the basis of unsupervised learning of clusters.
- It can be used for the purpose of estimating the parameters of Hidden Markov Model (HMM).
- It can be used for discovering the values of latent variables.

Advantages of EM algorithm –

- It is always guaranteed that likelihood will increase with each iteration.
- The E-step and M-step are often pretty easy for many problems in terms of implementation.
- Solutions to the M-steps often exist in the closed form.

Disadvantages of EM algorithm –

- It has slow convergence.
- It makes convergence to the local optima only.
- It requires both the probabilities, forward and backward (numerical optimization requires only forward probability).

Implementation:

Expectation -Maximization algorithm:

```
import random
r1 = ["H","H","H","H","H","T","T","T","T","T"]
r2 = ["H","H","H","H","H","H","H","H","H","T"]
r3 = ["H","H","H","H","H","H","H","H","T","T"]
r4 = ["H","H","H","H","T","T","T","T","T","T"]
r5 = ["H","H","H","H","H","H","H","T","T","T"]
d = [r1,r2,r3,r4,r5]

pa = random.random()
pb = random.random()
# pa = 0.6
# pb = 0.5
sah = 0
sat = 0
sbh = 0
sbt = 0
```

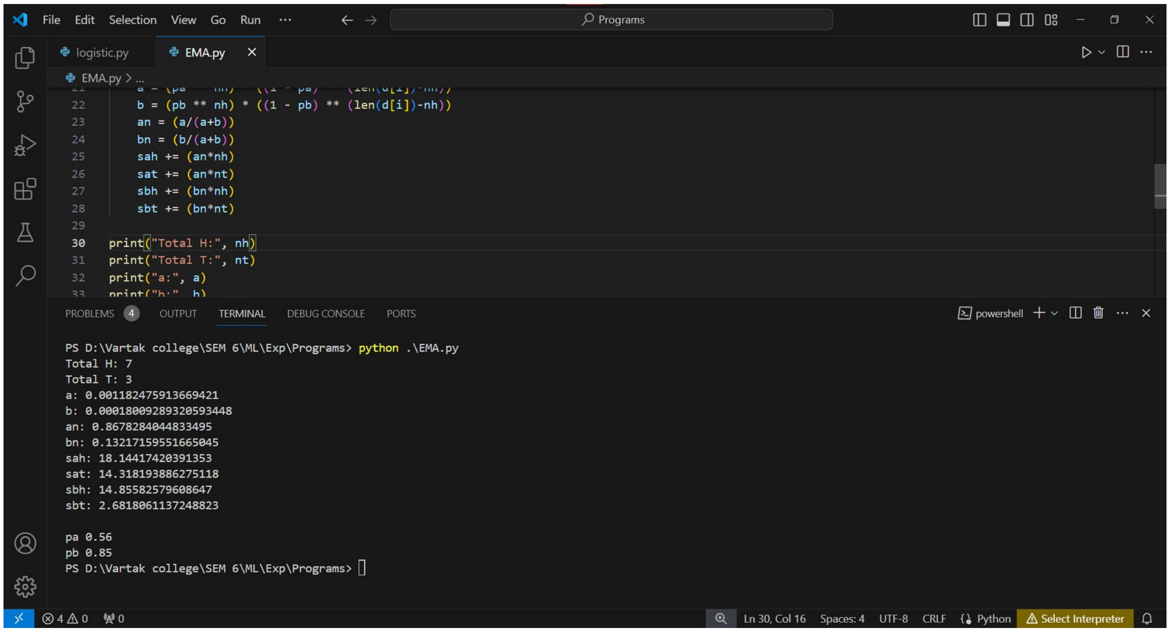
```
for i in range(0,5):
    nh = d[i].count("H")
    nt = d[i].count("T")
    a = (pa ** nh) * ((1 - pa) ** (len(d[i])-nh))
    b = (pb ** nh) * ((1 - pb) ** (len(d[i])-nh))
    an = (a/(a+b))
    bn = (b/(a+b))
    sah += (an*nh)
    sat += (an*nt)
    sbh += (bn*nh)
    sbt += (bn*nt)

print("Total H:", nh)
print("Total T:", nt)
print("a:", a)
print("b:", b)
print("an:", an)
print("bn:", bn)
print("sah:", sah)
print("sat:", sat)
print("sbh:", sbh)
print("sbt:", sbt)

fpa = (sah/(sah+sat))
fpb = (sbh/(sbh+sbt))

print("\npa",round(fpa,2))
print("pb",round(fpb,2))
```

Output:



```
        a = (pa ** nh) * ((1 - pa) ** (len(d[i])-nh))
        b = (pb ** nh) * ((1 - pb) ** (len(d[i])-nh))
        an = (a/(a+b))
        bn = (b/(a+b))
        sah += (an*nh)
        sat += (an*nt)
        sbh += (bn*nh)
        sbt += (bn*nt)

    print("Total H:", nh)
    print("Total T:", nt)
    print("a:", a)
    print("b:", b)
```

```
PS D:\Vartak college\SEM 6\ML\Exp\Programs> python .\EMA.py
Total H: 7
Total T: 3
a: 0.001182475913669421
b: 0.00018009289320593448
an: 0.8678284044833495
bn: 0.13217159551665045
sah: 18.14417420391353
sat: 14.318193886275118
sbh: 14.85582579608647
sbt: 2.6818061137248823

pa 0.56
pb 0.85
PS D:\Vartak college\SEM 6\ML\Exp\Programs>
```

Conclusion:

1) What is the use of Expectation-Maximization Algorithm?

The Expectation-Maximization (EM) Algorithm is used primarily for estimating parameters of statistical models in cases where there are latent variables, missing or incomplete data. It is commonly employed in various machine learning tasks such as clustering, filling in missing data, estimating parameters of Hidden Markov Models (HMM), and discovering the values of latent variables.

2) How it can be used to handle missing data?

The EM Algorithm can be used to handle missing data by iteratively estimating the missing values based on the observed data and updating the parameters of the model. In the Expectation (E) step, the algorithm estimates or guesses the values of the missing or incomplete data using the observed data. Then, in the Maximization (M) step, it uses the complete data generated in the E-step to update the values of the parameters of the model. This process is repeated until convergence occurs, resulting in a set of parameters that maximize the likelihood of the observed data given the model.