Experiment No. 7

AIM- Implement Expectation Maximization Algorithm Theory:

The EM algorithm is considered a latent variable model to find the local maximum likelihood parameters of a statistical model, proposed by Arthur Dempster, Nan Laird, and Donald Rubin in 1977.
The EM (Expectation-Maximization) algorithm is one of the most commonly used terms in machine learning to obtain maximum likelihood estimates of variables that are sometimes observable and sometimes not.
However, it is also applicable to unobserved data or sometimes called latent. It has various real-world applications in statistics, including obtaining the mode of the posterior marginal distribution of parameters in machine learning and data mining applications.

In most real-life applications of machine learning, it is found that several relevant learning features are available, but very few of them are observable, and the rest are unobservable. If the variables are observable, then it can predict the value using instances. On the other hand, the variables which are latent or directly not observable, for such variables Expectation-Maximization (EM) algorithm plays a vital role to predict the value with the condition that the general form of probability distribution governing those latent variables is known to us. In this topic, we will discuss a basic introduction to the EM algorithm, a flow chart of the EM algorithm, its applications, advantages, and disadvantages of EM algorithm, etc.

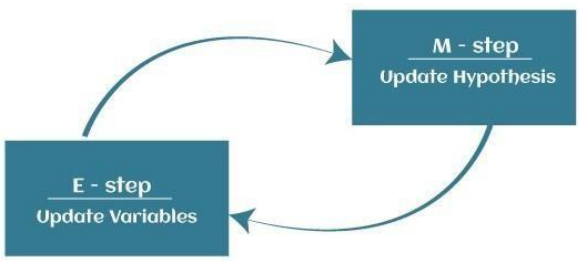
What is an EM algorithm?
The Expectation-Maximization (EM) algorithm is defined as the combination of various unsupervised machine learning algorithms, which is used to determine the local maximum likelihood estimates (MLE) or maximum a posteriori estimates (MAP) for unobservable variables in statistical models. Further, it is a

technique to find maximum likelihood estimation when the latent variables are present. It is also referred to as the latent variable model.

A latent variable model consists of both observable and unobservable variables where observable can be predicted while unobserved are inferred from the observed variable. These unobservable variables are known as latent variables.

It is known as the latent variable model to determine MLE and MAP parameters for latent variables. It is used to predict values of parameters in instances where data is missing or unobservable for learning, and this is done until convergence of the values occurs.

The EM algorithm is the combination of various unsupervised ML algorithms, such as the k-means clustering algorithm. Being an iterative approach, it consists of two modes. In the first mode, we estimate the missing or latent variables. Hence it is referred to as the Expectation/estimation step (E-step). Further,



the other mode is used to optimize the parameters of the models so that it can explain the data more clearly. The second mode is known as the maximization-step or M-step.

Expectation step (E - step): It involves the estimation (guess) of all missing values in the dataset so that after completing this step, there should not be any missing value.

Maximization step (M - step): This step involves the use of estimated data in the E-step and updating the parameters.

Repeat E-step and M-step until the convergence of the values occurs.

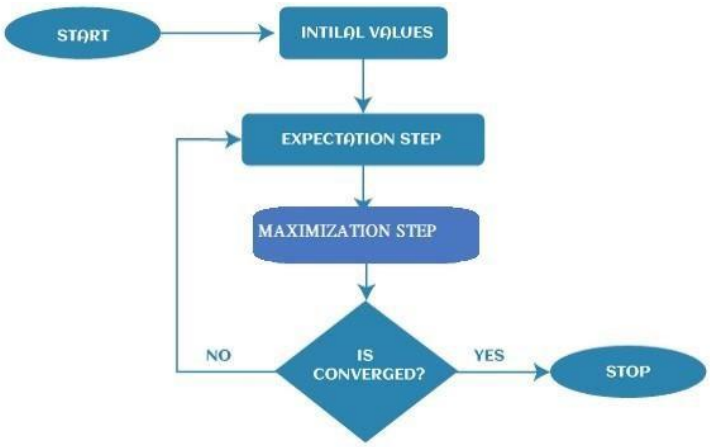
The primary goal of the EM algorithm is to use the available observed data of the dataset to estimate the missing data of the latent variables and then use that data to update the values of the parameters in the M-step.

Convergence in the EM algorithm

Convergence is defined as the specific situation in probability based on intuition, e.g., if there are two random variables that have very less difference in their probability, then they are known as converged. In other words, whenever the values of given variables are matched with each other, it is called convergence.

Steps in EM Algorithm

The EM algorithm is completed mainly in 4 steps, which include Initialization Step, Expectation Step, Maximization Step, and convergence Step. These steps are explained as follows:



EM Algorithm in Machine Learning

1st Step: The very first step is to initialize the parameter values. Further, the system is provided with incomplete observed data with the assumption that data is obtained from a specific model. **2nd Step:** This step is known as Expectation or E-Step, which is used to estimate or guess the values of the missing or incomplete data using the observed data. Further, E-step primarily updates the variables. **3rd Step:** This step is known as Maximization or M-step, where we use complete data obtained from the 2nd step to update the parameter values. Further, M-step primarily updates the hypothesis. **4th step:** The last step is to check if the values of latent variables are converging or not. If it gets "yes", then stop the process; else, repeat the process from step 2 until the convergence occurs.

Applications of EM algorithm

The primary aim of the EM algorithm is to estimate the missing data in the latent variables through observed data in datasets. The EM algorithm or latent variable model has a broad range of real-life applications in machine learning.

These are as follows:

The EM algorithm is applicable in data clustering in machine learning.

It is often used in computer vision and NLP (Natural language processing).

It is used to estimate the value of the parameter in mixed models such as the Gaussian Mixture Modeland quantitative genetics.

It is also used in psychometrics for estimating item parameters and latent abilities of item response theory models.
It is also applicable in the medical and healthcare industry, such as in image reconstruction and structural engineering.
It is used to determine the Gaussian density of a function.

Advantages of EM algorithm

It is very easy to implement the first two basic steps of the EM algorithm in various machine learning problems, which are E-step and M- step.
It is mostly guaranteed that likelihood will enhance after each iteration. It
often generates a solution for the M-step in the closed form.

Disadvantages of EM algorithm

The convergence of the EM algorithm is very slow.
It can make convergence for the local optima only.
It takes both forward and backward probability into consideration. It is opposite to that of numerical optimization, which takes only forward probabilities.

Conclusion

In real-world applications of machine learning, the expectation-maximization (EM) algorithm plays a significant role in determining the local maximum likelihood estimates (MLE) or maximum a posteriori estimates (MAP) for unobservable variables in statistical models. It is often used for the latent variables, i.e., to estimate the latent variables through observed data in datasets. It is generally completed in two important steps, i.e., the expectation step (E-step) and the Maximization step (M-Step), where E-step is used to estimate the missing data in datasets, and M-step is used to update the parameters after the complete data is generated in E-step. Further, the importance of the EM algorithm can be seen in various applications such as data clustering, natural language processing (NLP), computer vision, image reconstruction, structural engineering, etc.

POST LAB:

1. What are the challenges associated with clustering? Give real life applications.
2. Explain Gaussian mixtures.
3. What is the relation between the K-means algorithm and the EM algorithm?

```
import numpy as np


experiments = [(5, 5), (9, 1), (8, 2), (4, 6), (7, 3)]

theta_A = 0.6 theta_B
= 0.5 history = []


def binomial_probability(k, n, p):
    """Calculate binomial probability for k successes in n trials with probabilit
    return (p ** k) * ((1 - p) ** (n - k))

def run_em_iteration(theta_A, theta_B, experiments):
    """Run one iteration of EM algorithm"""
    probs_A = []
    probs_B = []

    for heads, tails in experiments:
        n = heads + tails
        likelihood_A = binomial_probability(heads, n, theta_A)
likelihood_B = binomial_probability(heads, n, theta_B)

        total = likelihood_A + likelihood_B
        prob_A = likelihood_A / total
        prob_B = likelihood_B / total

        probs_A.append(prob_A)
        probs_B.append(prob_B)

    heads_A = 0
    tails_A = 0
    heads_B = 0
    tails_B = 0

    for i, (heads, tails) in enumerate(experiments):
        heads_A += heads * probs_A[i]
        tails_A += tails * probs_A[i]
        heads_B += heads * probs_B[i]
        tails_B += tails * probs_B[i]

    new_theta_A = heads_A / (heads_A + tails_A)
    new_theta_B = heads_B / (heads_B + tails_B)

    return new_theta_A, new_theta_B, probs_A, probs_B, heads_A, tails_A, heads_B,


tolerance = 1e-4
```

```
for iteration in range(max_iterations):
    new_theta_A, new_theta_B, probs_A, probs_B, heads_A, tails_A, heads_B, tails_
history.append({
        'iteration': iteration + 1,
        'theta_A': theta_A,
        'theta_B': theta_B,
        'new_theta_A': new_theta_A,
        'new_theta_B': new_theta_B,
        'probs_A': probs_A,
        'probs_B': probs_B,
        'heads_A': heads_A,
        'tails_A': tails_A,
        'heads_B': heads_B,
        'tails_B': tails_B
    })

    if (abs(theta_A - new_theta_A) < tolerance and abs(theta_B - new_theta_B) < t
        break

    theta_A, theta_B = new_theta_A, new_theta_B

for h in history:
    print(f"\n=== Iteration {h['iteration']} ===")
    print(f"Current parameters: θ_A = {h['theta_A']:.4f}, θ_B = {h['theta_B']:.4f
    for i, (pA, pB) in enumerate(zip(h['probs_A'], h['probs_B'])):
        print(f"Experiment {i+1}: P(Coin A) = {pA:.4f}, P(Coin B) = {pB:.4f}")
    print("\nExpected counts:")
    print(f"Coin A: Heads = {h['heads_A']:.4f}, Tails = {h['tails_A']:.4f}")
print(f"Coin B: Heads = {h['heads_B']:.4f}, Tails = {h['tails_B']:.4f}")
    print(f"\nUpdated parameters: θ_A = {h['new_theta_A']:.4f}, θ_B = {h['new_the
```

```
Updated parameters: θ_A = 0.7945, θ_B = 0.5224

=== Iteration 7 ===
Current parameters: θ_A = 0.7945, θ_B = 0.5224
Experiment 1: P(Coin A) = 0.1071, P(Coin B) = 0.8929
Experiment 2: P(Coin A) = 0.9493, P(Coin B) = 0.0507
Experiment 3: P(Coin A) = 0.8413, P(Coin B) = 0.1587
Experiment 4: P(Coin A) = 0.0328, P(Coin B) = 0.9672 Experiment 5: P(Coin
A) = 0.5999, P(Coin B) = 0.4001

Expected counts:
Coin A: Heads = 20.1398, Tails = 5.1637
Coin B: Heads = 12.8602, Tails = 11.8363

Updated parameters: θ_A = 0.7959, θ_B = 0.5207

=== Iteration 8 ===
Current parameters: θ_A = 0.7959, θ_B = 0.5207
Experiment 1: P(Coin A) = 0.1046, P(Coin B) = 0.8954
Experiment 2: P(Coin A) = 0.9510, P(Coin B) = 0.0490
Experiment 3: P(Coin A) = 0.8438, P(Coin B) = 0.1562
Experiment 4: P(Coin A) = 0.0315, P(Coin B) = 0.9685
Experiment 5: P(Coin A) = 0.6007, P(Coin B) = 0.3993
```

```
    Expected counts:
    Coin A: Heads = 20.1628, Tails = 5.1526
    Coin B: Heads = 12.8372, Tails = 11.8474

    Updated parameters: θ_A = 0.7965, θ_B = 0.5200

    === Iteration 9 ===
    Current parameters: θ_A = 0.7965, θ_B = 0.5200
    Experiment 1: P(Coin A) = 0.1036, P(Coin B) = 0.8964
    Experiment 2: P(Coin A) = 0.9516, P(Coin B) = 0.0484
    Experiment 3: P(Coin A) = 0.8448, P(Coin B) = 0.1552
    Experiment 4: P(Coin A) = 0.0310, P(Coin B) = 0.9690
    Experiment 5: P(Coin A) = 0.6012, P(Coin B) = 0.3988

    Expected counts:
    Coin A: Heads = 20.1729, Tails = 5.1487
    Coin B: Heads = 12.8271, Tails = 11.8513

    Updated parameters: θ_A = 0.7967, θ_B = 0.5198

    === Iteration 10 ===
    Current parameters: θ_A = 0.7967, θ_B = 0.5198
    Experiment 1: P(Coin A) = 0.1032, P(Coin B) = 0.8968
    Experiment 2: P(Coin A) = 0.9518, P(Coin B) = 0.0482
    Experiment 3: P(Coin A) = 0.8452, P(Coin B) = 0.1548
    Experiment 4: P(Coin A) = 0.0308, P(Coin B) = 0.9692
    Experiment 5: P(Coin A) = 0.6013, P(Coin B) = 0.3987

    Expected counts:
    Coin A: Heads = 20.1772, Tails = 5.1474
    Coin B: Heads = 12.8228, Tails = 11.8526

    Updated parameters: θ_A = 0.7967, θ_B = 0.5197
```

```
print("\n=== Final Parameters ===")
print(f"θ_A converges to: {theta_A:.2f}")
print(f"θ_B converges to: {theta_B:.2f}")
```

```
    === Final Parameters ===
    θ_A converges to: 0.80 θ_B
    converges to: 0.52
```

Machine Learning

- Expectation Maximisation Numerical.

B H T T T H H T H T H.
A H H H H T H H H H H.
A H T H H H H H T H H.
B H T H T T T H H T T.
A T H H H T H H H T H.

Formula :- $^{n}C_{k}\, p^{k}(1-p)^{(n-k)}$.

Sol :-

Assumed probabilities.

$q_A = 0.6$.
$q_B = 0.5$

$P(E|Z_A) = {}^{10}C_5 (0.6)^5 (0.4)^5$

$= \frac{10 \times 9 \times 8 \times 7 \times 6}{5 \times 4 \times 3 \times 2 \times 1} = 1260 \times (0.6)^5 (0.4)^5 = 1.00$.

$P(E|Z_B) = {}^{10}C_5 (0.5)^5 (0.5)^5$

$= 1260 \times (0.5)^5 (0.5)^5$

$= 1.23$.

$P(Z_A|E) = \frac{1}{1+1.23} = 0.44$

$P(Z_B|E) = \frac{1.23}{1+1.23} = 0.55$.

$P(E|Z_A) = {}^{10}C_9 (0.6)^9 (0.4)^{(1)}$

$= 10 \cdot (0.6)^9 (0.4)^1$

$= 0.04.$

$P(E|Z_B) = {}^{10}C_9 (0.5)^9 (0.5)^1.$

$= 10 \times (0.5)^{10}.$

$= 9.76 \times 10^{-3}.$

$= \frac{9.76}{10^3}.$

$= 0.009$

(2) $P(Z_A|E) = \frac{0.04}{0.04 + 0.009} = 0.81.$

$P(Z_B|E) = \frac{0.009}{0.04 + 0.009} = 0.18.$

$P(E|Z_A) = {}^{10}C_8 (0.6)^8 (0.4)^2$

$= \frac{10 \times 9}{8 \times 1} = 45 (0.6)^8 (0.4)^2.$

$= 0.12$

$P(E|Z_B) = {}^{10}C_8 (0.5)^8 (0.5)^2$

$= 45 \times 11 \times 11 = 0.04.$

(3)

$$P(Z_A|E) = \frac{0.12}{0.12 + 0.04}$$

$$= 0.75$$

$$P(Z_B|E) = \frac{0.04}{0.12 + 0.04}$$

$$= 0.25$$

$$P(E|Z_A) = {}^{10}C_4 (0.6)^4 (0.4)^6$$

$$= \frac{10 \times 9 \times 8 \times 7}{4 \times 3 \times 2} = 210 (0.6)^4 (0.4)^6$$

$$= 0.11.$$

$$P(E|Z_B) = {}^{10}C_4 (0.5)^4 (0.5)^6.$$

$$= 210 (0.5)^{10}.$$

$$= 0.20$$

(4) $$P(Z_A|E) = \frac{0.11}{0.11 + 0.20} = 0.35$$

$$P(Z_B|E) = \frac{0.20}{0.11 + 0.20} = 0.64.$$

$P(E|Z_A) = {}^{10}C_7 (0.6)^7 (0.4)^3$

$= \frac{10 \times 9 \times 8}{3 \times 2} (0.6)^7 (0.4)^3$

$= 120 (0.6)^7 (0.4)^3$

$= 0.21$

$P(E|Z_B) = {}^{10}C_7 (0.5)^7 (0.5)^3$

$= 120 (0.5)^{10}$

$= 0.11$

3) $P(Z_A|E) = \frac{0.21}{0.11 + 0.21}$

$= 0.65$

$P(Z_B|E) = \frac{0.11}{0.11 + 0.21}$

$= 0.35$

$5 \times 0.45 = 2.2H$ $5 \times 0.45 = 2.2T$

$5 \times 0.55 = 2.8H$ $5 \times 0.55 = 2.8T$

∴

2.2H	2.2T	2.8H	2.8T
7.2H	0.8T	1.8H	1.2T
5.9H	1.7T	5.9H	2.1H 0.5T
1.4H	2.1T	2.8H	3.9T
4.6H	1.8T	2.5H	1.1T
21.3H	8.6T	11.7H	8.4T