Evaluating the EM Algorithm

Computer-Intensive in Statistics Project III Report

Scott Williams

1 Introduction

1.1 Goal of the Project

A mixture model problem of the form

$$\pi_1 * N(\mu_1, \sigma_1^2) + \pi_2 * N(\mu_2, \sigma_2^2)$$

is given. π_1 and π_2 are values between 0 and 1 that add up to 1 that represent the proportion of samples that come from the attached distribution. N(μ_1, σ_1^2) and N(μ_2, σ_2^2) are the two different normal distributions the samples come from with mean μ_i and variance σ_i^2. In this study, the two proportions, π_1 and π_2, are to be found using the EM algorithm. The EM algorithm finds the maximum likelihood estimator for the parameter of interest. The algorithm consists of two steps, the E step standing for expectation and the M step standing for maximization. We want to evaluate how well the EM algorithm works on this mixture model problem.

2 Approach

2.1 Programming Environment & Preparation

To perform the EM algorithm, we will be using a Python 3 environment. The necessary dependencies for the code are the following Python libraries: math and numpy. Before any programming, the parameters of the distributions and the proportion parameter π must be chosen. Observe that for the mixture model, π_1 and π_2 can be expressed in terms of one parameter. Let $\pi = \pi_1$. Since π tells us the proportion of samples coming from the first distribution, then the remaining samples from the second distribution will have the proportion $1-\pi$. The mean and variance for both distributions were chosen arbitrarily trying to avoid anything too simple that could be considered a toy problem. Here we have chosen $\mu_1 = 4$, $\sigma_1^2 = 2$, $\mu_2 = 1.7$, and $\sigma_2^2 = 5$. The choice of π will vary for the purposes of evaluating the EM algorithm.

2.2 EM Algorithm for Mixture Model

The following process is used to find the maximum likelihood estimator for π in the mixture model problem:

1. Give an initial value of $\pi = \pi_0$.

2. E step: Calculate

$$z_j^{(k)} = \frac{\pi^{(k)} * f_1(y_j)}{\pi^{(k)} * f_1(y_j) + (1 - \pi^{(k)}) * f_2(y_j)} \; \forall \; j = 1, 2, 3, ..., n$$

   where the superscript k indicates the k^{th} iteration of the EM algorithm, f_1 and f_2 indicate the normal distribution's pdf:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right\}$$

   with the parameters of the first and second normal distributions specified earlier in section 2.1 respectively, and y_j meaning the j^{th} sample from the mixture samples.

3. M step: Calculate

$$\pi^{(k+1)} = \frac{\sum_{j=1}^{n} z_j^{(k)}}{n}$$

   where n is the total amount of elements in the z array.

4. Repeat E step and M step until convergence. Convergence means that the error between $\pi^{(k+1)}$ and $\pi^{(k)}$ is less than or equal to a threshold we will set to know the stopping point of the EM algorithm.

2.3 Implementation in Python

Here is how the process detailed in section 2.2 was implemented in Python:

1. Choose the true value of the mixing proportion π to evaluate the EM algorithm's performance.

2. Generate mixture samples. Here, 10,000 mixture samples were generated.

3. Set the threshold for the error so a convergence check can be done. Here, 10^{-15} was used.

4. Provide an arbitrary error value greater than the threshold so that the EM loop can start. This error value will later be updated to reflect the calculated error between iterations of the estimates of π.

5. Give an initial value of π to start the algorithm. Here, $\pi_0 = 0.5$ was used.

6. In a while loop in which the condition is while error $>$ threshold, perform steps 2, 3, and 4 detailed in section 2.2.

7. To perform step 4 in section 2.2, calculate the relative error between $\pi^{(k+1)}$ and $\pi^{(k)}$.

$$Relative\ Error\ =\ \frac{\left|\pi^{(k+1)} - \pi^{(k)}\right|}{\pi^{(k)}}$$

8. In an if statement, check if the error is greater than the threshold.

9. If so, update $\pi^{(k)} = \pi^{(k+1)}$ and the loop will continue.

10. If not, the maximum likelihood estimator for π has been found to be $\pi^{(k+1)}$ and the loop will end.

3 Performing the EM Algorithm on Mixing Proportions

3.1 One Mixing Proportion

We first focus on one mixing proportion to test whether the EM algorithm code works as expected. In this first attempt, the distribution parameters are $\mu_1 = 4$, $\sigma_1^2 = 2$, $\mu_2 = 1.7$, and $\sigma_2^2 = 5$. For this test, the mixing proportion parameter $\pi = 0.286$. Recall that $\pi_1 = \pi$ and $\pi_2 = (1 - \pi)$. With the choice of $\pi = 0.286$, $\pi_1 = 0.286$ and $\pi_2 = 0.714$. After running the code with these parameters, the maximum likelihood estimator found for π results in $\hat{\pi_1} = 0.277$ and $\hat{\pi_2} = 0.723$. The hat indicates that it is the result from the EM algorithm. The EM algorithm provides quite accurate estimations for the mixing proportions in this specific case. Calculating the relative error between the true values and the estimations for the mixing proportions, we have only 3.14% error for π_1 and 1.26% error for π_2. These results are encouraging, however, we want to evaluate how well the EM algorithm works for the mixture model problem more robustly. It is possible that we got lucky with this choice of π and that this was an ideal case for the algorithm.

3.2 Multiple Mixing Proportions

We generalize our code by turning it into a function titled EM. The EM function takes μ_1, μ_2, σ_1^2, σ_2^2, π, and the threshold as arguments. Now multiple cases can be tested by passing any mixing proportion into the function. Taking advantage of this the numpy function linspace is used to create an array of 50 different values of π ranging from 0 to 1. Keeping the same normal distribution parameters, we pass these 50 different values of π to the EM function in a for loop to estimate all the different mixing proportions and measure the error. Note that first and last cases of the true values of the mixing proportions will have π_1 and π_2 equal to 0 respectively. To avoid any problems with division by zero when calculating the relative error we instead use the absolute error for these cases.

$$absolute\ error\ =\ |true - estimate|$$

Otherwise, the relative error is calculated as before.

Before summarizing results, it should be noted that Python was formatted to only display 3 decimal places for readability. After using the EM algorithm function on all of these 50 different mixing proportions we obtain overwhelmingly accurate results. Most cases returned results with very low error. Out of the 50 cases, only 2 seemed to challenge the EM algorithm. In the following explanation of results, each number represents $\hat{\pi_1}$ and $\hat{\pi_2}$, the true values of π_1 and π_2, and the error for π_1 and π_2. The first case that led to error noticeably larger than the others was when the true mixing proportions were 0.041 and 0.959. The EM algorithm estimated the mixing proportions to be 0.050 and 0.950. The relative error between the estimates and the true values was 0.214 and 0.009. The second case that led to error noticeably larger than the others was when the true mixing proportions were 0.082 and 0.918. The EM algorithm estimated the mixing proportions to be 0.092 and 0.908. The relative error between the estimates and the true values was 0.129 and 0.011. All code and output will be in the section titled Figures and Python Code. The full list of estimates vs true values as well as the error for each case will be shown in that section.

4 Conclusion

4.1 Final Thoughts

The EM algorithm works very well. The algorithm accurately found the maximum likelihood estimator for π given many different mixing proportions. For some cases, the algorithm even returned the exact mixing proportion up to 3 decimal places with no error. All different mixture scenarios were tested. Cases included mixture samples coming all from one of the distributions, mixture samples being very close to evenly split between the two distributions, and mixture samples being unevenly split between the two distributions. Only 2 out of the 50 cases in our evaluation of the EM algorithm resulted in estimations that had noticeable error. However, even in these two cases, the estimations of π_1 and π_2 were still close to the true values of π_1 and π_2.

5 Figures and Python Code

Importing Dependencies

```
# Importing necessary libraries
import math
import numpy as np
```

Defining distributions and simulating mixture samples

```
# Defining distributions
f1 = lambda x: 1/math.sqrt(2*math.pi*2) * math.exp(-((x - 4)**2)/(2*2))
f2 = lambda x: 1/math.sqrt(2*math.pi*5) * math.exp(-((x - 1.7)**2)/(2*5))

# Generating 10,000 values from the mixture model: 0.35 * f1 + 0.65 * f2
pi = .286
mix = []
for i in range(10000):
    if(i < pi*10000):
        mix.append(np.random.normal(4.0, math.sqrt(2)))
    else:
        mix.append(np.random.normal(1.7, math.sqrt(5)))
```

EM Algorithm for one mixing proportion

```
# EM Algorithm
err = 10
threshold = 10**(-15)

# Give an initial value of pi
pi_0 = 0.5

while err > threshold:
    # E Step
    z = []
    for i in range(len(mix)):
        value = (pi_0 * f1(mix[i]))/(pi_0*f1(mix[i]) + (1-pi_0)*f2(mix[i]))
        z.append(value)

    # M Step
    pi_new = sum(z)/len(z)

    # Calculate error to check for convergence
    err = abs(pi_new - pi_0)/pi_0

    if(err > threshold):
        pi_0 = pi_new
    else:
        pi_EM = pi_new
        print("Estimations:" + str(("{:.3f}".format(pi_EM), "{:.3f}".format(1-pi_EM))) + " vs True:" + str((pi,(1-pi))))
```

```
Estimations:('0.277', '0.723') vs True:(0.286, 0.714)
```

Generalizing the example for any mixing proportion

```
def EM(u1, u2, var1, var2, pi, threshold):
    # Defining distributions
    f1 = lambda x: 1/math.sqrt(2*math.pi*var1) * math.exp(-((x - u1)**2)/(2*var1))
    f2 = lambda x: 1/math.sqrt(2*math.pi*var2) * math.exp(-((x - u2)**2)/(2*var2))

    mix = []
    for i in range(10000):
        if(i < pi*10000):
            mix.append(np.random.normal(u1, math.sqrt(var1)))
        else:
            mix.append(np.random.normal(u2, math.sqrt(var2)))

    # EM Algorithm
    err = 10

    # Give an initial value of pi
    pi_0 = 0.5

    while err > threshold:
        # E Step
        z = []
        for i in range(len(mix)):
            value = (pi_0 * f1(mix[i]))/(pi_0*f1(mix[i]) + (1-pi_0)*f2(mix[i]))
            z.append(value)

        # M Step
        pi_new = sum(z)/len(z)

        # Calculate relative error to check for convergence
        err = abs(pi_new - pi_0)/pi_0
```

```
        if(err > threshold):
            pi_0 = pi_new
        else:
            pi_EM = pi_new
            print("Estimations:" + str(("{:.3f}".format(pi_EM), "{:.3f}".format(1-pi_EM))) + " vs True:" + str(("{:.3f}".for

            if(pi == 0 or (1-pi) == 0):
                # Calculating the absolute error between the estimates and the true values
                error = ("{:.3f}".format(abs(pi - pi_EM)), "{:.3f}".format(abs((1 - pi) - (1 - pi_EM))))
                print("To avoid division by zero, the absolute error between the estimates and the true values is: " + str(e
            else:
                # Calculating the relative error between the estimates and the true values
                error = ("{:.3f}".format(abs(pi - pi_EM)/pi), "{:.3f}".format(abs((1 - pi) - (1 - pi_EM))/(1-pi)))
                print("The relative error between the estimates and the true values is: " + str(error))
```

```
EM(4, 1.7, 2, 5, 0.286, 10**(-15))
```

```
Estimations:('0.292', '0.708') vs True:('0.286', '0.714')
The relative error between the estimates and the true values is: ('0.022', '0.009')
```

Testing on multiple mixing proportions

```
pi_mult = np.linspace(0, 1, 50, True)
```

```
for i in range(len(pi_mult)):
    # Calculating mixing proportions
    EM(4, 1.7, 2, 5, pi_mult[i], 10**(-15))
```

```
Estimations:('0.000', '1.000') vs True:('0.000', '1.000')
To avoid division by zero, the absolute error between the estimates and the true values is: ('0.000', '0.000')
Estimations:('0.021', '0.979') vs True:('0.020', '0.980')
The relative error between the estimates and the true values is: ('0.007', '0.000')
Estimations:('0.050', '0.950') vs True:('0.041', '0.959')
The relative error between the estimates and the true values is: ('0.214', '0.009')
Estimations:('0.061', '0.939') vs True:('0.061', '0.939')
The relative error between the estimates and the true values is: ('0.004', '0.000')
Estimations:('0.092', '0.908') vs True:('0.082', '0.918')
The relative error between the estimates and the true values is: ('0.129', '0.011')
Estimations:('0.110', '0.890') vs True:('0.102', '0.898')
The relative error between the estimates and the true values is: ('0.077', '0.009')
Estimations:('0.124', '0.876') vs True:('0.122', '0.878')
The relative error between the estimates and the true values is: ('0.014', '0.002')
Estimations:('0.146', '0.854') vs True:('0.143', '0.857')
The relative error between the estimates and the true values is: ('0.022', '0.004')
Estimations:('0.170', '0.830') vs True:('0.163', '0.837')
The relative error between the estimates and the true values is: ('0.044', '0.009')
Estimations:('0.184', '0.816') vs True:('0.184', '0.816')
The relative error between the estimates and the true values is: ('0.003', '0.001')
Estimations:('0.204', '0.796') vs True:('0.204', '0.796')
The relative error between the estimates and the true values is: ('0.001', '0.000')
Estimations:('0.220', '0.780') vs True:('0.224', '0.776')
The relative error between the estimates and the true values is: ('0.022', '0.006')
Estimations:('0.250', '0.750') vs True:('0.245', '0.755')
The relative error between the estimates and the true values is: ('0.023', '0.007')
Estimations:('0.265', '0.735') vs True:('0.265', '0.735')
The relative error between the estimates and the true values is: ('0.000', '0.000')
Estimations:('0.283', '0.717') vs True:('0.286', '0.714')
```

```
The relative error between the estimates and the true values is: ('0.010', '0.004')
Estimations:('0.317', '0.683') vs True:('0.306', '0.694')
The relative error between the estimates and the true values is: ('0.035', '0.016')
Estimations:('0.341', '0.659') vs True:('0.327', '0.673')
The relative error between the estimates and the true values is: ('0.044', '0.021')
Estimations:('0.341', '0.659') vs True:('0.347', '0.653')
The relative error between the estimates and the true values is: ('0.017', '0.009')
Estimations:('0.382', '0.618') vs True:('0.367', '0.633')
The relative error between the estimates and the true values is: ('0.040', '0.023')
Estimations:('0.396', '0.604') vs True:('0.388', '0.612')
The relative error between the estimates and the true values is: ('0.022', '0.014')
Estimations:('0.412', '0.588') vs True:('0.408', '0.592')
The relative error between the estimates and the true values is: ('0.009', '0.006')
Estimations:('0.419', '0.581') vs True:('0.429', '0.571')
The relative error between the estimates and the true values is: ('0.023', '0.017')
Estimations:('0.454', '0.546') vs True:('0.449', '0.551')
The relative error between the estimates and the true values is: ('0.011', '0.009')
Estimations:('0.468', '0.532') vs True:('0.469', '0.531')
The relative error between the estimates and the true values is: ('0.003', '0.003')
Estimations:('0.489', '0.511') vs True:('0.490', '0.510')
The relative error between the estimates and the true values is: ('0.002', '0.002')
Estimations:('0.522', '0.478') vs True:('0.510', '0.490')
The relative error between the estimates and the true values is: ('0.022', '0.023')
Estimations:('0.533', '0.467') vs True:('0.531', '0.469')
The relative error between the estimates and the true values is: ('0.004', '0.004')
Estimations:('0.555', '0.445') vs True:('0.551', '0.449')
The relative error between the estimates and the true values is: ('0.008', '0.010')
Estimations:('0.560', '0.440') vs True:('0.571', '0.429')
The relative error between the estimates and the true values is: ('0.021', '0.028')
Estimations:('0.573', '0.427') vs True:('0.592', '0.408')
The relative error between the estimates and the true values is: ('0.031', '0.046')
Estimations:('0.612', '0.388') vs True:('0.612', '0.388')
The relative error between the estimates and the true values is: ('0.001', '0.002')
Estimations:('0.621', '0.379') vs True:('0.633', '0.367')
The relative error between the estimates and the true values is: ('0.018', '0.032')
Estimations:('0.642', '0.358') vs True:('0.653', '0.347')
The relative error between the estimates and the true values is: ('0.017', '0.032')
Estimations:('0.688', '0.312') vs True:('0.673', '0.327')
The relative error between the estimates and the true values is: ('0.022', '0.045')
Estimations:('0.689', '0.311') vs True:('0.694', '0.306')
The relative error between the estimates and the true values is: ('0.007', '0.016')
Estimations:('0.714', '0.286') vs True:('0.714', '0.286')
The relative error between the estimates and the true values is: ('0.000', '0.000')



Estimations:('0.731', '0.269') vs True:('0.735', '0.265')
The relative error between the estimates and the true values is: ('0.005', '0.014')
Estimations:('0.753', '0.247') vs True:('0.755', '0.245')
The relative error between the estimates and the true values is: ('0.002', '0.007')
Estimations:('0.767', '0.233') vs True:('0.776', '0.224')
The relative error between the estimates and the true values is: ('0.011', '0.037')
Estimations:('0.800', '0.200') vs True:('0.796', '0.204')
The relative error between the estimates and the true values is: ('0.005', '0.018')
Estimations:('0.817', '0.183') vs True:('0.816', '0.184')
The relative error between the estimates and the true values is: ('0.000', '0.002')
Estimations:('0.838', '0.162') vs True:('0.837', '0.163')
The relative error between the estimates and the true values is: ('0.001', '0.005')
Estimations:('0.855', '0.145') vs True:('0.857', '0.143')
The relative error between the estimates and the true values is: ('0.002', '0.014')
Estimations:('0.873', '0.127') vs True:('0.878', '0.122')
The relative error between the estimates and the true values is: ('0.005', '0.033')
Estimations:('0.904', '0.096') vs True:('0.898', '0.102')
The relative error between the estimates and the true values is: ('0.007', '0.059')
Estimations:('0.917', '0.083') vs True:('0.918', '0.082')
The relative error between the estimates and the true values is: ('0.002', '0.023')
Estimations:('0.944', '0.056') vs True:('0.939', '0.061')
The relative error between the estimates and the true values is: ('0.006', '0.086')
Estimations:('0.958', '0.042') vs True:('0.959', '0.041')
The relative error between the estimates and the true values is: ('0.001', '0.029')
Estimations:('0.978', '0.022') vs True:('0.980', '0.020')
The relative error between the estimates and the true values is: ('0.001', '0.070')
Estimations:('0.999', '0.001') vs True:('1.000', '0.000')
To avoid division by zero, the absolute error between the estimates and the true values is: ('0.001', '0.001')
```