Photo by Marvin Ronsdorf on Unsplash

Expectation Maximization Explained

A versatile algorithm for clustering, NLP, and more

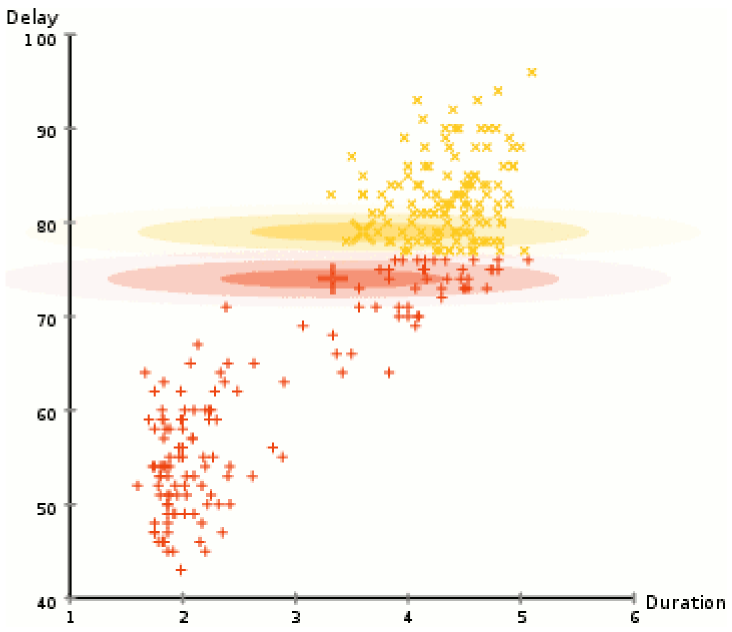
Ravi Charan Jul 12, 2020 · 11 min read ★

Expectation Maximization (EM) is a classic algorithm developed in the 60s and 70s with diverse applications. It can be used as an unsupervised clustering algorithm and procedure, it is an alternative to gradient descent and the like with the major advantage that, in many circumstances, the updates can be computed analytically. More than that, it is a flexible framework for thinking about optimization.

In this article, we'll start with a simple clustering example and then discuss the algorithm in generality.

Unsupervised Clustering

Consider the situation where you have a variety of data points with some measurements of them. We wish to assign them to different groups.



Expectation Maximization for Old Faithful Eruption Data (Wikipedia)

In the example here we have data on eruptions of the iconic Old Faithful geyser in Yellowstone. For each eruption, we have measured its length and the time since the previous eruption. We might suppose that there are two "types" of eruptions (red and yellow in the diagram), and for each type of eruption the resulting data is generated by a (multivariate) Normal distribution. This is called a Gaussian Mixture Model, incidentally.

Similar to k-means clustering, we start with a random guess for what those two distributions/clusters are and then proceed to improve iteratively by alternating two steps:

respectively.

2. (**Maximization**) Update the parameters for each cluster (weighted mean location and variance-covariance matrix) based on the points in the cluster (weighted by their probability assigned in the first step).

Note that unlike in k-means clustering, our model is *generative*: it purports to tell us the process by which the data was generated. And we could, in turn, re-sample the model to generate more (fake) data.

Got it? Now we are going to do a 1-dimensional example with equations.

Consider data points with a single measurement x. We suppose these are generated by 2 clusters each following a normal distribution N(μ, σ^2). The probability of data being generated by the first cluster is π.

So we have 5 parameters: a mixing probability π, and a mean μ and standard deviation σ for each cluster. I will denote them collectively as θ.

$$\theta := \pi, \mu_1, \sigma_1, \mu_2, \sigma_2$$

The 5 parameters for our model, represented collectively as θ

What is the probability of observing a datapoint with value x? Let the normal distribution's probability-density function be denoted by ϕ. To keep notation less cluttered, I will use the standard deviation as a parameter instead of the usual variance.

$$p(x;\theta) = \pi\phi(x;\mu_1,\sigma_1) + (1-\pi)\phi(x;\mu_2,\sigma_2)$$

Probability of observing a point with value x

The probability (likelihood) of observing our entire dataset of *n* points is:

$$L(\theta;\text{data}) = \prod^{n} p(x_i;\theta)$$

Likelihood of observing our whole dataset

And we typically choose to take the logarithm of this to turn our product into a more manageable sum, the log-likelihood.

$$\begin{aligned}\ell(\theta; \text{data}) &= \sum_{i=1}^{n} \log p(x_i; \theta) \\ &= \sum_{i=1}^{n} \log \left[\pi \phi(x_i; \mu_1, \sigma_1) + (1-\pi)\phi(x_i; \mu_2, \sigma_2)\right]\end{aligned}$$

Log-likelihood of observing our data

Our goal is to maximize this: we want our parameters to be the ones where it is most likely that we observe the data we observed (a Maximum Likelihood Estimator).

Now the question is, how are we going to optimize it? Doing so directly and analytically would be tricky because of the sum in the logarithm.

The trick is to imagine there is a **latent variable** which we will call Δ. It is a binary (0/1-valued) variable that determines whether a point is in cluster 1 or cluster 2. If we know Δ for each point, it would be very easy to calculate the maximum-likelihood estimates of our parameters. For convenience to match our choice of Δ being 1 for the second cluster, we will switch π to be the probability of a point being in the second cluster.

$$\begin{aligned}\ell(\theta; \text{data}, \Delta) = &\sum_{\text{data}} \left[(1-\Delta_i)\log\phi(x_i; \mu_1, \sigma_1) + \Delta_i \log\phi(x_i; \mu_2, \sigma_2)\right] + \\ &\sum_{\text{data}} \left[(1-\Delta_i)\log(1-\pi) + \Delta_i \log \pi\right]\end{aligned}$$

Notice that the sums are now outside the logarithm. Also, we pick up an extra sum to account for the likelihood of observing each Δ.

standard deviation (the population formula, which is the MLE). For π, the sample proportion of points in the second cluster. These are the usual maximum likelihood estimators for each parameter.

Of course, we didn't observe Δ. The solution to this is the heart of the Expectation-Maximization algorithm. Our plan is:

1. Start with an arbitrary initial choice of parameters.
2. (**Expectation**) Form an estimate of Δ.
3. (**Maximization**) Compute the maximum-likelihood estimators to update our parameter estimate.
4. Repeat steps 2 and 3 to convergence.

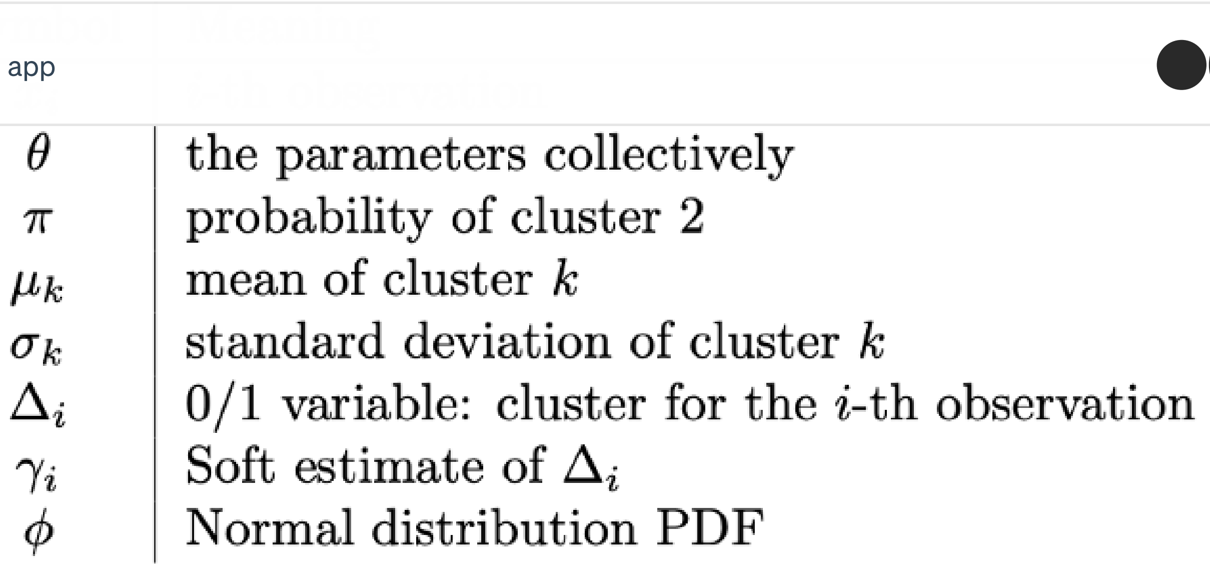
Again, you may find it helpful to think about k-means clustering, where we do the same thing. In k-means clustering, we assign each point to the closest centroid (expectation step). In essence, this is a hard estimate of Δ. Hard because it is 1 for one of the clusters and 0 for all the others. Then we update the centroids to be the mean of the points in the cluster (maximization step). This is the maximum-likelihood estimator for μ. In k-means clustering, the "model" for the data doesn't have a standard deviation. (The "model" is in scare quotes because it isn't generative).

In our example, we will instead do a *soft* assignment of Δ. We sometimes call this the *responsibility* (how responsible is each cluster for each observation). We will denote the responsibility as ɣ.

$$\begin{aligned}\gamma_i(\theta) &= \mathbb{E}\left[\Delta_i \middle| \theta, \text{data}\right] \\ &= \text{Prob}\left(\Delta_i = 1 \middle| \theta, \text{data}\right)\end{aligned}$$

Responsibility ɣ of each cluster for data point i

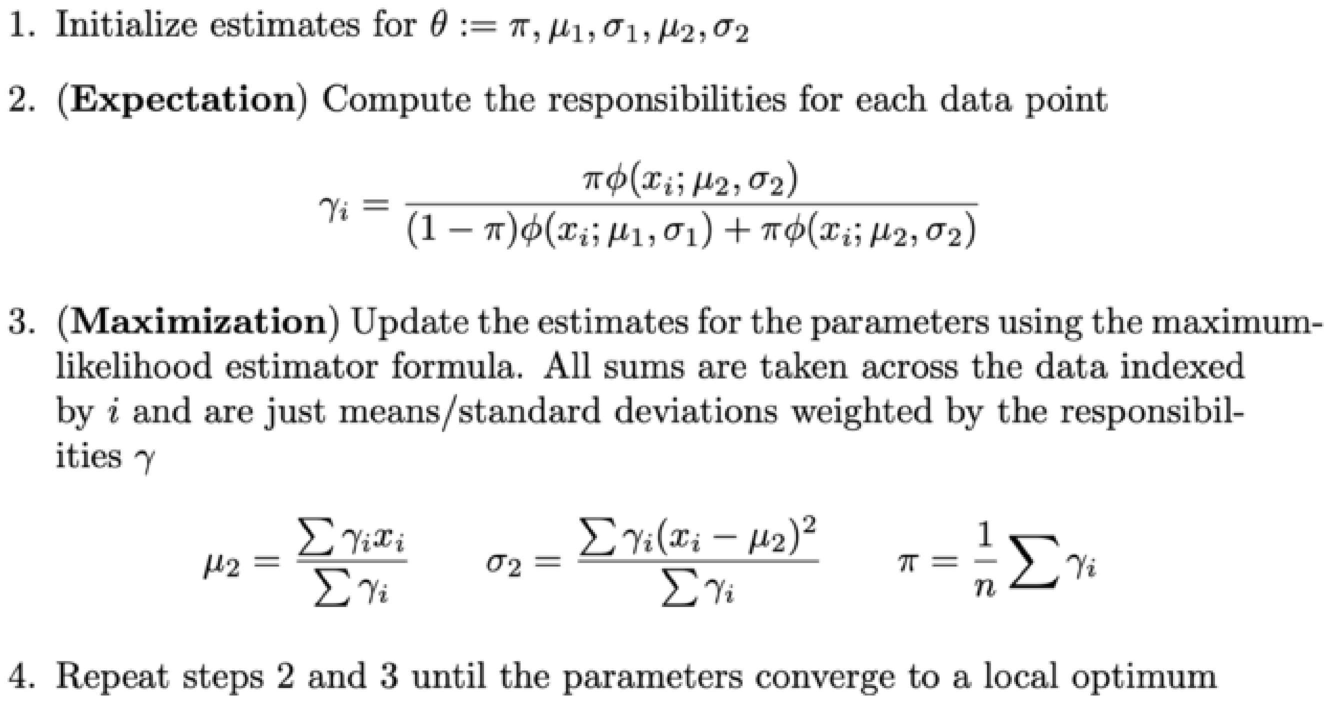
Now we can write down the full algorithm for this example. But before we do so, we will quickly review a table of symbols we've defined (there were a lot).



Symbol	Meaning
x_i	i-th observation
θ	the parameters collectively
π	probability of cluster 2
μ_k	mean of cluster k
σ_k	standard deviation of cluster k
Δ_i	0/1 variable: cluster for the i-th observation
γ_i	Soft estimate of Δ_i
ϕ	Normal distribution PDF

Table of Symbols

And here is the algorithm:



1. Initialize estimates for $\theta := \pi, \mu_1, \sigma_1, \mu_2, \sigma_2$
2. (**Expectation**) Compute the responsibilities for each data point

$$\gamma_i = \frac{\pi\phi(x_i; \mu_2, \sigma_2)}{(1-\pi)\phi(x_i; \mu_1, \sigma_1) + \pi\phi(x_i; \mu_2, \sigma_2)}$$

3. (**Maximization**) Update the estimates for the parameters using the maximum-likelihood estimator formula. All sums are taken across the data indexed by i and are just means/standard deviations weighted by the responsibilities γ

$$\mu_2 = \frac{\sum \gamma_i x_i}{\sum \gamma_i} \qquad \sigma_2 = \frac{\sum \gamma_i (x_i - \mu_2)^2}{\sum \gamma_i} \qquad \pi = \frac{1}{n}\sum \gamma_i$$

4. Repeat steps 2 and 3 until the parameters converge to a local optimum

The Expectation Maximization algorithm for our example

Note that the estimates of μ and σ for cluster 1 are analogous but using 1–ɣ as the weights instead.

Now that we have given an example of the algorithm, you hopefully have a feel for it. We'll move on to discussing the algorithm in general. This basically amounts to dressing up everything we did with slightly more complicated variables. And it will put us in a position to explain why it works.

1. We have some data X of whatever form.
2. We posit there is also unobserved (latent) data Δ, again of whatever form.
3. We have a model with parameters θ.
4. We have the ability to compute the log-likelihood $\ell(\theta; X, \Delta)$. Specifically, the log of the probability of observing our data *and* specified assignments of the latent variables given the parameters.
5. We also have the ability to use the model to compute the conditional distribution Δ|X given a set of parameters. We will denote this P(Δ|X; θ).
6. Consequently we can compute the log-likelihood $\ell(\theta; X)$. This is the log of the probability of observing our data given the parameters (without assuming an assignment for the latent variables).

Using P to denote the probability, we can now use the chain rule to write:

$$P(\Delta, X|\theta) = P(\Delta|X, \theta) \times P(X|\theta)$$

The Chain Rule for Probability

The notation can be subtle here. All three terms take the parameters θ as a given.

1. The first term on the left is the probability of observing the data and a specified latent variable assignment.
2. The first term on the right hand side is the probability of the specified assignment of the latent variables given the observed data.
3. The last term is the probability of observing the data.

We can take logarithms and rearrange terms. Then in the second line we will make a notation change (and a confusing one at that. Don't blame me, I didn't invent it):

$$\log P(X|\theta) = \log P(\Delta, X|\theta) - \log P(\Delta|X, \theta)$$

For the first two terms, it's worth reviewing what they are in the context of our previous example. The first, ℓ(θ; X), is the one we want to optimize. The second, ℓ(θ; X, Δ), was the one that was analytically tractable.

$$\ell(\theta; X) = \sum_{i=1}^{n} \log\left[\pi\phi(x_i; \mu_1, \sigma_1) + (1-\pi)\phi(x_i; \mu_2, \sigma_2)\right]$$

$$\ell(\theta; X, \Delta) = \sum_{i=1}^{n} [(1-\Delta_i)\log\phi(x_i; \mu_1, \sigma_1) + \Delta_i \log\phi(x_i; \mu_2, \sigma_2) + (1-\Delta_i)\log(1-\pi) + \Delta_i \log \pi]$$

Likelihood Formulas from the Gaussian Mixture Model example

Now, remember that I said we can compute the conditional distribution Δ|X given the parameters θ? This is where things get wild.

We are going to introduce a *second* set of the same parameters, call it θ′. I will also sometimes denote it with a hat (circumflex) over it, like the one this "ê" has.[2] Think of this set of parameters as our *current* estimate. The θ currently in our formulas will be optimized to improve our estimate.

Now we are going to take the expectation of the log likelihoods with respect to the conditional distribution Δ|X, θ′ – namely, the distribution of our latent variables given the data and our current parameter estimate.

$$\ell(\theta, X) = \mathbb{E}\left[\ell(\theta; X, \Delta)|X, \hat{\theta}\right] - \mathbb{E}\left[\ell(\theta; \Delta|X)|X, \hat{\theta}\right]$$

The term on the left hand side doesn't change since it doesn't know/care about Δ anyways (it's a constant). Again, the expectation is over the possible values of Δ. If you are following along with respect to our example, the term ℓ(θ; X, Δ) changes after we take the expectation so that Δ is replaced by ɣ.

n

$$+ (1-\gamma_i)\log(1-\pi) + \gamma_i \log \pi]$$

$$\gamma_i = \mathbb{E}\left[\Delta_i | X, \hat{\theta}\right] = P(\Delta_i = 1 | X, \hat{\theta})$$

Expectation of the Likelihood in the Gaussian Mixture Model example.

Now, very quickly, to ameliorate the notational nightmare we've got going on here, let's introduce shorthand notation for the two expectations we have on the right hand side

$$\begin{aligned} \ell(\theta, X) &= \mathbb{E}\left[\ell(\theta; X, \Delta) | X, \hat{\theta}\right] - \mathbb{E}\left[\ell(\theta; \Delta | X) | X, \hat{\theta}\right] \\ &\equiv \qquad Q(\theta, \hat{\theta}) \qquad - \qquad R(\theta, \hat{\theta}) \end{aligned}$$

Shorthand Notations for the Expected Likelihoods

The algorithm becomes:

1. Initialize a guess for the parameters, call it $\hat{\theta}_0$
2. (**Expectation**) On step j, compute

   $$Q(\theta, \hat{\theta}_j) = \mathbb{E}\left[\ell(\theta; X, \Delta) | X, \hat{\theta}_j\right]$$

   regarded as a function of θ
3. (**Maximization**) Maximize $Q(\theta, \hat{\theta}_j)$ with regards to θ and call the result $\hat{\theta}_{j+1}$, the new estimate.
4. Loop through steps 2 and 3 until convergence

The General Expectation–Maximization Algorithm

Why it Works

The heavy lifting to prove this will work is to consider the function R(θ, θ′). The claim is that R is maximized when θ=θ′. In lieu of a full proof let's think about what R computes. Stripping away the dependence on the data X (which is shared between the

$$R(\theta, \hat{\theta}) = \mathbb{E}_{\Delta|\hat{\theta}}\left[\log P(\Delta|\theta)\right]$$

Schematic form of the function R

In other words, we have two probability distributions. We use one (parameterized by θ′) to generate data Δ and we use the other (parameterized by θ) to compute the probability of what we saw. If Δ represents just a single number and the distributions have probability-density functions, we could write (again, schematically)

$$R(q||p) = \int p(x) \log q(x) dx$$

Suggestive schematic form for the function R in a special case

I have suggestively written this in a form similar to that of the Kullback-Leibler (KL) Divergence which is (almost) a measure of the distance between two probability distributions. If we subtract R(q||p) from a constant R(p||p) we will get the KL-divergence which is bounded below at 0 and is only 0 when q=p. (The only thing a distance[3] 0 from the distribution p is p itself). In other words, R is maximized when q=p. This is a standard result about the KL-divergence and may be proved with Jensen's inequality.[4]

Now the only thing left to do is consider the difference between the likelihoods before and after our update step:

$$\ell(\hat{\theta}_{j+1}; X) - \ell(\hat{\theta}_j; X) = \underbrace{\left[Q(\hat{\theta}_{j+1}, \hat{\theta}_j) - Q(\hat{\theta}_j, \hat{\theta}_j)\right]}_{>0} - \underbrace{\left[R(\hat{\theta}_{j+1}, \hat{\theta}_j) - R(\hat{\theta}_j, \hat{\theta}_j)\right]}_{<0}$$

Improvement in likelihood after an update step

We chose the new parameters to maximize Q, so the first term is definitely positive. By the argument above, R is maximized by taking the old parameters as its first argument, better.

Notice also that we don't have to optimize Q. All we have to is find some way to make it better and our updates are still guaranteed to make things better.

Conclusion

Hopefully, you now have a good feel for the algorithm. In terms of the mathematics, the key equation is just the likelihoods below. After that, we just take expectations with respect to the old parameters (the expectation step) and show that we're fine to just optimize the first term on the right hand side. As we motivated with the Gaussian Mixture Model example, this second term is often easier to optimize. The third term we don't have to worry about, it won't mess anything up.

$$\begin{aligned} \log P(X|\theta) &= \log P(\Delta, X|\theta) - \log P(\Delta|X,\theta) \\ \ell(\theta;X) &= \ell(\theta;X,\Delta) \qquad - \ell(\theta;\Delta|X) \end{aligned}$$

Stepping back a bit, I want to emphasize the power and usefulness of the EM algorithm. First of all, it represents the idea that we can introduce latent variables and then compute by alternately dealing with the latent variables (taking the parameters as fixed and known) and dealing with the parameters (taking the latent variables as fixed and known). This is a powerful idea that you'll see in a variety of contexts.

Second, the algorithm is inherently fast because it doesn't depend on computing gradients. Anytime you can solve a model analytically (like using a linear regression), it's going to be faster. And this lets us take analytically intractable problems and solve parts of them analytically, extending that power to an iterative context.

Finally, I want to note that there is plenty more to say about the EM algorithm. It generalizes to other forms of doing the maximization step and to variational Bayesian techniques and can be understood in different ways (for example as maximization-maximization or as alternating projections to a submanifold under mutually dual affine connections on a statistical manifold (the e- and m- connections)). More on that in the future!

Acknowledgments

This discussion largely follows that in the Elements of Statistical Learning, though a bit more deliberate pace. Special thanks to Fangfang Lee for informing me about this wonderful algorithm.

Notes

[1] Latent Dirichlet Allocation is often fit with variational Bayes methods, an extension of Expectation Maximization. See the sklearn implementation, for example.

[2] Impossible to type a hat over a θ on this platform. Come on Medium please give us LaTeX.

[3] I did not put distance in quotes because I am not referring to the Kullback-Leibler divergence (which is not a metric) but to the Fisher information metric (which is). The two are related in a deep way; for us we can just say that the KL-divergence is only 0 when the actual distance is 0.

[4] The standard proof of this seems to be a handwavy appeal to Jensen's inequality. My version is also handwavy but incorporates Jensen's through the KL-divergence instead.

Towards Data Science Mathematics Statistics Machine Learning Data Science