Tech Review: Expectation Maximization

Steven Ma

November 14 2020

Introduction

Expectation maximization is an algorithmic approach less often heard of than its derivations, but an underlying core to unsupervised machine learning models. The goal of EM, as it is often called, is to find parameters to maximize the likelihood of the data. In this tech review, we will be discussing expectation maximization's motivation, core theorems, and its applications in text information systems. Most of the information in this paper can be attributed to coursework at the University of Illinois at Urbana-Champaign the writer has learned; specifically, Professor Sanmi Koyejo's Machine Learning course.

A Crash Course of Expectation Maximization

Oftentimes when observing a dataset, we assume that all the variables that factor into our model estimation are known. However, it is often the case that there are variables that are not directly observable through our dataset that the model is dependent upon. In these cases, expectation-maximization comes into play by defining tractable latent variables that correspond to non-tractable variables. This allows us to make conclusions about unobserved variables despite our inability to observe them.

To discuss how expectation-maximization works, we assume a background understanding of basic mathematical notation, statistics, and probability. Given a generative model X with density

$p(X|\theta)$, where $l(\theta) = log\ p(X|\theta)$, this model is equivalent to the marginal of joint probability $p(X, Y|\theta)$, where Y is a latent variable. That is:

$$p(X|\theta) = \sum_{Y} p(X, Y|\theta) = E_{Y \sim p(Y|\theta)}[p(X|Y)]$$

Note that $E_{Y \sim p(Y|\theta)}[p(X|Y)]$ is the expectation of $p(X|Y)$ with respect to Y which is sampled from the distribution $p(Y|\theta)$.

The algorithm is as follows:

1. Initialize θ_0 at step t:
2. Compute $p(Y|X, \theta_{t-1}) = A_{\theta_{t-1}}$, where $A_{\theta_{t-1}}$ is defined in:

   $$L(A_{\theta_{t-1}}, \theta) = E_{Y \sim p(Y|X, \theta_{t-1})}[log\ p(X, Y|\theta)];$$

3. Update θ with the objective function

   $$max_{\theta} L(A_{\theta_{t-1}}, \theta)$$

4. Repeat 2 & 3 until convergence to a solution

Because the second step computes an expectation of the likelihood and the third step computes maximum likelihood estimates, the process is thus called Expectation Maximization. This algorithm runs in rather slowly on its own, and at first it is difficult to find meaning in it. However, we will see its power by noting the following theorem:

Theorem: $l(\theta) = L(A_{\theta}, \theta) \leq L(A_{\theta}, \theta') \leq L(A_{\theta'}, \theta') = l(\theta')$

Where $\theta' = max_{\theta} L(A_{\theta_{t-1}}, \theta)$, our objective function in step 3. What does this mean? It means that each sequence of iterates improves the likelihood of our model. **If none of this made sense to you, the short answer is that expectation maximization is used in probabilistic models that depend on unobserved hidden variables to find maximum likelihood estimates of all parameters, including the unobserved ones.**

Derivations of Expectation Maximization

Though we will not entertain in-depth analyses of various derivations of expectation maximization, we will discuss a few that are particularly useful, especially in document clustering, which is discussed in the next section.

Gaussian Mixture Models (GMMs)

Gaussian mixture models are, in short, a mixture of Gaussian (also known as normal) distributions. These are mixed with a mixture parameter typically defined as π. It works well in situations where a single Gaussian may not fit the data, but more Gaussians will fit the data well. The result in cases where the method works well is that each respective Gaussian distribution corresponds with a cluster of the data.
The GMM algorithm can be defined as follows:

1. Initialize our assignments in a matrix $A \in [0,1]^{n \times k}$, with $i \in [1,n]$ and $j \in [1,k]$
2. Fix model parameters, update assignments:
$$A_{ij} := \frac{\pi_j \cdot \mathcal{N}(x_i, \mu_j, \Sigma_j)}{\Sigma_l \pi_l \cdot \mathcal{N}(x_i, \mu_l, \Sigma_l)}$$
3. Fix assignments, update parameters
$$\pi_j' := \frac{\Sigma_i A_{ij}}{n}$$
$$\mu_j' := \frac{\Sigma_i A_{ij} x_i}{n\pi_j'}$$
$$\Sigma_j' := \frac{\Sigma_i A_{ij} (x_i - \mu_j')(x_i - \mu_j')^T}{n\pi_j'}$$
4. Repeat steps 2 & 3 until convergence to a solution.

Where μ and Σ are respectively the mean and covariance of $\mathcal{N}$, the Gaussian distribution, and $(x_i - \mu_j')^T$ is the transpose of $(x_i - \mu_j')$. It should be obvious that step 2 is the expectation step and step 3 is the maximization step, resulting in an expectation-maximization algorithm.
It's often a simplification of the algorithm to uses diagonal covariance matrices, so the model only requires $O(kd)$ parameters instead of $O(kd^2)$.

K-means

Perhaps the most well known of all unsupervised learning models is k-means, which is actually derived from GMMs, though we will not discuss said derivation in this paper. We define the objective function as:

$$\sum_{i=1}^{n} min_j ||x_i - \mu_j||_2^2$$

Where $\{\mu_1, ..., \mu_k\}$ are the k means and $||x_i - \mu_j||_2^2$ is the Euclidean (also known as L^2) distance between the data and one of the k means. Do note that the k-means objective is not the same as the k-means method. The objective is sometimes called the distortion J. This is the core example of hard clustering discussed in this paper.

The algorithm, commonly known as Lloyd's method, is as follows:

1. Choose initial clusters $\{C_1, ..., C_k\}$
2. Set $\mu_j := mean(C_j)$ for $j \in \{1, ..., k\}$
3. Update $C_j := \{x_i : \mu(x_i) = \mu_j\}$ for $j \in \{1, ..., k\}$ (break ties arbitrarily)
4. Repeat step 2 and 3 until convergence

It should be clear that the second step corresponds to the E-step and the third step corresponds to the maximization step. This is an alternating minimization on cluster assignments and cluster centers that ultimately costs $O(nkd)$ per iteration. We will note that initialization of the clusters matters, but will not discuss it in detail in this paper. This model will prevalent in our discussion of document classification in the next section.

Naive Bayes Clustering Model

Naive Bayes clustering models all assume that the value of a particular feature is independent of the value of any other feature, given the class variable. Wikipedia's example is of an apple; if an apple is 10 cm in diameter, red, and round, a naive Bayes classifier deems these features statistically independent of each other, regardless of possible correlations, an oft "naive" assumption to make. It is important to know that semi-supervised parameter estimation, a way of training naive Bayesian classifiers from labeled data, is an instance of EM, but we will not go

into details about it as it is not necessarily useful for our discussion. However, due to its prevalence in other forms of machine learning we briefly mention it here.

Hidden Markov Models

HMMs rely upon what is known as the Markov process, defined by Wikipedia as "a process for which predictions can be made regarding future outcomes based solely on [the] present state." In addition, it claims that these predictions are equivalent in accuracy to a prediction knowing the full history. This is only possible because it surmises that future and past states are independent of one another.

Hidden Markov Models function in many ways similarly to Gaussian Mixture Models, except for the fact that they have a time series over hidden states. We will not explore this in depth, like Naive Bayes clustering, because it is not prevalent to our discussion of document classification, but still mention it because of its usefulness in various other applications. EM for HMMs is called the Baum-Welch algorithm, where the E-step is called sum-product inference, and the M-step is extremely similar to GMM. If more information regarding HMMs is desired, further exploration beyond the scope of this paper is recommended.

Expectation Maximization in Document Classification

Now that we have discussed more than enough on expectation-maximization, we will discuss document clustering and classification. The main motivations are:

1. **Data with Labels:** Given two clusters of documents, we want to know what cluster a new document belongs to.
2. **Data with No Labels:** Given a pile of documents that we know come from two classes, we want to organize them into their respective clusters.

Because we can observe information about the documents but not the classification of the document itself, we know expectation-maximization serves its purpose here.

K-Means for Document Clustering

In the case where we have data with no labels, we want to be able to categorize a new document into one of the two classes. We can represent each document as feature vectors (which have been discussed previously in the scope of this class, so we will skip defining it). These features can be any NLP feature from word id to POS tags or word context. Our objective is then the minimization of the square distance from the class means to the document feature vector.
It's important to be reminded that k-means assigns each point to the closest cluster, which means that near equidistant points can affect the algorithm, and outliers affect the mean computation equally.

GMMs for Document Clustering

There are generally two cases in which GMMs are chosen over k-means:

1. Clusters of data are non-spherical; in our case, some documents may be closer together to the feature vector mean in squared distance based on one feature than another feature. "Rescaling" of our data is also unintuitive.
2. We want to define clustering of documents with soft assignment.

In these cases, GMM generally works better than k-means, allowing one to give a range of probabilities that a document is part of one of the clusters, in addition to stretching the spherical clusters of k-means into ellipses that equally weigh all features of the feature vector equally. Oftentimes in GMMs we maximize the log likelihood instead of straight up likelihood estimates.

Conclusion

We have discussed through the scope of this tech review the approach of EM as a generic algorithm and the motivations behind it. We continued by discussing a variety of derivations of EM, then concluded by discussing useful algorithms such as k-means and Gaussian mixture models in document clustering and subsequent classification. Through this paper, the reader has gained knowledge in the field of machine learning; specifically, unsupervised learning through the exploration of EM, and learned of its specific applications in document classification, an important problem in the field of text information systems. Further discussions, such as varying proximity measures for objective functions (cosine and Manhattan distance), comparison of hard

and soft EMs for GMMs and actual code relating a specific example of document clustering, are beyond the scope of this paper. The reader is encouraged to explore the abundance of resources outside of this paper to learn more about any of the aforementioned fields.