EM for Semi-Supervised Learning and Mixture Models

Mark Schmidt

February 4, 2021

This document goes over how to apply EM in a semi-supervised learning and a mixture model context.

1 Generic Algorithm

We have observed variables O, discrete hidden variables H, and parameters Θ. We want to do maximum likelihood given our observed variables

$$\max_{\Theta} p(O|\Theta) = \sum_{H} p(O, H|\Theta).$$

A common approach is the expectation maximization (EM) algorithm, which uses iterations of the form

$$\Theta^{k+1} = \operatorname*{argmax}_{\Theta} Q(\Theta|\Theta^k),$$

where

$$Q(\Theta|\Theta^k) = \mathbb{E}_{H|O,\Theta^k}[\log p(O, H|\Theta)] = \sum_{H} p(H|O, \Theta^k) \log p(O, H|\Theta).$$

The algorithm is most suitable when $\log p(O, H|\Theta)$ has a "nice" form.

2 Semi-Supervised Learning

Consider the semi-supervised supervised learning scenario where we have labeled data (X, y) as well as unlabeled data $\tilde{X}$ (and all examples are IID). We'll assume that y only contains binary (0 or 1) values, but the below extends in a straightforward way to the case where we have more than two classes. We're going to consider using a *generative* classifier that models the joint probability $p(x^i, y^i)$ of features x^i and labels y^i.

In this setting, the "complete-data" likelihood (where we assume that we know the labels $\tilde{y}$) is

$$p(X, y, \tilde{X}, \tilde{y}) = \left(\prod_{i=1}^{n} p(x^i, y^i)\right)\left(\prod_{i=1}^{t} p(\tilde{x}^i, \tilde{y}^i)\right),$$

but since we don't actually have the $\tilde{y}^i$ values we'll apply the EM algorithm.

2.1 Expectation Maximization Algorithm

If we want to optimize parameters Θ using the EM algorithm, the expected complete-data log-likelihood which can be written as

$$
\begin{aligned}
Q(\Theta|\Theta^k) &= \mathbb{E}_{\tilde{y}|X,y,\tilde{X},\Theta^k}[\log p(X,y,\tilde{X},\tilde{y}|\Theta)] \\
&= \sum_{\tilde{y}} p(\tilde{y}|X,y,\tilde{X},\Theta^k) \log p(X,y,\tilde{X},\tilde{y}|\Theta) \\
&= \sum_{\tilde{y}} p(\tilde{y}|X,y,\tilde{X},\Theta^k) \left[\sum_{i=1}^{n} \log p(y^i,x^i|\Theta) + \sum_{i=1}^{t} \log p(\tilde{y}^i,\tilde{x}^i|\Theta) \right] \\
&= \sum_{i=1}^{n} \log p(y^i,x^i|\Theta) \underbrace{\sum_{\tilde{y}} p(\tilde{y}|X,y,\tilde{X},\Theta^k)}_{=1} + \sum_{i=1}^{t} \sum_{\tilde{y}} p(\tilde{y}|X,y,\tilde{X},\Theta^k) \log p(\tilde{y}^i,\tilde{x}^i|\Theta) \\
&= \sum_{i=1}^{n} \log p(y^i,x^i|\Theta) + \sum_{i=1}^{t} \sum_{\tilde{y}} \left(\prod_{j=1}^{t} p(\tilde{y}^j|X,y,\tilde{X},\Theta^k) \right) \log p(\tilde{y}^i,\tilde{x}^i|\Theta) \\
&= \sum_{i=1}^{n} \log p(y^i,x^i|\Theta) + \sum_{i=1}^{t} \sum_{\tilde{y}} \left(\prod_{j=1}^{t} p(\tilde{y}^j|\tilde{x}^j,\Theta^k) \right) \log p(\tilde{y}^i,\tilde{x}^i|\Theta) \qquad (*) \\
&= \sum_{i=1}^{n} \log p(y^i,x^i|\Theta) + \sum_{i=1}^{t} \sum_{\tilde{y}^1\in\{0,1\}} \sum_{\tilde{y}^2\in\{0,1\}} \cdots \sum_{\tilde{y}^{t-1}\in\{0,1\}} \sum_{\tilde{y}^t\in\{0,1\}} \left(\prod_{j=1}^{t} p(\tilde{y}^j|\tilde{x}^j,\Theta^k) \right) \log p(\tilde{y}^i,\tilde{x}^i|\Theta) \\
&= \sum_{i=1}^{n} \log p(y^i,x^i|\Theta) + \sum_{i=1}^{t} \sum_{\tilde{y}^i\in\{0,1\}} p(\tilde{y}^i|\tilde{x}^i,\Theta^k) \log p(\tilde{y}^i,\tilde{x}^i|\Theta) \underbrace{\sum_{\tilde{y}^1} p(\tilde{y}^1|\tilde{x}^1,\Theta^k) \underbrace{\sum_{\tilde{y}^2} p(\tilde{y}^2|\tilde{x}^2,\Theta^k) \cdots \underbrace{\sum_{\tilde{y}^t} p(\tilde{y}^t|\tilde{x}^t,\Theta^k)}_{=1}}_{=1}}_{=1} \\
&= \sum_{i=1}^{n} \log p(y^i,x^i|\Theta) + \sum_{i=1}^{t} \sum_{\tilde{y}^i\in\{0,1\}} p(\tilde{y}^i|\tilde{x}^i,\Theta^k) \log p(\tilde{y}^i,\tilde{x}^i|\Theta) \\
&= \sum_{i=1}^{n} \log p(y^i,x^i|\Theta) + \sum_{i=1}^{t} \sum_{\tilde{y}^i\in\{0,1\}} r^i_{\tilde{y}^i} \log p(\tilde{y}^i,\tilde{x}^i|\Theta). \qquad (**)
\end{aligned}
$$

Although the above looks complicated, the only three things we have used are the distributive law ($\sum_i ab_i = a\sum_i b_i$), that conditional probabilities sum to 1, and in $(*)$ that $\tilde{y}^j$ is conditionally independent of the other examples given $\tilde{x}^j$ and Θ^k. In the last line we've defined

$$r^i_0 = p(\tilde{y}^i = 0|\tilde{x}^i,\Theta^k), \quad r^i_1 = p(\tilde{y}^i = 1|\tilde{x}^i,\Theta^k),$$

which are our current estimates for the probabilities of each of the labels in the unlabeled examples. Typically, we'll compute these quantities by normalizing the joint probability, which for r^i_0 would give:

$$r^i_0 = p(\tilde{y}^i = 0|\tilde{x}^i,\Theta^k) = \frac{p(\tilde{y}^i,\tilde{x}^i|\Theta^k)}{\sum_{y\in\{0,1\}} p(y,\tilde{x}^i|\Theta^k)}.$$

Thus, in the end the EM algorithm for semi-supervised learning alternates between two simple steps:

1. E-step: Compute probabilities r_0^i and r_1^i for all the unlabeled examples i based on the current Θ^k.
2. M-step: Maximize the expected complete-data log-likelihood (**), which is a weighted version of the complete-data log-likelihood.

2.2 Multi-Class and Class-Probability Estimates

Following the argument above, the EM algorithm with k classes is given by

$$\Theta^{k+1} = \underset{\Theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log p(y^i, x^i|\Theta) + \sum_{i=1}^{t} \sum_{\tilde{y}^i \in \{1,2,\ldots,k\}} r_{\tilde{y}^i}^i \log p(\tilde{y}^i, \tilde{x}^i|\Theta),$$

where r_c^i is defined as above.

We often use the product rule to write the joint probability of an x^i and y^i as

$$p(y^i, x^i|\Theta) = p(x^i|y^i, \Theta)p(y^i|\Theta).$$

Many supervised learning algorithms differ only in how they address the class-conditional density estimation problem of modelling $p(x^i|y^i, \Theta)$. Naive Bayes assumes that the x_j^i are conditionally independent given y^i, while Gaussian discriminant analysis assumes that this is a Gaussian distribution. Regardless of the choice of density estimator, updating the Θ associated with $p(x^i|y^i, \Theta)$ in the semi-supervised case will take the form a weighted version of the problem we solve in the supervised case.

On the other hand, estimating the parameters of $p(y^i|\Theta)$ is typically easy. If we assume a categorical distribution for y^i then the solution to the EM update is given by

$$\pi_c^{k+1} = \frac{n_c + \sum_{i=1}^{t} r_c^i}{n + t},$$

where n_c is the number of instances of class c in the labeled data. This result is intuitive: we count the number of times we saw c (as in the fully observed case), plus the probability that we associate with class c on the unlabeled examples. (This result can be shown by using Lagrange multipliers to enforce the constraint on the update that $\sum_c \pi_c^{k+1} = 1$.)

2.3 Efficient Calculation of Observed-Data Log-Likelihood

The "observed-data" likelihood that EM is trying to maximize is given by

$$\begin{aligned} p(X, y, \tilde{X}) &= \sum_{\tilde{y}} \left(\prod_{i=1}^{n} p(x^i, y^i) \right) \left(\prod_{i=1}^{t} p(\tilde{x}^i, \tilde{y}^i) \right) \\ &= \sum_{\tilde{y}^1 \in \{0,1\}} \sum_{\tilde{y}^2 \in \{0,1\}} \cdots \sum_{\tilde{y}^t \in \{0,1\}} \left(\prod_{i=1}^{n} p(x^i, y^i) \right) \left(\prod_{i=1}^{t} p(\tilde{x}^i, \tilde{y}^i) \right). \end{aligned}$$

where we've used $\sum_{\tilde{y}}$ to denote the sum over all 2^t possible values of the $\tilde{y}$ vector. Technically, we don't need to be able to compute the observed-data log-likelihood to apply the EM algorithm: we could just run the EM algorithm knowing that it will increase the observed-data log-likelihood on each iteration. But computing the observed-data log-likelihood can useful for debugging an implementation, for diagnosing convergence of EM, or as a direct target of optimization if we don't like EM.

Even though the observed-data likelihood seems intractable to compute because it sums over 2^t terms, the independence between examples makes it easy to compute. In particular, the observed-data likelihood can

be written

$$
\begin{aligned}
p(X, y, \tilde{X}) &= \sum_{\tilde{y}^1 \in \{0,1\}} \sum_{\tilde{y}^2 \in \{0,1\}} \cdots \sum_{\tilde{y}^t \in \{0,1\}} \left(\prod_{i=1}^{n} p(x^i, y^i) \right) \left(\prod_{i=1}^{t} p(\tilde{x}^i, \tilde{y}^i) \right) \\
&= \underbrace{\left(\prod_{i=1}^{n} p(x^i, y^i) \right)}_{\text{labeled}} \sum_{\tilde{y}^1 \in \{0,1\}} \sum_{\tilde{y}^2 \in \{0,1\}} \cdots \sum_{\tilde{y}^t \in \{0,1\}} \underbrace{\left(\prod_{i=1}^{t} p(\tilde{x}^i, \tilde{y}^i) \right)}_{\text{unlabeled}} \\
&= \left(\prod_{i=1}^{n} p(x^i, y^i) \right) \left(\prod_{i=1}^{t} \left[\sum_{y^i \in \{0,1\}} p(\tilde{x}^i, \tilde{y}^i) \right] \right) \qquad (***)
\end{aligned}
$$

In the second line we use that the observed examples (x^i, y^i) do not depend on any $\tilde{y}^i$ so we can take the factor outside the sum (in other words, we are again using that $\sum_i ab_i = a \sum_i b_i$). The third line is based on the same idea, but uses independence to "factorize" the sum. We'll see arguments like this several times when we discuss graphical models, so it's worth going over it in detail. In particular, in the third line the logic is that

$$
\begin{aligned}
\sum_{\tilde{y}^1 \in \{0,1\}} \sum_{\tilde{y}^2 \in \{0,1\}} \cdots \sum_{\tilde{y}^{t-1} \in \{0,1\}} \sum_{\tilde{y}^t \in \{0,1\}} \left(\prod_{i=1}^{t} p(\tilde{x}^i, \tilde{y}^i) \right) &= \sum_{\tilde{y}^1 \in \{0,1\}} \sum_{\tilde{y}^2 \in \{0,1\}} \cdots \sum_{\tilde{y}^{t-1} \in \{0,1\}} \sum_{\tilde{y}^t \in \{0,1\}} \left(\prod_{i=1}^{t-1} p(\tilde{x}^i, \tilde{y}^i) \right) p(\tilde{x}^t, \tilde{y}^t) \\
&= \sum_{\tilde{y}^1 \in \{0,1\}} \sum_{\tilde{y}^2 \in \{0,1\}} \cdots \sum_{\tilde{y}^{t-1} \in \{0,1\}} \left(\prod_{i=1}^{t-1} p(\tilde{x}^i, \tilde{y}^i) \right) \sum_{\tilde{y}^t \in \{0,1\}} p(\tilde{x}^t, \tilde{y}^t) \\
&= \left(\sum_{\tilde{y}^t \in \{0,1\}} p(\tilde{x}^t, \tilde{y}^t) \right) \sum_{\tilde{y}^1 \in \{0,1\}} \sum_{\tilde{y}^2 \in \{0,1\}} \cdots \sum_{\tilde{y}^{t-1} \in \{0,1\}} \left(\prod_{i=1}^{t-1} p(\tilde{x}^i, \tilde{y}^i) \right) \\
&= \left(\sum_{\tilde{y}^t \in \{0,1\}} p(\tilde{x}^t, \tilde{y}^t) \right) \left(\sum_{\tilde{y}^{t-1} \in \{0,1\}} p(\tilde{x}^{t-1}, \tilde{y}^{t-1}) \right) \cdots \left(\sum_{\tilde{y}^1 \in \{0,1\}} p(\tilde{x}^1, \tilde{y}^1) \right) \\
&= \prod_{i=1}^{t} \left[\sum_{y^i \in \{0,1\}} p(\tilde{x}^i, \tilde{y}^i) \right].
\end{aligned}
$$

The first line just takes the last term outside of the product. In the second line we take the terms not depending on $\tilde{y}^t$ outside the sum over $\tilde{y}^t$. This is the same as before, but we are now still left with the remaining term $p(\tilde{x}^t, \tilde{y}^t)$. In the third line we *treat the sum over* $p(\tilde{x}^t, \tilde{y}^t)$ *as a constant* that does not depend on the sums over the other $\tilde{y}^i$, so we can take it out of all the other sums. The fourth line comes from repeating the first three lines for $(t-1)$, then $(t-2)$, and so on until we only have the sum over $\tilde{y}^1$ left. The last line writes this in product notation which gives the result $(***)$. Thus, even though this is a sum over 2^t terms, each containing a product over j terms, we *only need to compute the product of* j *numbers that are each a sum over only* 2 *terms*. This trick is a simple form of what is known as *dynamic programming*.

The expression $(***)$ lets us write the observed log-likelihood as

$$
\log p(X, y, \tilde{X}) = \sum_{i=1}^{n} \log p(y^i, x^i) + \sum_{i=1}^{t} \log \left(\sum_{\tilde{y}^i \in \{0,1\}} p(\tilde{y}^i, \tilde{x}^i) \right).
$$

which is easy to compute but non-convex.

3 Mixture Models

The most common usage of the EM algorithm is for fitting Gaussian mixture models. A mixture model assumes that probability of x^i is given by

$$\begin{aligned} p(x^i) &= \sum_{c=1}^{k} p(x^i, z^i = c) \\ &= \sum_{c=1}^{k} p(z^i = c) p(x^i | z^i = c). \end{aligned}$$

We can view the values c as a set of k clusters of the data, while z^i is the cluster membership of x^i. One way to intrepret this formula is is that we first generate the cluster z^i according to $p(z^i)$, and then given z^i we sample the data point x^i from $p(x^i|z^i)$. Typically, $p(x^i|z^i)$ has a simple form like a Gaussian distribution, so it would be easy to fit the model if we knew the z^i values. But, we don't actually know the z^i values. This is where EM fits in: if we treat the z^i as hidden values, then the EM iterations typically have a simple form.

3.1 General Mixture Models

Before specifically talking about the Gaussian case, let's first look at the case of a general mixture model. First, since the samples are IID the complete-data log-likelihood with parameters Θ is given by

$$\log p(X, z|\Theta) = \sum_{i=1}^{n} \log p(x^i, z^i|\Theta).$$

Following the same steps as the semi-supervised case, we can write the Q function in the EM algorithm in the form

$$\begin{aligned} Q(\Theta|\Theta^k) &= \mathbb{E}_{z|X,\Theta}[\log p(X, z|\Theta)] \\ &= \sum_{z} p(z|X, \Theta^k) \log p(X, z|\Theta) \\ &= \sum_{z} p(z|X, \Theta^k) \sum_{i=1}^{n} \log p(x^i, z^i|\Theta) \\ &= \sum_{i=1}^{n} \sum_{z} p(z|X, \Theta^k) \log p(x^i, z^i|\Theta) \\ &= \sum_{i=1}^{n} \sum_{z} \left(\prod_{j=1}^{n} p(z^i|x^i, \Theta^k) \right) \log p(x^i, z^i|\Theta) \\ &= \sum_{i=1}^{n} \left(\sum_{z^1=1}^{k} \sum_{z^2=1}^{k} \cdots \sum_{z^n=1}^{k} \left(\prod_{j=1}^{n} p(z^i|x^i, \Theta^k) \right) \right) \log p(x^i, z^i|\Theta) \\ &= \sum_{i=1}^{n} \sum_{z^i=1}^{k} p(z^i|x^i, \Theta^k) \log p(x^i, z^i|\Theta) \qquad :) \\ &= \sum_{i=1}^{n} \sum_{z^i=1}^{k} r^i_{z^i} \log p(x^i, z^i|\Theta) \\ &= \sum_{i=1}^{n} \sum_{z^i=1}^{k} r^i_{z^i} \log p(z^i|\Theta) + \sum_{i=1}^{n} \sum_{z^i=1}^{k} r^i_{z^i} \log p(x^i|z^i, \Theta). \end{aligned}$$

where we've introduced the "responsibilities" $r^i_{z^i} = p(z^i|x^i, \Theta^k)$ and where :) uses the earlier logic that the sum over the k^n products turns into a simple sum over k values by using that the terms are independent and probabilities sum to 1. We can compute the responsibility of cluster c for example i given the old parameters Θ^k using

$$r^i_c = p(z^i = c|x^i, \Theta^k) = \frac{p(z^i = c, x^i|\Theta^k)}{\sum_{c'=1}^k p(z^i = c', x^i|\Theta^k)}.$$

If we assume that z_i follows a categorical distribution with parameters $\{\theta_1, \theta_2, \ldots, \theta_k\}$, then the minimizer of this expression in terms of a generic θ_c is given by

$$\theta_c = \frac{\sum_{i=1}^n r^i_c}{n}.$$

This follows from the definition of the categorical distribution and introducing Lagrange multipliers to enforce the constraints that $\sum_c \theta_c = 1$. Note that this is always the update for θ_c, as long as we make the (very realistic) assumption that $p(x^i|z^i, \Theta)$ is not a function of any θ_c.

3.2 Gaussian Mixture Models

We obtain a mixture of Gaussians in the case that each $p(x^i|z^i, \Theta)$ is a Gaussian distribution. Although it's straightforward to use other parameterizations, if we assume that each cluster c has its own mean μ_c and full covariance Σ_c, then we have

$$\log p(x^i|z^i = c, \Theta) = \frac{1}{2}(x^i - \mu_c)^T \Sigma_c^{-1}(x^i - \mu_c) + \frac{1}{2}\log|\Sigma_c| + \text{const.}$$

This implies that

$$Q(\Theta|\Theta^k) = \sum_{i=1}^n \sum_{z'=1}^k r^i_{z^i} \log p(z^i|\theta_{z^i}) + \sum_{i=1}^n \sum_{z^i=1}^k r^i_{z^i} \left(\frac{1}{2}(x^i - \mu_{z^i})^T \Sigma_{z^i}^{-1}(x^i - \mu_{z^i}) + \frac{1}{2}\log|\Sigma_{z^i}| \right) + \text{const.}$$

The update for θ_c is given in the previous section. Taking the derivative with respect to μ_c and setting it equal to zero we get

$$0 = \sum_{i=1}^n -r^i_c \Sigma_c^{-1}(x^i - \mu_c),$$

or equivalently that

$$\Sigma_c^{-1}\mu_c \sum_{i=1}^n r^i_c = \Sigma_c^{-1} \sum_{i=1}^n r^i_c x^i.$$

Pre-multiplying by Σ_c and solving for μ_c gives

$$\mu_c = \frac{\sum_{i=1}^n r^i_c x^i}{\sum_{i=1}^n r^i_c} = \frac{\sum_{i=1}^n r^i_c x^i}{n\theta_c},$$

which is the mean of the points that have been partially assigned to cluster c. Following a similar argument, we get that the covariance is given by

$$\Sigma_c = \frac{\sum_{i=1}^n r^i_c (x^i - \mu_c)(x^i - \mu_c)^T}{n\theta_c}.$$