The EM algorithm

A very general and well-studied algorithm
I cover only the specific case we use in this course: maximum-likelihood estimation for models with discrete hidden variables
(For continuous case, sums go to integrals; for MAP estimation, changes to accommodate prior)
As an easy example we estimate parameters of an \( n \)-gram mixture model
For all details of EM, try McLachlan and Krishnan (1996)

Maximum-Likelihood Estimation

In practice, it’s often hard to get expressions for the derivatives needed by gradient methods
EM is one popular and powerful way of proceeding, but not the only way.
Remember, EM is doing MLE

Finding parameters of a \( n \)-gram mixture model

\( P \) may be a mixture of \( k \) pre-existing multinomials:
\[
P(x_j | \Theta) = \sum_{j=1}^{k} \theta_j P_j(x_j)
\]
\[
\hat{P}(w_3 | w_1, w_2) = \theta_3 P_3(w_3 | w_1, w_2) + \theta_2 P_2(w_3 | w_2) + \theta_1 P_1(w_3)
\]
We treat the \( P_j \) as fixed. We learn by EM only the \( \theta_j \).
\[
P(X|\Theta) = \prod_{i=1}^{n} P(x_i | \Theta)
= \prod_{i=1}^{n} \sum_{j=1}^{k} P_j(x_i | \Theta_j)
\]
\( X = [x_1 \ldots x_n] \) is a sequence of \( n \) words drawn from a vocabulary \( V \), and \( \Theta = [\theta_1 \ldots \theta_k] \) are the mixing weights

EM and Hidden Structure

In the first case you might be using EM to “fill in the blanks” where you have missing measurements.
The second case is strange but standard. In our mixture model, viewed generatively, if each data point \( x \) is assigned to a single mixture component \( y \), then the probability expression becomes:
\[
P(X,Y|\Theta) = \prod_{i=1}^{n} P(x_i, y_i | \Theta)
= \prod_{i=1}^{n} P_{y_i}(x_i | \Theta)
\]
Where \( y_i \in \{1, \ldots, k\} \). \( P(X,Y|\Theta) \) is called the complete-data likelihood.
EM and Hidden Structure

Note:
- The sum over components is gone, since \( y_i \) tells us which single component \( x_i \) came from. We just don’t know what the \( y_i \) are.
- Our model for the observed data \( X \) involved the “unobserved” structures – the component indexes – all along. When we wanted the observed-data likelihood we summed out over indexes.
- There are two likelihoods floating around: the observed-data likelihood \( P(X|\Theta) \) and the complete-data likelihood \( P(X,Y|\Theta) \). EM is a method for maximizing \( P(X|\Theta) \).

The EM algorithm

The actual algorithm is as follows:

Initialize Start with a guess at \( \Theta \) – it may be a very bad guess

Until tired

E-Step Given a current, fixed \( \Theta' \), calculate completions: \( P(Y|X,\Theta') \)

M-Step Given fixed completions \( P(Y|X,\Theta') \), maximize \( \sum Y P(Y|X,\Theta') \log P(X,Y|\Theta) \) with respect to \( \Theta \).

EM made easy

Want: \( \Theta \) which maximizes the data likelihood

\[
L(\Theta) = P(X|\Theta) = \sum Y P(X,Y|\Theta)
\]

The \( Y \) ranges over all possible completions of \( X \). Since \( X \) and \( Y \) are vectors of independent data items,

\[
L(\Theta) = \prod_x \sum_y P(x,y|\Theta)
\]

We don’t want a product of sums. It’d be easy to maximize if we had a product of products.

Each \( x \) is a data item, which is broken into a sum of sub-possibilities, one for each completion \( y \). We want to make each completion be like a mini data item, all multiplied together with other data items.

EM and Hidden Structure

Looking at completions is useful because finding

\[
\Theta = \arg \max_{\Theta} P(X|\Theta)
\]

is hard (it’s our original problem – maximizing products of sums is hard)

On the other hand, finding

\[
\Theta = \arg \max_{\Theta} P(X,Y|\Theta)
\]

would be easy – if we knew \( Y \).

The general idea behind EM is to alternate between maximizing \( \Theta \) with \( Y \) fixed and “filling in” the completions \( Y \) based on our best guesses given \( \Theta \).

The EM algorithm

In the E-step we calculate the likelihood of the various completions with our fixed \( \Theta' \).

In the M-step we maximize the expected log-likelihood of the complete data. That’s not the same thing as the likelihood of the observed data, but it’s close.

The hope is that even relatively poor guesses at \( \Theta \), when constrained by the actual data \( X \), will still produce decent completions.

Note that “the complete data” changes with each iteration.

EM made easy

Want: a product of products

Arithmetic-mean-geometric-mean (AMGM) inequality says that, if \( \sum w_i = 1 \),

\[
\prod_{i} z_i^{w_i} \leq \sum w_i z_i
\]

In other words, arithmetic means are larger than geometric means (for 1 and 9, arithmetic mean is 5, geometric mean is 3)

This equality is promising, since we have a sum and want a product.

We can use \( P(x,y|\Theta) \) as the \( z_i \), but where do the \( w_i \) come from?
EM made easy

- The answer is to bring our previous guess at $\Theta$ into the picture.
- Let’s assume our old guess was $\Theta'$. Then the old likelihood was
  \[ L(\Theta') = \prod_x P(x|\Theta') \]
- This is just a constant. So rather than trying to make $L(\Theta)$ large, we could try to make the relative change in likelihood
  \[ R(\Theta|\Theta') = \frac{L(\Theta)}{L(\Theta')} \]
  large.

EM made easy

- We can use our identity to turn the sum into a product:
  \[ R(\Theta|\Theta') = \prod_x \sum_y \frac{P(x,y|\Theta)}{P(x,y|\Theta')} P(y|x,\Theta') \]
- $\Theta$, which we’re maximizing, is a variable, but $\Theta'$ is just a constant. So we can just maximize
  \[ Q(\Theta|\Theta') = \prod_x \sum_y P(x,y|\Theta) P(y|x,\Theta') \]

The EM Algorithm

- So here’s EM, again:
  - Start with an initial guess $\Theta'$.
  - Iteratively do
    - **E-Step** Calculate $P(y|x,\Theta')$
    - **M-Step** Maximize $Q(\Theta|\Theta')$ to find a new $\Theta'$
- In practice, maximizing $Q$ is just setting parameters as relative frequencies in the complete data – these are the maximum likelihood estimates of $\Theta$