Take-home message of EM algorithm

Han Xiao
Technical University of Munich, Germany
xiaoh@in.tum.de
July 8, 2012

1 Modeling and Derivation

- Always set prior for your parameter \( \theta \). Instead of maximizing the conditional density \( \log p(x | \theta) \), maximize the joint density \( \log p(x, \theta) \). It doesn’t complicate your derivation!

- Given a graphical model, it is always easier to write down the complete joint density \( \log p(x, z, \theta) \) than some marginal densities, i.e. \( \log p(x, \theta) \).

- Don’t use EM unless necessary! First try to write down \( \log p(x, \theta) \), if you find it is tractable, then just maximize it with your favorite gradient-based approach! By intractable I mean the latent variable \( z \) can be very hard to be “integrated out”. In this case, \( \log p(x, \theta) \) is in a very messy form. On the other hand, the complete density \( \log p(x, z, \theta) \) is usually (almost always) clean and elegant. That’s where the EM algorithm comes to the play.

- Again, make sure that you have used all tricks and there is really no way to write down the marginal density. I emphasis this again since some densities (e.g. Gaussian) have very nice property, which can be used to simplify the integral! A good example is probabilistic PCA model. Although there is a latent variable involved in the mode, the marginal density is still tractable and can be readily calculated. Thus, we don’t need EM algorithm for probabilistic PCA.

- In M-step, by taking the derivative to zero, you need to consider whether there is a close form solution for the stationary point. If not, use gradient-based approach.

2 Approximate Inference in E-step

- If the posterior \( p(z | x) \) is intractable (which is quite often if you deal with continuous distributions), you will need approximate inference to approximate the posterior. Generally, there are two options (actually three options, but I haven’t fully understand the Expectation Propagation algorithm): Monte Carlo sampling and variational inference.

- Sampling method includes: reject sampling, Metropolis Hastings sampling, Gibbs sampling, etc. The upside is sometimes you don’t need much derivation. For instance, in Metropolis Hastings sampling, all you need is an unnormalized density. The downside is the low efficiency. Especially if the dimension is high, sampling method will encounter many problems (e.g. high rejection rate) and becomes extremely slow!

- In high-dimensional space, a proper way to avoid high rejection rate is to use component-wise Metropolis Hastings. But it does not improve the efficiency!

- To perform variational inference, you need derivation. It is completely worth the price, when you realize how fast it is. In fact, if your solution is in close form, then you can compute the latent variable in one line.

- If you have only one latent variable, variational inference could be extremely easy!

3 Implementation

- For the sake of numerical stability, do not use \( \text{inv}(A) \) to invert a matrix! Use Cholesky decomposition instead: \( LL^T = A \). Thus, \( A^{-1} = (L^{-1})(L^{-1})^T \).

- Cholesky decomposition requires your matrix to be positive definite. If your matrix supposed to be positive definite, you can add a small jitter (e.g. \( 10^{-5} \)) to the diagonal of the matrix until it becomes a positive definite matrix. If your derivation is wrong, then don’t bother it.

- Do not write \( \text{inv}(A)B \) for \( A^{-1}B \), write \( A\backslash B \), and \( B/A \) for \( BA^{-1} \). Always use Cholesky decomposition in the first place.

- A big advantage of the close form solution in M-step is that you don’t need to compute the \( \log |A| \) for checking the convergence. Calculating is determinant of a big matrix is very time consuming, which should be avoided! Besides, computing the determinant for an arbitrary matrix is not numerically safe. If you really need to compute the determinant (usually happens when your model involves multivariate Gaussian and you apply gradient based method for M-step), then do Cholesky decomposition and use \( 2 \times \sum \log (\text{diag}(L)) \).

- Matrix manipulation (especially multiplication and inversion) is more stable for triangle matrix.

- Always compute \( \log p(x) \) instead of \( p(x) \).
In the derivation, you usually maximize the likelihood. However, many gradient-based optimization algorithms minimize the function by default. Therefore, it is important to keep your function and derivatives in the correct direction. Often, \( f = -f; \ df = -df \) is needed.

Tom Minka’s “lightspeed” toolbox for MATLAB provides some helpful functions for matrix manipulation that are more stable and faster than the original MATLAB function.

4 Debug

- Do not ignore the “matrix close to singular” warning! This usually indicates your algorithm is not numerically stable and the result could be completely wrong! Also pay attention to Inf, NaN warning!

- You probably will encounter such warnings after several iterations. It’s time to check your code again and make sure every line is correct and numerically stable!

- The most easiest way to validate your derivation of the derivative is to use MATLAB \texttt{fminunc} and set

  \[ \text{options} = \text{optimset}('GradObj','on','FunValCheck','on','DerivativeCheck','on'); \]

  If the difference between your gradient and the artificial gradient is too big (e.g. \( > 10^{-4} \)), then you must have a wrong derivation.

- \texttt{fminunc} is only for debug and toy examples, it’s too slow for even a medium-scale experiment. Once you double checked your code and derivation, try some other optimization packages instead.