10-708: Probabilistic Graphical Models, Spring 2015

8 : Learning Partially Observed GM: the EM algorithm

Lecturer: Eric P. Xing *Scribes: Aurick Qiao, Hao Zhang, Bing Liu*

1 Introduction

Two fundamental questions in graphical models are learning and inference. Learning is the process of estimating the parameters, and in some cases, the topology of the network, from data. We have covered learning methods for completely observed graphical models, both directed and undirected, in previous chapters. Focus of this chapter is on learning of partially observed or unobserved graphical models. Two estimation principles that will be covered are maximal margin and maximum entropy.

On inference approaches, we previously covered the elimination algorithm and message-passing algorithm as exact inference techniques. Other exact inference algorithms include the junction tree algorithm, which is less often used these days. Doing exact inference, however, can be expensive, especially in the domain of big data. Thus, many efforts have been placed on approximate inference techniques. These include stochastic simulation, sampling methods, Markov chain Monte Carlo methods, Variational algorithms, etc.

2 Mixture Models

Mixture models are models where the probability distribution of a data point is defined by a weighted sum of conditional likelihoods. We say that a distribution f is a mixture of K component distribution $p_1, p_2, ..., p_K$ if:

$$p(x) = \sum_k \lambda_k P_k(x)$$

with the $\lambda_k > 0$ being the mixture proportion, and $\sum_k \lambda_k = 1$

2.1 Unobserved Variables

A variable can be unobserved (latent) for a number of reasons. It might be an imaginary quantity meant to provide some simplified and abstract view of the data generation process, for example the speech recognition models. It also can be a real-world object or phenomena which is difficult or impossible to measure, such as the temperature of a star, causes of a disease, and evolutionary ancestors. Or else it might be a real-world object or phenomena which was not sometimes measured, due to faulty system components, etc.

Latent variable can be either discrete or continuous. Discrete latent variables can be used to partition or cluster data into sub-groups. Continuous latent variables (factors), on the other hand, can be used for dimensionality reduction (factor analysis, etc).

2.2 Gaussian Mixture Models

A Gaussian mixture model is a probabilistic model in which we assume all data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. Consider a mixture of K Gaussian mixtures,

$$P(X_n|\mu, \Sigma) = \sum_k \pi_k \mathcal{N}(X|\mu_k, \Sigma_k)$$

where π_k is the mixture proportion, and $\mathcal{N}(X|\mu_k, \Sigma_k)$ is the mixture component. Let's see how this expression is derived. By introducing Z as a latent class indicator vector:

$$P(z_n) = muitinomial(z_n : \pi) = \prod_k (\Pi_k)^{z_n^k}$$

We have X as a conditional Gaussian variable with a class-specific mean and covariance:

$$P(x_n|z_n^k = 1, \mu, \Sigma) = \frac{1}{(2\pi)^{m/2} |\Sigma_k|^{1/2}} exp \left\{ -\frac{1}{2}(x_n - \mu_k)^T \Sigma_k^{-1} (x_n - \mu_k) \right\}$$

Therefore, the likelihood of a sample x_n can be expressed as:

$$\begin{aligned} P(x_n|\mu, \Sigma) &= \sum_k P(z_k, x) \\ &= \sum_k p(z^k = 1|\pi) p(x|z^k = 1, \mu, \Sigma) \\ &= \sum_{z_n} \prod_k \left[(\pi_k)^{z_n^k} N(x_n : \mu_k, \Sigma_k)^{z_n^k}) \right] \\ &= \sum_k \pi_k N(x|\mu_k, \Sigma_k)) \end{aligned}$$

Now we want to do learning of the mixture model. In fully observed i.i.d settings, the log likelihood can be decomposed into a sum of local terms (at least for directed models). This can be illustrated as:

$$\ell(\theta, D) = \log p(x, z|\theta) = \log p(z|\theta_z) + \log p(x|z, \theta_x)$$

With latent variables, however, all the parameters become coupled together via marginalization:

$$\ell(\theta, D) = \log \sum_z p(x, z|\theta) = \log \sum_z p(z|\theta_z) p(x|z, \theta_x)$$

It can be observed that the summation is inside the log and the log likelihood does not decompose nicely. It is hard to find a closed form solution to the maximum likelihood estimates. We then turn to learn the mixture densities using gradient descent on the log likelihood, with:

$$\ell(\theta) = \log p(x|\theta) = \log \sum_k \pi_k p_k(x|\theta_k)$$

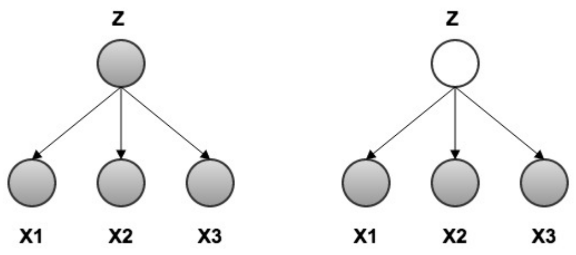

Figure 1: Mixture Model Learning with Latent Variables

By taking the derivative with respect to θ we have:

$$\begin{aligned}\frac{\partial \ell}{\partial \theta} &= \frac{1}{p(x|\theta)} \sum_k \pi_k \frac{\partial p_k(x|\theta_k)}{\partial \theta} \\ &= \sum_k \frac{\pi_k}{p(x|\theta)} p_k(x|\theta_k) \frac{\partial \log p_k(x|\theta_k)}{\partial \theta} \\ &= \sum_k \pi_k \frac{p_k(k|\theta_k)}{p(x|\theta)} \frac{\partial \log p_k(x|\theta_k)}{\partial \theta_k} \\ &= \sum_k r_k \frac{\partial \ell_k}{\partial \theta_k} = 0\end{aligned}$$

Thus, the gradient is the responsibility weighted sum of the individual log likelihood gradients. This can be passed to a conjugate gradient routine.

3 Parameter Constraints

As we can see above, mixture densities with latent variables are not easy to learn. By using gradient approach when no closed form solution can be derived, one has less controls on the conditions of the parameters. In order to mitigate such effect, people did come up with solutions by imposing constraints on parameters. Often we have constraints on the parameters, e.g.$\sum_k \pi_k = 1$, and letting Σ being symmetric positive definite (hence $\Sigma_{ii} > 0$). We can either use constrained optimization, or we can re-parametrize in terms of unconstrained values. For normalized weights, one can use the Softmax transform. For covariance matrices, one can use the Cholesky decomposition in form of:

$$\Sigma^{-1} = A^T A$$

where A is upper diagonal with positive diagonal:

$$A_{ii} = exp(\lambda_i) > 0;$$

$$A_{ij} = \eta_{ij} (j > i);$$

$$A_{ij} = 0 (j < i)$$

where the parameters λ_i and η_{ij} are unconstrained. Then one can use the chain rule to compute $\frac{\partial \ell}{\partial \pi}$ and $\frac{\partial \ell}{\partial A}$.

4 Identifiability

Identifiability is a property that a model has to satisfy in order do to precise inference. A mixture model induces a multi-modal likelihood. Hence, gradient ascent can only find a local maximum. Mixture models are unidentifiable, since we can always switch the hidden labels without affecting the likelihood. Therefore, we should be careful in trying to interpret the "meaning" of latent variables.

5 Expectation-Maximization Algorithm

We will continue to use the Gaussian mixture model to demonstrate the expectation-maximization algorithm. If we assume that all variables are observed, we can learn the parameters simply by using MLE. The data log-likelihood function is:

$$\begin{aligned}\ell(\theta, D) &= \log \prod_n p(z_n, x_n) = \log \prod_n p(z_n|\pi) p(x_n|z_n, \mu, \sigma) \\ &= \sum_n \log \prod_k \pi_k^{z_n^k} + \sum_n \log \prod_k N(x_n; \mu_k, \sigma)^{z_n^k} \\ &= \sum_n \sum_k z_n^k \log \pi_k - \sum_n \sum_k z_n^k \frac{1}{2\sigma^2}(x_n - \mu_k)^2 + C\end{aligned}$$

C is some constant that we do not bother calculating, since maximizing the log-likelihood is independent of any additive constant. Our maximum likelihood estimates of the parameters are then:

$$\begin{aligned}\hat{\pi}_{k,MLE} &= \arg\max_{\pi} \ell(\theta, D) \\ \hat{\mu}_{k,MLE} &= \arg\max_{\mu} \ell(\theta, D) \\ \hat{\sigma}_{k,MLE} &= \arg\max_{\sigma} \ell(\theta, D)\end{aligned}$$

We can easily put these functions into an optimization algorithm, or in the case of $\hat{\mu}$, obtain a simple closed form:

$$\hat{\mu}_{k,MLE} = \frac{\sum_n z_n^k x_n}{\sum_n z_n^k}$$

However, the above calculations break down in the case of unobserved z_n. For example, it is not possible to use the formula for $\hat{\mu}_{k,MLE}$ since we do not know what values of z_n to use.

5.1 K-means

We can obtain some insight into solving this problem by considering a related algorithm: K-means. The K-means algorithm finds k clusters of data points as follows:

1. Initialize k points ("centers") randomly.
2. Assign each data point to the closest center.
3. Move each center to the mean of all data points assigned to it.
4. Repeat from 2. until converged.

We can view the K-means problem as a graphical model, where the hidden variables is the center each data point is assigned to. More formally, and writing the above in a form that is closer to our Gaussian mixture model, we get:

$$z_n^{(t)} = \arg\max_k \left(x_n - \mu_k^{(t)}\right)^T \Sigma_k^{-1(t)} \left(x_n - \mu_k^{(t)}\right)$$

$$\mu_k^{(t+1)} = \frac{\sum_n \delta\left(z_n^{(t)}, k\right) x_n}{\sum_n \delta\left(z_n^{(t)}, k\right)}$$

Here, z_n are the centers assigned to each data point, and μ_k are the centers.

5.2 The EM Algorithm

Moving closer to the Gaussian mixture model, what if instead of using the mean as centers, we use Gaussian distributions? This way, each center is associated with a covariance matrix as well as a mean. Now, instead of assigning each data point to the closest center, we can do a "soft assignment" to all centers. For example, a data point p can be $\frac{1}{2}$ assigned to center a, $\frac{1}{4}$ center b, and $\frac{1}{4}$ center c. This essentially gives us the EM algorithm for Gaussian mixture models, and iteratively improves the log-likelihood:

$$\begin{aligned}\langle \ell_c(\theta; x, z)\rangle &= \sum_n \langle \log p(z_n|\pi)\rangle_{p(z|x)} + \sum_n \langle \log p(x_n|z_n, \mu, \Sigma)\rangle_{p(z|x)} \\ &= \sum_n \sum_k \langle z_n^k\rangle \log \pi_k - \frac{1}{2}\sum_n \sum_k \langle z_n^k\rangle \left((x_n - \mu_k)^T \Sigma_k^{-1}(x_n - \mu_k) + \log|\Sigma_k| + C\right)\end{aligned}$$

The E step

During the expectation step, we compute the expected values of the sufficient statistics of the hidden variables, given the current estimates of the parameters. In the case of Gaussian mixture models, we compute the following:

$$\tau_n^{k(t)} = \langle z_n^k\rangle_{q^{(t)}} = p\left(z_n^k = 1 | x, \mu^{(t)}, \Sigma^{(t)}\right) = \frac{\pi_k^{(t)} N\left(x_n|\mu_k^{(t)}, \Sigma_k^{(t)}\right)}{\Sigma_i \pi_i^{(t)} N\left(x_n|\mu_i^{(t)}, \Sigma_i^{(t)}\right)}$$

The E step is essentially performing inference with an estimate of the parameters. In k-means, we calculate the hard assignments $z_n^{(t)}$ while in the EM algorithm, we calculate the expected value of a sufficient statistic $\tau_n^{k(t)}$. In the case of Gaussian mixture models, $\tau_n^{k(t)} = \langle z_n^k\rangle_{q^{(t)}}$.

The M step

During the M step, we estimate the parameters with MLE, using the sufficient statistics of the hidden variables we calculated in the E step. In the case of Gaussian mixture models, the sufficient statistic is the expected value of the hidden variables.

$$\pi_k^* = \arg\max_{\pi_k} \langle \ell_c(\theta)\rangle \Longrightarrow \pi_k^{(t+1)} = \frac{\sum_n \langle z_n^k\rangle_{q^{(t)}}}{N} = \frac{\sum_n \tau_n^{k(t)}}{N} = \frac{\langle n_k\rangle}{N}$$

$$\mu_k^* = \arg\max_{\mu_k} \langle \ell_c(\theta)\rangle \Longrightarrow \mu_k^{(t+1)} = \frac{\sum_n \tau_n^{k(t)} x_n}{\sum_n \tau_n^{k(t)}}$$

$$\Sigma_k^* = \arg\max_{\Sigma_k} \langle \ell_c(\theta) \rangle \Longrightarrow \Sigma_k^{(t+1)} = \frac{\sum_n \tau_n^{k(t)} \left(x_n - \mu_k^{(t+1)}\right)\left(x_n - \mu_k^{(t+1)}\right)^T}{\sum_n \tau_n^{k(t)}}$$

In k-means, μ_k^* is calculated as a weighted sum where each weight is 0 or 1, while in EM, it is a proper weighted sum.

5.3 Theory of the EM Algorithm

Supposing that the latent variables z can be observed, we define the *complete log-likelihood* as

$$\ell_c(\theta; x, z) = \log p(x, z|\theta)$$

With observed z, the above can usually be optimized easily. $p(x, z|\theta)$ is the product of many factors, so the log-likelihood function above decomposes into the sum of many factors, where each can be obtimized separately. However, z is not observed, so the log-likelihood cannot be optimized in this way. The likelihood function we are really trying to optimize is the log of a marginal probability, call the *incomplete log-likehihood*:

$$\ell_c(\theta, x) = \log p(x|\theta) = \log \sum_z p(x, z|\theta)$$

Unfortunately, this objective function is a sum inside of a log, which does not decouple as nicely. Instead, we will optimize the following *expected complete log-likelihood*:

$$\langle \ell_c(\theta; x, z) \rangle_q = \sum_z q(z|x, \theta) \log p(x, z|\theta)$$

where q is any arbitrary distribution. Notice, now, that the $\log p(x, z|\theta)$ inside the sum, like the complete log-likelihood function, factors nicely. So, the expected complete log-likelihood function factors, and is easier to optimize. However, it is not clear if maximizing this function actually maximizes the incomplete log-likelihood. As a first step, we can easily show that the expected complete log-likelihood is a lower bound for the incomplete log-likelihood:

$$\begin{aligned}
\ell(\theta; x) &= \log p(x|\theta) \\
&= \log \sum_z p(x, z|\theta) \\
&= \log \sum_z q(z|x) \frac{p(x, z|\theta)}{q(z|x)}
\end{aligned}$$

Using Jensen's inequality,

$$\begin{aligned}
&\geq \sum_z q(z|x) \log \frac{p(x, z|\theta)}{q(z|x)} \\
&= \sum_z q(z|x) \log p(x, z|\theta) - \sum_z q(z|x) \log q(z|x) \\
&= \langle \ell_c(\theta; x, z) \rangle_q + H_q
\end{aligned}$$

Note that the above is the sum of the expected complete log-likelihood and the entropy of the distribution q.

As Coordinate Ascent

The EM algorithm is also in some sense a coordinate descent algorithm. To see this, define the following functional for a fixed data x, called the free energy:

$$F(q,\theta) = \sum_z q(z|x)\log\frac{p(x,z|\theta)}{q(z|x)} \leq \ell(\theta;x)$$

The EM algorithm is then

- **E step**: $q^{t+1} = \arg\max_q F(q,\theta^t)$
- **M step**: $\theta^{t+1} = \arg\max_\theta F(q^{t+1},\theta^t)$

In performing the E step, we obtain the posterior distribution of the latent variables given the data and the parameters, ie.

$$q^{t+1} = \arg\max_q F(q,\theta^t) = p(z|x,\theta^t)$$

To prove this, we just have to show that this value of q^{t+1} attains the bound $\ell(\theta;x) \geq F(q^{t+1},\theta)$.

$$\begin{aligned}
F(p(z|x,\theta^t),\theta^t) &= \sum_z p(z|x,\theta^t)\log\frac{p(x,z|\theta^t)}{p(z|x,\theta^t)} \\
&= \sum_z q(z|x)\log p(x|\theta^t) \\
&= \log p(x|\theta^t) \\
&= \ell(\theta^t;x)
\end{aligned}$$

Without loss of generality, assume that $p(x,z|\theta)$ is a generalized exponential family distribution, so that

$$p(x,z|\theta) = \frac{1}{Z(\theta)}h(x,z)\exp\left\{\sum_i \theta_i f_i(x,z)\right\}$$

Then the complete log likelihood under $q^{t+1} = p(z|x,\theta^t)$ is

$$\begin{aligned}
\langle \ell_c(\theta^t;x,z)\rangle_{p^{t+1}} &= \sum_z q(z|x,\theta^t)\log p(x,z|\theta^t) - A(\theta) \\
&= \sum_i \theta_i^t \langle f_i(x,z)\rangle_{q(z|x,\theta^t)} - A(\theta)
\end{aligned}$$

In the special case of p being a generalized linear model,

$$= \sum_i \theta_i^t \langle \eta_i(z)\rangle_{q(z|x,\theta^t)}\xi_i(x) - A(\theta)$$

Let's now take a look at the M step, and remember that the free energy is the sum of two terms $F(q,\theta) = \langle \ell_c(\theta;x,z)\rangle_q + H_q$. The second term is constant when maximizing with respect to θ, so we need only consider the first term.

$$\theta^{t+1} = \arg\max_\theta \sum_z q(z|x)\log p(x,z|\theta)$$

When q is optimal, this is the same as MLE of the fully observed $p(x,z|\theta)$, but instead of the sufficient statistics for z, their expectations with respect to $p(z|x,\theta)$ are used.

6 Example: Hidden Markov Model

With the EM algorithm, it becomes very straitforward for us to solve some simple graphical models with hidden variables. Let's take the Hidden Markov Model (HMM) as an example.

Generally, the learning paradigms for an HMM model are classfied into the following two categories:

- Supervised Learning: we estimate the model parameters when the 'label' information is provided, i.e., the "right answer" is known. For example, we learn from a genomic region $x = x_1, x_2, \ldots, x_{1000000}$ where we have good annotations of the CpG islands.
- Unsupervised Learning: we estimate the model without label information. For instance, the procupine genome; we don't know how frequent are the CpG islands there, neither do we know their composition, but we still want to learn from the data.

Our goal is to learn from the data using Maximal Likelihood Estimation (MLE). Specifically, we update the model parameters θ by maximizing $p(x|\theta)$.

Now we introduce an algorithm named "Baum-Welch" algorithm, which is a variant of EM algorithm for learning HMM. Recall that the complete log-likelihood of HMM is written as follows:

$$\ell_e(\theta; x, y) = \log p(x, y) = \log \prod_n (p(y_{n,1} \prod_{t=2}^{T} p(y_{n,t}|y_{n,t-1}) \prod_{t=1}^{T} p(x_{n,t}|y_{n,t}))) \tag{1}$$

It can be further written as:

$$\begin{aligned} < \ell_c(\theta; x, y) > = &\sum_n (< Y_{n,1}^i >_{p(Y_{n,1}|x_n)} \log \pi_i) + \sum_n \sum_{t=2}^{T} (< Y_{n,t-1}^i Y_{n,t}^j >_{p(Y_{n,t-1} Y_{n,t}|x_n)} \log a_{i,j}) \\ &+ \sum_n \sum_{t=1}^{T} (x_{n,t}^k < Y_{n,t}^i >_{p(Y_{n,t}|x_n)} \log b_{i,k}) \end{aligned} \tag{2}$$

where we denote $A = \{a_{i,j}\} = p(Y_t = j|Y_{t-1} = i)$ as the transition matrix and $B = \{b_{i,j}\} = p(Y_t = i|x_t = j)$ as the emission matrix.

Accordingly, we derive the EM algorithm of HMM as follows:

- The E step

$$\begin{aligned} \gamma_{n,t}^i &=< Y_{n,t}^i >= p(Y_{n,t}^i = 1|x_n) \\ \xi_{n,t}^{i,j} &=< Y_{n,t-1}^i Y_{n,t}^j >= p(Y_{n,t-1}^i = 1, Y_{n,t}^j = 1|x_n) \end{aligned} \tag{3}$$

- The M step

$$\begin{aligned} \pi_i^{ML} &= \frac{\sum_n \gamma_{n,1}^i}{N} \\ a_{ij}^{ML} &= \frac{\sum_n \sum_{t=2}^{T} \xi_{n,t}^{i,j}}{\sum_n \sum_{t=1}^{T-1} \gamma_{n,t}^i} \\ b_{i,k}^{ML} &= \frac{\sum_n \sum_{t=1}^{T} \gamma_{n,t}^i x_{n,t}^k}{\sum_n \sum_{t=1}^{T-1} \gamma_{n,t}^i} \end{aligned} \tag{4}$$

In the unsupervised learning setting, we are given $x = x_1, \dots, x_N$ while their state path $y = y_1, \dots, y_N$ is not unknown. The EM algorithm changes accordingly:

1. Start with the best guess of parameters θ for the model
2. Estimate $A_{i,j}, B_{i,j}$ in the training data:

$$\begin{aligned} A_{ij} &= \sum_{n,t} < Y^i_{n,t-1} Y^j_{n,t} > \\ B_{ik} &= \sum_{m,t} < Y^i_{n,t} > x^k_{n,t} \end{aligned} \tag{5}$$

3. Update θ according to A_{ij}, B_{jk}. Now, problem becomes supervised learning.
4. Repeat steps 2 and 3 until convergence.

The above algorithm is called the Baum-Welch algorithm.

7 EM for general BNs

Now we demonstrate the EM algorithm for general BNs:

```
while not converged do
    for each node i do
        $ESS_i = 0$
    end
    for each data sample n do
        do inference with $x_{n,H}$ for each node i do
            $ESS_{i+} =< SS_i(x_{n,i}, x_{n,\pi_i}) >_{p(x_{n,H}|x_{n,-H})}$
        end
    end
    for each node i do
        $\theta_i = MLE(ESS_i)$
    end
end
```

8 Summary: EM algorithm

In summary, EM algorithm can be regarded as a way of maximizing likelihood function for latent variable models. It finds MLE of parameters when the original (hard) problem can be broken up into two (easy) pieces:

- Estimate some missing or unobserved data from observed data and current parameters
- Using this complete data, find the maximum likelihood parameter estimates

The EM algorithm alternates between filling in the latent variables using the best guess and updating the parameters based on this guess. Specifically, in the M-step we optimize a lower bound on the likelihood. In the E-step we close the gap, making the bound equals the likelihood.

9 Conditional Mixture Model and EM Algorithm

Consider the following situation: we will model $p(Y|X)$ using different experts, each responsible for different regions of the input space. We have a latent variable Z, which chooses expert using softmax gating function:

$$p(z^k = 1|x) = Softmax(\xi^T x) \tag{6}$$

While, each expert can be a linear regression model (or more complex models):

$$p(y|x, z^k = 1) = N(y; \theta_k^T x, \sigma_k^2) \tag{7}$$

The posterior expert responsibilities are:

$$p(z^k = 1|x, y, \theta) = \frac{p(z^k = 1|x) p_k(y|x, \theta_k, \sigma_k^2)}{\sum_j p(z^j = 1|x) p_j(y|x, \theta_j, \sigma_j^2)} \tag{8}$$

Now we derive the EM for conditional mixture model:
The objective function is as follows:

$$\begin{aligned} < \ell_c(\theta; x, y, z) > &= \sum_n < \log p(z_n|x_n, \xi) >_{p(z|x,y)} + \sum_n < \log p(y_n|x_n, z_n, \theta, \sigma) >_{p(z|x,y)} \\ &= \sum_n \sum_k < z_n^k > \log(softmax(\xi_k^T x_n)) - \frac{1}{2} \sum_n \sum_k < z_n^k > (\frac{(y_n - \theta_k^T x_n)}{\sigma_k^2} + \log \sigma_k^2 + C) \end{aligned} \tag{9}$$

The EM algorithm:

- E-step:

$$\tau_n^{k(t)} = p(z_n^k = 1|x_n, y_n, \theta) = \frac{p(z_n^k = 1|x_n) p_k(y_n|x_n, \theta_k, \sigma_k^2)}{\sum_j p(z_n^j = 1|x_n) p_j(y_n|x_n, \theta_j, \sigma_j^2)} \tag{10}$$

- M-step:
  - We use the normal equation for standard linear regression: $\theta = (X^T X)^{-1} X^T Y$, but with the data reweighted by τ
  - We can also use IRLS or reweighted IRLS to update $\{\xi_k, \theta_k, \sigma_k\}$ based on data pair (x_n, y_n), with weights $\tau_n^{k(t)}$.

10 Hierarchical Mixture of experts

The hierarchial mixture of experts is like a soft version of a depth-2 classification/regression tree. In the model, the $P(Y|X, G_1, G_2)$ can be modeled as a GLIM, with parameters dependent on the values of G_1 and G_2, which specify a "conditional path" to a given leaf in the tree.

11 Mixture of overlapping experts

Different from the typical mixture of experts, by removing the link between X and Z in the graphical model of mixture of experts, we can make the partitions independent of the input, thus allowing overlapping. Accordingly, this leads to a mixture of linear regressors, in which each subpopulation has a different conditional mean:

$$p(z^k = 1|x, y, \theta) = \frac{p(z^k = 1|x) p_k(y|x, \theta_k \sigma_k^2)}{\sum_j p(z^j = 1|x) p_j(y|x, \theta_j, \sigma_j^2)} \tag{11}$$

12 Partially Hidden Data

We can also learn the model parameters when there are missing(hidden) variables. In this case, the cost function becomes:

$$\ell_c(\theta; D) = \sum_{n \in Complete} \log p(x_n, y_n|\theta) + \sum_{m \in Missing} \log \sum_{y_n} p(x_m, y_m|\theta) \tag{12}$$

Now we can think of this in a new way: in the E-step we estimate the hidden variables on the incomplete cases only. Then, in the M-step we optimize the log likelihood on the complete data plus the expected likelihood on the incomplete data using the E-step.

13 EM Variants

In this section, we briefly introduce two variants of the basic EM algorithm: The first one is the sparse EM. It does not re-compute exactly the posterior probability on each data point under all models, because it is almost zero. Instead, it keeps an active list which you update every once in a while. The second one is the generalized (incomplete) EM. Sometimes we may find that it's very difficult to find the ML parameters in the M-step, even given the completed data. However, we can still make progress by doing an M-step that improves the likelihood a bit. This is in the line of the IRLS step in the mixture of experts models.

14 A Report Card for EM

EM is one of the most popular methods to get the MLE or MAP estimation for a model with latent variables. The advantages of EM are listed as follows.

- no learning rate (step-size) parameter
- automatically enforces parameter constraints
- very fast for low dimensions
- each iteration guaranteed to improve likelihood

While, the disadvantages of EM are

- can get stuck in local minima
- can be slower than conjugate gradient (especially near convergence)
- requires expensive inference step
- is a maximum liklihood/MAP method

There are still some modern developments upon traditional EM, such as convex relaxation, direct non-convex optimization.