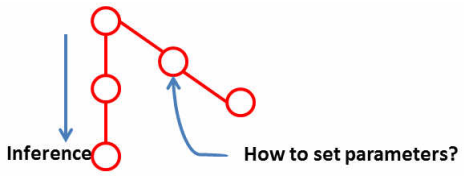
CSC 446 Notes: Lecture 11

Typed by Xiang Xiang

February 27, 2012

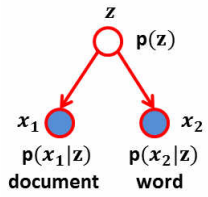
1 Preface

In last lecture, we introduced Tree Decomposition. Till now, we have covered a lot as regards how to do inference in a graphical model. In this lecture, we will move back to the learning part. We will consider how to set parameters for the variables.



2 Parameter Setting: An Example

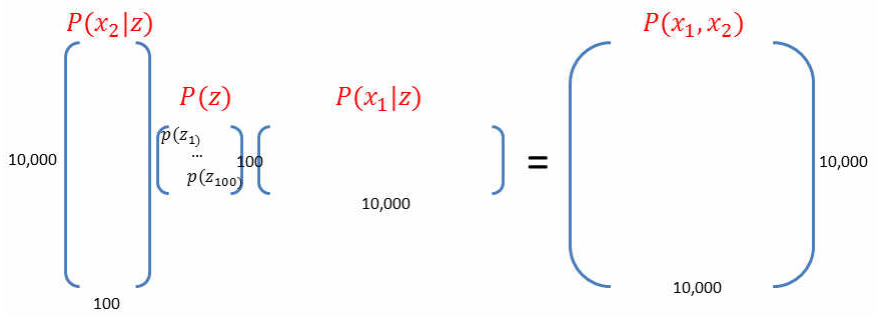
In the discrete case, we set the parameters just by counting how often each variable occurs. However, we may not know the value of some variables. Thus, in the following, we will discuss learning with hidden (latent) variables. The simplest model is shown below. This model has been used in the Aspect Model for



probabilistic Latent Semantic Analysis (pLSA). The variable's value is called an aspect. pLSA has been widely used in information retrieval. In this example, let x_1 and x_2 respectively denote the document ID and word ID. Then, there is a sequence of pair (x_1, x_2), e.g., (1, "the"), (1, "green"), ..., (1000, "the"). In this context, we may have various tasks, e.g., to find the words which co-occur, to find the words on the same topic, or to find the documents containing the same words.

Now, we will introduce the term *cluster*. The reasons why we need the cluster representation are as followed: (1) There are a large amount (e.g., 10,000) of documents, each of which is formed of a large amount (e.g., 10,000) of words. Without a cluster representation, we have to handle a huge query table with too many (say, $10,000^2$) entries. That makes any query difficult. (2) If we still set the parameters just by counting how often each variable occurs, then there is a risk of over-fitting the individuals (i.e., the pair of (document, word)). Because of them, we need to do something smart: clustering. Recall the graphical model displayed above, the hidden (latent) variable z is just the cluster ID.

Note that $x_1 \perp x_2 \| z$. Therefore, the joint distribution $p(x_1, x_2) = \sum_z p(z)p(x_1|z)p(x_2|z)$. As shown in the figure below, now we will not directly compute each entry to obtain the $10,000 \times 10,000$ query table $P(x_1, x_2)$. Instead, we maintain low-rank matrices $P(z)$, $P(x_1|z)$ and $P(x_2|z)$.



Now, we have a set of observed variables $\mathbf{X} = \{(x_1^1, x_2^1), (x_1^2, x_2^2), ...\}$, a set of hidden variables $\mathbf{Z} = \{z_1, z_2, ...\}$ and a set of parameters $\theta = \{\theta_z, \theta_{x_1|z}, \theta_{x_2|z}\}$. Note that x_1^i are i.i.d. variables, and the same for x_2^i. To choose θ, we maximize the likelihoods (MLE): $\max_\theta P(\mathbf{X}; \theta)$.

$$\theta = \underset{\theta}{\operatorname{argmax}} \prod_n P_\theta(x_1^n, x_2^n) = \underset{\theta}{\operatorname{argmax}} \prod_n \sum_z p(z)p(x_1^n|z)p(x_2^n|z) = \underset{\theta}{\operatorname{argmax}} \sum_n \log \sum_z p(z)p(x_1^n|z)p(x_2^n|z) \tag{1}$$

If there is no hidden variable z, we will just count, instead of summing over z. However now, we need to sum over z and find the maximum of the above objective function, which is not a closed-form expression. Thus, it is not feasible to directly set the derivative to zero. To solve this tough optimization problem, we will introduce the Expectation-Maximization (EM) algorithm.

3 Expectation-Maximization Algorithm

The EM algorithm is an elegant and powerful method for finding maximum likelihood solutions for models with hidden (latent) variables. It breaks down the potentially tough problem of MLE into two stages. The basic idea is shown below.

E-step:
Guess $\mathbf{z}$
M-step:
MLE to fit θ to $\mathbf{X}, \mathbf{Z}$

Following the above example, we present the EM algorithm in detail.

REPEAT
E-step:
for $i = 1 \dots N$
for $z = 1 \dots K$
$p(z, n) = p_{\theta_t}(z) \cdot p_{\theta_t}(x_1^n|z) \cdot p_{\theta_t}(X_2^n|z)$
$sum \mathrel{+}= p(z, n)$
for z
$p(z, n) \leftarrow \frac{p(z,n)}{sum}$
(An alternative:
$ec(z) \mathrel{+}= p(z, n)$
$ec(z, x_1^n) \mathrel{+}= p(z, n)$
$ec(z, x_2^n \mathrel{+}= p(z, n)$)

M-step:
for z
$ec(z) = \sum_1^N p(z, n)$

$\theta_{t+1} \leftarrow \frac{ec(z)}{N}$

for z, x_1

$ec(z, x) = \sum_1^N I(x_1^n = x_1) p(z, n)$

$\theta_{x_1|z} = \frac{\theta(z,x_1)}{ec(z)}$

for z, x_2

$ec(z, x_2) = \sum_1^N I(x_2^n = x_2) p(z, n)$

$\theta_{x_2|z} = \frac{\theta(z,x_2)}{ec(z)}$

UNTIL convergence

where *sum* is for normalization and $ec(\cdot)$ denotes the expected count, which is not a real count but an average on what we think z is. Namely, this count is probabilistic. The intuition is to assign some credit to each possible value. Also note that $I(\cdot)$ is an indicator function (return 1/0 if the condition is true/fasle). In the following, we will derive how to approximate the maximum of the likelihood by maximizing the joint probability's log likelihood iteratively through E-M steps. For the example present in Sec.2, now let us go further using the same formulation with Eqn. (1).

$$
\begin{aligned}
&\theta = \text{argmax}_\theta \, Q(\theta; \theta^{old}) \\
&= \text{argmax}_\theta \, E_{p(z|x,\theta^{old})} \log p(\mathbf{X}, \mathbf{Z}) \\
&= \text{argmax}_\theta \, E_{p(z|x,\theta^{old})} \big[\log \textstyle\prod_n p(x_1^n, x_2^n, z^n) \big] \\
&= \text{argmax}_\theta \, E \big[\log \textstyle\prod_n p(z^n) \cdot p(x_1^n|z^n) \cdot p(x_2^n|z^n) \big] \\
&= \text{argmax}_\theta \, E \big[\textstyle\sum_n \log p(z^n) + \sum_n \log p(x_1^n|z^n) + \sum_n \log p(x_2^n|z^n) \big] \\
&\Big(\quad = \text{argmax}_\theta \, E \big[\textstyle\sum_n log\theta_{z^n} + \sum_n \log \theta_{x_1^n|z^n} + \sum_n \log \theta_{x_2^n|z^n} \big] \quad \Big) \\
&= \text{argmax}_\theta \, E \big[\textstyle\sum_k \sum_n I(z^n = k) \log p(z = k) + \sum_k \sum_n I(x_1^n = x_1 | z^n = k) \log p(x_1 | z = k) \\
&\qquad\qquad + \sum_k \sum_n I(x_2^n = x_2 | z^n = k) \log p(x_2 | z = k) \big] \\
&\big(sum \quad over \quad values \quad \text{z} \quad can \quad take \quad and \quad individually \quad group \quad together \big) \\
&= \text{argmax}_\theta \, E \big[\textstyle\sum_k c(z = k) \log p(z = k) + \sum_k c(x_1, z = k) \log p(x_1 | z = k) + \sum_k c(x_2, z = k) \log p(x_2 | z = k) \big] \\
&\Big(\quad = \text{argmax}_\theta \textstyle\sum_k E\big[c(z = k)\big] \log \theta_{z=k} + \sum_k E\big[c(x_1, z = k)\big] \log \theta_{x_1|z=k} + \sum_k E\big[c(x_2, z = k)\big] \log \theta_{x_2|z=k} \\
&\qquad = \text{argmax}_\theta \textstyle\sum_k ec(z) \log \theta_{z=k} + \sum_k ec(x_1, z) \log \theta_{x_1|z=k} + \sum_k ec(x_2, z) \log \theta_{x_2|z=k} \quad \Big)
\end{aligned}
\tag{2}
$$

Therefore, $\theta = \frac{1}{sum_0} ec(z)$, $\theta_{x_1|z} = \frac{1}{sum_1} ec(x_1, z)$, and $\theta_{x_2|z} = \frac{1}{sum_2} ec(x_2, z)$, but make sure that normalization is done (sum to 1). Notably, $c(\cdot)$ denotes the count and $ec(\cdot)$ denotes the expected count. Also note that $E\big[c(z = k)\big] = ec(z)$ which we have mentioned in the EM algorithm flow, and similar for $ec(x_1, z)$ and $ec(x_2, z)$.

4 EM Algorithm in General

Now, we will give a general derivation for the EM algorithm. The denotation will be the same with the above. Similarly, we have $\theta = \text{argmax}_\theta \, Q(\theta; \theta^{old}) = \text{argmax}_\theta \, E_{p(z|x,\theta^{old})} \log p(\mathbf{X}, \mathbf{Z})$. Let us focus on the objective function Q.

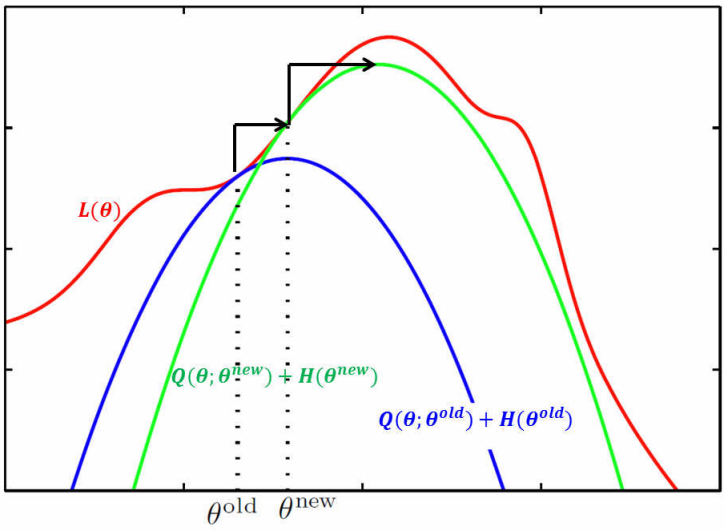
$$
\begin{aligned}
&Q(\theta; \theta^{old}) \\
&= E_{p(z|x,\theta^{old})} \log p(\mathbf{X}, \mathbf{Z}) \\
&= E_{p(z|x,\theta^{old})} \log \big[p(\mathbf{Z}|\mathbf{X}) \cdot p(\mathbf{X}) \big] \\
&= E_{p(z|x,\theta^{old})} \big[\log p(\mathbf{Z}|\mathbf{X}) + \log p(\mathbf{X}) \big] \\
&= E\Big[\log \tfrac{p(\mathbf{Z}|\mathbf{X})}{p_{\theta^{old}}(\mathbf{Z}|\mathbf{X})} \cdot p_{\theta^{old}}(\mathbf{Z}|\mathbf{X}) \Big] + \log p(\mathbf{X}) \\
&\big(make \quad it \quad look \quad like \quad K - L \quad divergence \big) \\
&= E\Big[- \log \tfrac{p_{\theta^{old}}(\mathbf{Z}|\mathbf{X})}{p(\mathbf{Z}|\mathbf{X})} \Big] - E\big[- \log p_{\theta^{old}}(\mathbf{Z}|\mathbf{X}) \big] + \log p(\mathbf{X}) \\
&= -D\Big(\mathbf{Z}|\mathbf{X}, \theta^{old} \Big\| \mathbf{Z}|\mathbf{X}, \theta \Big) - H\Big(\mathbf{Z}|\mathbf{X}, \theta^{old} \Big) + L(\theta)
\end{aligned}
\tag{3}
$$

where D, H, L are respectively the K-L divergence, the entropy and the likelihood. Note that our objective is to maximize the likelihood $\log p(\mathbf{X})$. It does not have Z inside, so it can be put out of $E(\cdot)$. Now, we

write down $L(\theta)$ with simplified notations:

$$L(\theta) = Q(\theta; \theta^{old}) + H(\theta^{old}) + D(\theta^{old} \| \theta) \tag{4}$$

where Q, H and D are all dynamic functions. The approximation can be illustrated in the space of parameter θ, as shown schematically in the figure below. Here the red curve depicts $L(\theta)$ (incomplete data) which we



wish to maximize. We start with some initial parameter value θ^{old}, and in the first E step we evaluate the distribution of $Q(\theta; \theta^{old}) + H(\theta^{old})$, as shown by the blue curve. Since the K-L divergence $D(\theta^{old} \| \theta)$ is always positive, the blue curve gives a lower bound to the red curve $L(\theta)$. And $D(\theta^{old} \| \theta)$ just gives the gap between the two curves. Note that the bound makes a tangent contact with $L(\theta)$ at θ^{old}, so that both curves have the same gradient and $D(\theta^{old} \| \theta^{old}) = 0$. Thus, $L(\theta^{old}) = Q(\theta^{old}; \theta^{old}) + H(\theta^{old})$. Besides, the bound is a convex function having a unique maximum at $\theta^{new} = \text{argmax}_\theta \left[Q(\theta; \theta^{old}) \right]$. In the M step, the bound is maximized giving the value θ^{new}, which also gives a larger value of $L(\theta)$ than θ^{old}: $L(\theta^{new}) = Q(\theta^{new}; \theta^{old}) + H(\theta^{old}) + D(\theta^{old} \| \theta^{new})$. In practice, during the beginning iterations, this point is usually still far away from the maximum of $L(\theta)$. However, if we run one more iteration, the result will get better. The subsequent E step then constructs a bound that is tangential at θ^{new} as shown by the green curve. Iteratively, the maximum will be accessed in the end, in a manner kind of similar with gradient ascent. In short, there are two properties for the EM algorithm: (1) the performance gets better step by step. (2) it will converge. At last, it should also be emphasized that EM is not guaranteed to find the global maximum, for there will generally be multiple local maxima. Being sensitive to the initialization, EM may not find the largest of these maxima.

In this lecture, we quickly go through the details of the EM algorithm, which maximizes L through maximizing Q at each iteration step. Then, the remaining problems are how to compute Q, and exactly how to compute $p(\mathbf{Z}|\mathbf{X}, \theta^{old})$.