Chapter 19: Partially Observed Data

Tal Daniel Gerbi & Shai Kazaz

Introduction

In this chapter we will deal with Parameter Estimation (as in chapter 17), and with Structure Learning (as in chapter 18).

In both chapters we had observations and we did some learning using this data. Now the data we hold is not complete.

As we will see it’s make the hardness of the problems even harder. In many real life cases we need to know how to handle these problems.

Parameter Estimation

The Unobservable Case

Parameter Estimation Remainder

Given a network structure G and it CPDs, we wanted to define $P(X|\theta)$.

Given a set of data $D = \{X[1], \ldots X[m]\}$ (samples/observations) we defined the maximum likelihood function to be:

$$L(\theta : D) = P(D \mid \theta) \underset{iid}{=} \prod_i P(x[i] \mid \theta)$$

And we looked for the Maximum Likelihood Estimator (MLE), $\hat{\theta}$, to be:

$$\hat{\theta} = \operatorname{argmax}_{\theta} l(\theta : D)$$

Where $l(\theta : D) = \log(L(\theta : D))$.

Hidden Variables

As before, given a network structure G over a set of variables X, also we have a set D of samples, but this time for each sample i we denote $h(i)$ to be a sub-set of the sample witch we do not observe .

We denote $H = \bigcup_i h(i)$ possible assignment to all of the missing values in the data set.

The likelihood function is set to be:

$L(\theta : D) = P(D \mid \theta) \overset{*}{=} \sum_H P(D, H \mid \theta)$. (* is true, due to assignment for all missing data)

Unfortunately, the number of possible assignments in this sum is exponential in the number of missing values in the entire data set.

Hidden Variables

To be more accurate, for a sample i , we define $o[i]$ the values of the sample that we do observe, now the maximum likelihood can written as (assuming iid):

$$L(\theta : D) = \prod_i P(o[i] | \theta) = \prod_i \sum_{h[i]} P(o[i], h[i] | \theta)$$

Main Issue

If we wish the evaluate the MLE, $\hat{\theta} = \arg\max_{\theta} l(\theta : D)$, as we discussed, in the presence of incomplete data, the likelihood does not decompose. And so the problem requires optimizing a highly nonlinear and multimodal function over a high-dimensional space (one consisting of parameter assignments to all CPDs).

The Methods

There are two main classes of methods for performing this optimization:

- A generic nonconvex optimization algorithm, such as Gradient Ascent.

- *Expectation Maximization*, a more specialized approach for optimizing likelihood functions.

Gradient Ascent

The idea: “Going up the hill”, at each point we shell go in the direction where the function grows.

In this approach the main technical question is how to calculate the gradient.

We shell see how to use the divertive single CPD entry $P(x|u)$ for our goal.

Lemma

Let $\mathcal{B}$ be a Bayesian network with structure $\mathcal{G}$ over $\mathcal{X}$ that induces a probability distribution P, let $\boldsymbol{o}$ be a tuple of obserations for some of the variables, and let $X \in \mathcal{X}$ be some random variable. Then

$$\frac{\partial}{\partial P(x \mid \boldsymbol{u})} P(\boldsymbol{o}) = \frac{1}{P(x \mid \boldsymbol{u})} P(x, \boldsymbol{u}, \boldsymbol{o})$$

if $P(x \mid \boldsymbol{u}) > 0$, where $x \in Val(X)$, $\boldsymbol{u} \in Val(\mathrm{Pa}_X)$.

Result From The Lemma

If the observation o do not agree with x or u then $P(x,u,o)=0$, thus the gradient is 0.

Theorem

Let $\mathcal{G}$ be a Bayesian network structure over $\mathcal{X}$, and let $\mathcal{D} = \{\boldsymbol{o}[1], \ldots, \boldsymbol{o}[M]\}$ be a partially observable data set. Let X be a variable and $\boldsymbol{U}$ its parents in $\mathcal{G}$. Then

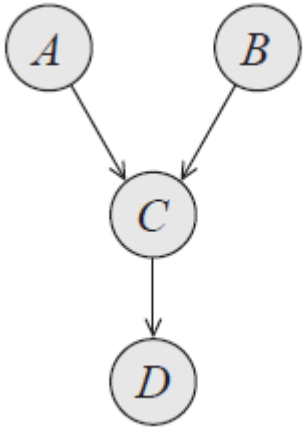
$$\frac{\partial \ell(\boldsymbol{\theta} : \mathcal{D})}{\partial P(x \mid \boldsymbol{u})} = \frac{1}{P(x \mid \boldsymbol{u})} \sum_{m=1}^{M} P(x, \boldsymbol{u} \mid \boldsymbol{o}[m], \boldsymbol{\theta}).$$

Conclusion

This theorem provides the form of the gradient for table-CPDs. For other CPDs, such as noisy-or CPDs, we can use the *chain rule of derivatives* to compute the gradient. Suppose that the CPD entries of $P(X \mid \boldsymbol{U})$ are written as functions of some set of parameters $\boldsymbol{\theta}$. Then, for a specific parameter $\theta \in \boldsymbol{\theta}$, we have

$$\frac{\partial \ell(\boldsymbol{\theta} : \mathcal{D})}{\partial \theta} = \sum_{x, \boldsymbol{u}} \frac{\partial \ell(\boldsymbol{\theta} : \mathcal{D})}{\partial P(x \mid \boldsymbol{u})} \frac{\partial P(x \mid \boldsymbol{u})}{\partial \theta},$$

Example



Looking on this network, witch all the variables are binary.

Let us compute gradient of one family of parameters $P(D|C)$

by using observation $o = \langle a^1, ?, ?, d^0 \rangle$. From the previous theorem we have

$\frac{\partial \log P(o)}{\partial P(d^0 \mid c^0)} = \frac{P(d^0, c^0 \mid o)}{P(d^0 \mid c^0)}$ (C is D's father). Assume the current θ is:

$$
\begin{aligned}
\theta_{a^1} &= 0.3 \\
\theta_{b^1} &= 0.9 \\
\theta_{c^1|a^0,b^0} &= 0.83 \\
\theta_{c^1|a^0,b^1} &= 0.09 \\
\theta_{c^1|a^1,b^0} &= 0.6 \\
\theta_{c^1|a^1,b^1} &= 0.2 \\
\theta_{d^1|c^0} &= 0.1 \\
\theta_{d^1|c^1} &= 0.8.
\end{aligned}
$$

So the probabilities are (with respect to o):

$$
\begin{aligned}
P(\langle a^1, b^1, c^1, d^0 \rangle) &= 0.3 \cdot 0.9 \cdot 0.2 \cdot 0.2 = 0.0108 \\
P(\langle a^1, b^1, c^0, d^0 \rangle) &= 0.3 \cdot 0.9 \cdot 0.8 \cdot 0.9 = 0.1944 \\
P(\langle a^1, b^0, c^1, d^0 \rangle) &= 0.3 \cdot 0.1 \cdot 0.6 \cdot 0.2 = 0.0036 \\
P(\langle a^1, b^0, c^0, d^0 \rangle) &= 0.3 \cdot 0.1 \cdot 0.4 \cdot 0.9 = 0.0108.
\end{aligned}
$$

Thus, the posterior probabilities are:

$$
\begin{aligned}
P(\langle a^1, b^1, c^1, d^0 \rangle \mid \boldsymbol{o}) &= 0.0492 \\
P(\langle a^1, b^1, c^0, d^0 \rangle \mid \boldsymbol{o}) &= 0.8852 \\
P(\langle a^1, b^0, c^1, d^0 \rangle \mid \boldsymbol{o}) &= 0.0164 \\
P(\langle a^1, b^0, c^0, d^0 \rangle \mid \boldsymbol{o}) &= 0.0492.
\end{aligned}
$$

(for example: $P(\langle a^1, b^1, c^1, d^0 \rangle \mid o) = \frac{P(\langle a^1, b^1, c^1, d^0 \rangle, o)}{P(o)} = \frac{P(\langle a^1, b^1, c^1, d^0 \rangle)}{\sum_{B,C} P(\langle a^1, B, C, d^0 \rangle)}$)

So we have:

$$
\begin{aligned}
\frac{\partial \log P(\boldsymbol{o})}{\partial P(d^1 \mid c^0)} &= \frac{P(d^1, c^0 \mid \boldsymbol{o})}{P(d^1 \mid c^0)} = \frac{0}{0.1} = 0 \\
\frac{\partial \log P(\boldsymbol{o})}{\partial P(d^0 \mid c^0)} &= \frac{P(d^0, c^0 \mid \boldsymbol{o})}{P(d^0 \mid c^0)} = \frac{0.8852 + 0.0492}{0.9} = 1.0382 \\
\frac{\partial \log P(\boldsymbol{o})}{\partial P(d^1 \mid c^1)} &= \frac{P(d^1, c^1 \mid \boldsymbol{o})}{P(d^1 \mid c^1)} = \frac{0}{0.8} = 0 \\
\frac{\partial \log P(\boldsymbol{o})}{\partial P(d^0 \mid c^1)} &= \frac{P(d^0, c^1 \mid \boldsymbol{o})}{P(d^0 \mid c^1)} = \frac{0.0492 + 0.0164}{0.2} = 0.328.
\end{aligned}
$$

We can see that in order to observe more observations from o we can achieve this goal by increasing both $P(d^0, c^0), P(d^0, c^1)$, and to give more weight to former will lead to a better results.

Now for observation $o' = \langle a^0, ?, ?, d^1 \rangle$, by repeating this process we will get:

$$\frac{\partial \log P(o')}{\partial P(d^1 \mid c^0)} = \frac{P(d^1, c^0 \mid o')}{P(d^1 \mid c^0)} = \frac{0.2836}{0.1} = 2.8358$$

$$\frac{\partial \log P(o')}{\partial P(d^0 \mid c^0)} = \frac{P(d^0, c^0 \mid o')}{P(d^0 \mid c^0)} = \frac{0}{0.9} = 0$$

$$\frac{\partial \log P(o')}{\partial P(d^1 \mid c^1)} = \frac{P(d^1, c^1 \mid o')}{P(d^1 \mid c^1)} = \frac{0.7164}{0.8} = 0.8955$$

$$\frac{\partial \log P(o')}{\partial P(d^0 \mid c^1)} = \frac{P(d^0, c^1 \mid o')}{P(d^0 \mid c^1)} = \frac{0}{0.2} = 0.$$

Suppose we have only these two observations, then the gradient of the log-likelihood function is the sum of the gradient with respect to the two observations. We get that:

$$\frac{\partial \ell(\boldsymbol{\theta} : \mathcal{D})}{\partial P(d^1 \mid c^0)} = 2.8358$$

$$\frac{\partial \ell(\boldsymbol{\theta} : \mathcal{D})}{\partial P(d^0 \mid c^0)} = 1.0382$$

$$\frac{\partial \ell(\boldsymbol{\theta} : \mathcal{D})}{\partial P(d^1 \mid c^1)} = 0.8955$$

$$\frac{\partial \ell(\boldsymbol{\theta} : \mathcal{D})}{\partial P(d^0 \mid c^1)} = 0.328.$$

Note that all the gradients are non-negative, so increasing any of the parameters CPD $P(D|C)$ will increase the likelihood of the data, but we cannot increase both $P(d^0 \mid c^0)$ and $P(d^1 \mid c^0)$ at the same time.

We can deal with this problem by defining:

$$P(d^1 \mid c^0) = \theta_{d^1|c^0} \qquad P(d^0 \mid c^0) = 1 - \theta_{d^1|c^0}.$$

And now we can compare them by (using the chain rule on conditional probabilities):

$$\begin{aligned}
\frac{\partial \ell(\boldsymbol{\theta} : \mathcal{D})}{\partial \theta_{d^1|c^0}} &= \frac{\partial P(d^1 \mid c^0)}{\partial \theta_{d^1|c^0}} \frac{\partial \ell(\boldsymbol{\theta} : \mathcal{D})}{\partial P(d^1 \mid c^0)} + \frac{\partial P(d^0 \mid c^0)}{\partial \theta_{d^1|c^0}} \frac{\partial \ell(\boldsymbol{\theta} : \mathcal{D})}{\partial P(d^0 \mid c^0)} \\
&= \frac{\partial \ell(\boldsymbol{\theta} : \mathcal{D})}{\partial P(d^1 \mid c^0)} - \frac{\partial \ell(\boldsymbol{\theta} : \mathcal{D})}{\partial P(d^0 \mid c^0)} \\
&= 2.8358 - 1.0382 = 1.7976.
\end{aligned}$$

Thus we prefer to increase $P(d^1 \mid c^0)$ witch will lead to more observations of o'.

CT-Calibration Remainder (In Short)

We saw in chapter 10, if we want to compute the probability on ALL the variables in a network, using the same message for all the network, we can use CT-Calibration for this purpose.

Algorithm 19.1 Computing the gradient in a network with table-CPDs

Procedure Compute-Gradient (
  $\mathcal{G}$, // Bayesian network structure over $X_1, \ldots, X_n$
  $\boldsymbol{\theta}$, // Set of parameters for $\mathcal{G}$
  $\mathcal{D}$ // Partially observed data set
)

1 // Initialize data structures
2 **for** each $i = 1, \ldots, n$
3 **for** each $x_i, \boldsymbol{u}_i \in Val(X_i, \mathrm{Pa}_{X_i}^{\mathcal{G}})$
4 $\bar{M}[x_i, \boldsymbol{u}_i] \leftarrow 0$
5 // Collect probabilities from all instances
6 **for** each $m = 1 \ldots M$
7 Run clique tree calibration on $\langle \mathcal{G}, \boldsymbol{\theta} \rangle$ using evidence $\boldsymbol{o}[m]$
8 **for** each $i = 1, \ldots, n$
9 **for** each $x_i, \boldsymbol{u}_i \in Val(X_i, \mathrm{Pa}_{X_i}^{\mathcal{G}})$
10 $\bar{M}[x_i, \boldsymbol{u}_i] \leftarrow \bar{M}[x_i, \boldsymbol{u}_i] + P(x_i, \boldsymbol{u}_i \mid \boldsymbol{o}[m])$
11 // Compute components of the gradient vector
12 **for** each $i = 1, \ldots, n$
13 **for** each $x_i, \boldsymbol{u}_i \in Val(X_i, \mathrm{Pa}_{X_i}^{\mathcal{G}})$
14 $\delta_{x_i|\boldsymbol{u}_i} \leftarrow \frac{1}{\theta_{x_i|\boldsymbol{u}_i}} \bar{M}[x_i, \boldsymbol{u}_i]$
15 **return** $\{\delta_{x_i,|\boldsymbol{u}_i} : \forall i = 1, \ldots, n, \forall (x_i, \boldsymbol{u}_i) \in Val(X_i, \mathrm{Pa}_{X_i}^{\mathcal{G}})\}$

Remarks On The Algorithm

As we did in the example, for the general case, using the result:

$$\frac{\partial \ell(\boldsymbol{\theta} : \mathcal{D})}{\partial \theta} = \sum_{x,\boldsymbol{u}} \frac{\partial \ell(\boldsymbol{\theta} : \mathcal{D})}{\partial P(x \mid \boldsymbol{u})} \frac{\partial P(x \mid \boldsymbol{u})}{\partial \theta}, \quad \text{and} \quad \frac{\partial \ell(\boldsymbol{\theta} : \mathcal{D})}{\partial P(x \mid \boldsymbol{u})} = \frac{1}{P(x \mid \boldsymbol{u})} \sum_{m=1}^{M} P(x, \boldsymbol{u} \mid \boldsymbol{o}[m], \boldsymbol{\theta}).$$

We need to compute $P(X[m],U[m]|o[m],\theta)$, for each sample we have, here we use CT-Calibration, since the family preservation property guarantees that X and its parents U will be together in some clique in the tree.

Another Remark…

We should notice (as in the example) that all the gradients are non-negative (as expected), but we cant move in ALL directions, don’t forget all the probabilities should be non-negative and should sum up to one. In the book there are 3 ways to deal with it, I choose to add constraints and using Lagrange-Multiplayers method (this is one of the 3 methods mentioned in the book).

The Bad News… ☹

In most missing value problems, the likelihood function has many local maxima. Unfortunately, gradient ascent procedures are guaranteed to achieve only a local maximum of the function.

There are still ways to deal with this problem, and many methods exist, actually most of non-convex optimization problem deal with this issue.

Expectation Maximization (EM)

Recall we wish to maximize some parameters using the statics on the data we observe, but the problem is that some data is missing.

For example think on the simple network $X \to Y$ and we see a sample $(?, y^1)$, what we should do? Should we refer it to $M[x^1, y^1]$ or to refer it to $M[x^0, y^1]$?

Few approaches raised, like fill the missing data randomly, this will have a bias terms and will effect the learning procedure.

The Main Issue

Actually we have a “chicken and egg” problem:

If we “take care” of the data first this will lead to bad learning on the parameters.

If we wish to learn the parameters on the missing data, the problem is that some of the observations we see conditionally independent on the data we don’t see, again this will lead to bad learning.

The Idea Behind EM

The EM algorithm use the Bootstrap method:

We can start from choice of parameters, or from initial assignment on the missing data. Let say we start with some choice of parameters.

The EM algorithm then fill the missing data using these parameters, then learn new parameters from this data, and so on….

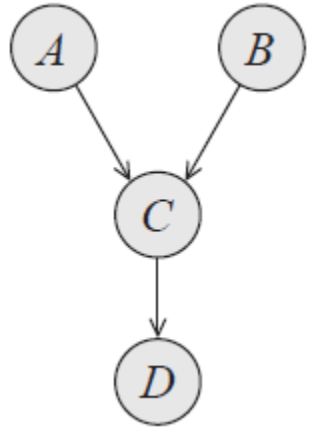
Is It A Good Idea?

If in each step we improve the likelihood function using this parameters, then it would be safe to say that it is ☺ (really???)

Let’s try to achieve this result!

But first, example…

Return To Our Example



We had the observation $o = \langle a^1, ?, ?, d^0 \rangle$.

Our maximum likelihood for the parameter $\hat{\theta}_{d^1|c^0}$ is :

$$\hat{\theta}_{d^1|c^0} = \frac{M[d^1, c^0]}{M[c^0]} = \frac{\sum_{m=1}^{M} \mathbb{1}\{\xi[m]\langle D, C\rangle = \langle d^1, c^0\rangle\}}{\sum_{m=1}^{M} \mathbb{1}\{\xi[m]\langle C\rangle = c^0\}}$$ where $\xi[m]$ is the m'th

sample. So for our observation we have four possible assignment for B,C.

We now define the distribution $Q(B, C) = P(B, C \mid o, \theta)$ where the posterior parameters are:

(as before)

$$\begin{aligned}
\theta_{a^1} &= 0.3 & \theta_{b^1} &= 0.9 \\
\theta_{d^1|c^0} &= 0.1 & \theta_{d^1|c^1} &= 0.8 \\
\theta_{c^1|a^0,b^0} &= 0.83 & \theta_{c^1|a^1,b^0} &= 0.6 \\
\theta_{c^1|a^0,b^1} &= 0.09 & \theta_{c^1|a^1,b^1} &= 0.2,
\end{aligned}$$

So we have:

$$
\begin{aligned}
Q(\langle b^1, c^1\rangle) &= 0.3 \cdot 0.9 \cdot 0.2 \cdot 0.2 / 0.2196 = 0.0492 \\
Q(\langle b^1, c^0\rangle) &= 0.3 \cdot 0.9 \cdot 0.8 \cdot 0.9 / 0.2196 = 0.8852 \\
Q(\langle b^0, c^1\rangle) &= 0.3 \cdot 0.1 \cdot 0.6 \cdot 0.2 / 0.2196 = 0.0164 \\
Q(\langle b^0, c^0\rangle) &= 0.3 \cdot 0.1 \cdot 0.4 \cdot 0.9 / 0.2196 = 0.0492,
\end{aligned}
$$

(where $P(a^1, d^0 \mid \theta) = 0.2196$)

We also have the observation $o' = \langle ?, b^1, ?, d^1 \rangle$ so $Q'(A, C) = P(A, C \mid b^1, d^1, \boldsymbol{\theta})$ is:

$$
\begin{aligned}
Q'(\langle a^1, c^1\rangle) &= 0.3 \cdot 0.9 \cdot 0.2 \cdot 0.8 / 0.1675 = 0.2579 \\
Q'(\langle a^1, c^0\rangle) &= 0.3 \cdot 0.9 \cdot 0.8 \cdot 0.1 / 0.1675 = 0.1290 \\
Q'(\langle a^0, c^1\rangle) &= 0.7 \cdot 0.9 \cdot 0.09 \cdot 0.8 / 0.1675 = 0.2708 \\
Q'(\langle a^0, c^0\rangle) &= 0.7 \cdot 0.9 \cdot 0.91 \cdot 0.1 / 0.1675 = 0.3423.
\end{aligned}
$$

So the update for our desired parameter $\hat{\theta}_{d^1 \mid c^0}$, $\tilde{\boldsymbol{\theta}}_{d^1 \mid c^0} = \dfrac{\bar{M}_{\boldsymbol{\theta}}[d^1, c^0]}{\bar{M}_{\boldsymbol{\theta}}[c^0]}$ is:

$\tilde{\boldsymbol{\theta}}_{d^1 \mid c^0} = \dfrac{0.4713}{1.4057} = 0.3353$, where

$$
\begin{aligned}
\bar{M}_{\boldsymbol{\theta}}[d^1, c^0] &= Q'(\langle a^1, c^0\rangle) + Q'(\langle a^0, c^0\rangle) \\
&= 0.1290 + 0.3423 = 0.4713 \\
\bar{M}_{\boldsymbol{\theta}}[c^0] &= Q(\langle b^1, c^0\rangle) + Q(\langle b^0, c^0\rangle) + Q'(\langle a^1, c^0\rangle) + Q'(\langle a^0, c^0\rangle) \\
&= 0.8852 + 0.0492 + 0.1290 + 0.3423 = 1.4057.
\end{aligned}
$$

The General Intuition

The missing observations we had are like “weighted average” on all the full observations agreed on the data we saw. Where the weights are based on the parameters we hold.

More Formally

Let $H[m]$ denote the missing variables in sample $o[m]$, also define the data set D^+ witch consists from:

$\bigcup_m \{\langle o[m], h[m]\rangle \mid h[m] \in Val(H[m])\}$

Where $\langle o[m], h[m]\rangle$ is weighted by $Q(\boldsymbol{h}[m]) = P(\boldsymbol{h}[m] \mid \boldsymbol{o}[m], \boldsymbol{\theta})$. Now do standard maximum likelihood estimation using these completed data cases, where:

$$\bar{M}_{\boldsymbol{\theta}}[\boldsymbol{y}] = \sum_{m=1}^{M} \sum_{\boldsymbol{h}[m] \in Val(\boldsymbol{H}[m])} Q(\boldsymbol{h}[m]) \mathbb{1}\{\xi[m]\langle \boldsymbol{Y} \rangle = \boldsymbol{y}\}$$

Until Now We Are Happy ☺ (Should We?)

However, it may require an unreasonable amount of computation.

To compute the expected sufficient statistics, we must sum over all the completed data cases.

The number of these completed data cases is much larger than the original data set. For each $o[m]$, the number of completions is exponential in the number of missing values. Thus, if we have more than few missing values in an instances, an implementation of this approach will not be able to finish computing the expected sufficient statistics.

Fortunately, it turns out that there is a better approach to computing the expected sufficient statistic than simply summing over all possible completions.

Back To The Example

Let say we want to calculate $\bar{M}_{\boldsymbol{\theta}}[c^1] = \sum_{m=1}^{M} \sum_{\boldsymbol{h}[m] \in Val(\boldsymbol{H}[m])} Q(\boldsymbol{h}[m]) \mathbb{1}\{\xi[m]\langle C \rangle = c^1\}$,

For the observation $\boldsymbol{o} = \langle a^1, ?, ?, d^0 \rangle$. We have four possible completions, but we sum only two of them $Q(b^1, c^1) + Q(b^0, c^1)$ witch agree with c^1 , this expression is equal to $Q(c^1) = P(c^1 \mid a^1, d^0, \boldsymbol{\theta})$. Thus we have:

$$\bar{M}_{\boldsymbol{\theta}}[c^1] = \sum_{m=1}^{M} P(c^1 \mid \boldsymbol{o}[m], \boldsymbol{\theta})$$

Algorithm 19.2 Expectation-maximization algorithm for BN with table-CPDs

Procedure Compute-ESS (
  $\mathcal{G}$, // Bayesian network structure over $X_1, \ldots, X_n$
  $\boldsymbol{\theta}$, // Set of parameters for $\mathcal{G}$
  $\mathcal{D}$ // Partially observed data set
)

// Initialize data structures
for each $i = 1, \ldots, n$
  for each $x_i, \boldsymbol{u}_i \in Val(X_i, \mathrm{Pa}_{X_i}^{\mathcal{G}})$
    $\bar{M}[x_i, \boldsymbol{u}_i] \leftarrow 0$
  // Collect probabilities from all instances
for each $m = 1 \ldots M$
  Run inference on $\langle \mathcal{G}, \boldsymbol{\theta} \rangle$ using evidence $\boldsymbol{o}[m]$
  for each $i = 1, \ldots, n$
    for each $x_i, \boldsymbol{u}_i \in Val(X_i, \mathrm{Pa}_{X_i}^{\mathcal{G}})$
      $\bar{M}[x_i, \boldsymbol{u}_i] \leftarrow \bar{M}[x_i, \boldsymbol{u}_i] + P(x_i, \boldsymbol{u}_i \mid \boldsymbol{o}[m])$
return $\{\bar{M}[x_i, \boldsymbol{u}_i] : \forall i = 1, \ldots, n, \forall x_i, \boldsymbol{u}_i \in Val(X_i, \mathrm{Pa}_{X_i}^{\mathcal{G}})\}$

Procedure Expectation-Maximization (
  $\mathcal{G}$, // Bayesian network structure over $X_1, \ldots, X_n$
  $\boldsymbol{\theta}^0$, // Initial set of parameters for $\mathcal{G}$
  $\mathcal{D}$ // Partially observed data set
)

for each $t = 0, 1 \ldots,$ until convergence
  // E-step
  $\{\bar{M}_t[x_i, \boldsymbol{u}_i]\} \leftarrow$ Compute-ESS$(\mathcal{G}, \boldsymbol{\theta}^t, \mathcal{D})$
  // M-step
  for each $i = 1, \ldots, n$
    for each $x_i, \boldsymbol{u}_i \in Val(X_i, \mathrm{Pa}_{X_i}^{\mathcal{G}})$
      $\theta^{t+1}_{x_i|\boldsymbol{u}_i} \leftarrow \frac{\bar{M}_t[x_i, \boldsymbol{u}_i]}{\bar{M}_t[\boldsymbol{u}_i]}$
return $\boldsymbol{\theta}^t$

The Steps

As we can see the algorithm have two steps in each iteration:

- Expectation (E-step): using our current parameters, for each family X, U we calculate the join distribution $P(X,U \mid o[m],\theta^t)$ for each $o[m]$, using these distribution we calculate $\bar{M}_{\boldsymbol{\theta}^t}[x,\boldsymbol{u}] = \sum_m P(x,\boldsymbol{u} \mid \boldsymbol{o}[m],\boldsymbol{\theta}^t).$

- Maximization (M-step): Treat the expected sufficient statistics as observed, and perform maximum likelihood estimation, with respect to them, to derive a new set of parameters. In other words:

$$\boldsymbol{\theta}^{t+1}_{x|\boldsymbol{u}} = \frac{M_{\boldsymbol{\theta}^t}[x,\boldsymbol{u}]}{\bar{M}_{\boldsymbol{\theta}^t}[\boldsymbol{u}]}.$$

2 Theorems

First:

During iterations of the EM procedure of algorithm 19.2, we have

$$\ell(\boldsymbol{\theta}^t : \mathcal{D}) \leq \ell(\boldsymbol{\theta}^{t+1} : \mathcal{D}).$$

Second:

Suppose that $\boldsymbol{\theta}^t$ *is such that* $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t$ *during EM, and* $\boldsymbol{\theta}^t$ *is also an interior point of the allowed parameter space. Then* $\boldsymbol{\theta}^t$ *is a stationary point of the log-likelihood function.*

Remarks…

Combining these 2 theorems seems like if we are in case of the second theorem we should be happy, so it is not the case.

Actually the EM algorithm can converge to a local minimum or to a saddle point , is seems not reasonable but it is true, the good news that in practice there are rare starting point witch will get us to these point. The EM algorithm will lead us almost all the time to a **local** maximum, and this is the second bad news…

Further Reading

Also mentioned in this chapter: The EM Algorithm for Bayesian Networks Clustering, witch describe the use of EM for this purpose.

Other issue is: how fast the algorithm converge? From little discussion in the book and from other sources, we can say that in most cases the algorithm is fast enough for us. Still there is a place for further discussion on this topic.

EM Vs. Gradient Ascent - Similarity

There is big similarity between these two algorithms, both of them starts at a random point and both of them perform some version of greedy optimization based on the current point. Both algorithms provide a guarantee to converge to local maxima (or, more precisely, to stationary points where the gradient is 0). They also similar in the calculation they do in the running process.

EM Vs. Gradient Ascent – Difference

Gradient ascent can be easily applied to various CPDs by using the chain rule of derivatives. But the EM works on a full data, thus it can implement on a full data problem. The EM begins good, in the first steps there is good improvement, but in the later iterations this is not the case. The Gradient Ascent works in the opposite way, he is slow at the begging and fast in later iterations. It is not so clear witch algorithm is better.

You can read more about it in the chapter.