6

Parameter Estimation

Assume that you know the structure of a Bayesian network model over the variables $\mathcal{U}$, but you do not have any estimates for the conditional probabilities. On the other hand, you have access to a database of cases, i.e., a set of simultaneous values for some of the variables in $\mathcal{U}$. You can now use these cases to estimate the parameters of the model, namely the conditional probabilities. In this chapter we consider two approaches for handling this problem: First we show how a database of cases can be used to estimate the parameters once and for all (so-called *batch learning*). After that, we shall investigate the situation in which the cases are accumulated sequentially and we wish to *adapt* the model as each new case arrives. The reader is expected to be familiar with Section 1.5.

6.1 Complete Data

Let $M = (S, \boldsymbol{\theta})$ be a Bayesian network with structure S and parameters $\boldsymbol{\theta}$, and let $\mathcal{U}$ be the variables in M. Moreover, let $\mathcal{D}$ be a data set of cases, where each case is a configuration over *all* the variables in $\mathcal{U}$. Such a case is said to be *complete case*. In the learning community, a parameter is typically denoted by θ (rather than t as we have done previously), and in this chapter we shall follow the same convention. Moreover, to ensure that the parameters can be learned independently we shall make the following two assumptions:

- *Global independence* says that the parameters for the various variables are independent. This means that we can modify the tables for the variables independently.
- *Local independence* says that the uncertainties of the parameters for different parent configurations are independent. To be more precise, let (b, c) and (b', c') be different configurations; then the uncertainty on $P(A \mid b, c)$ is independent of the uncertainty on $P(A \mid b', c')$, and the parameters for the two distributions can be modified independently.

6.1.1 Maximum Likelihood Estimation

For each case $\mathbf{d} \in \mathcal{D}$, the probability $P(\mathbf{d}|M)$ is called the *likelihood* of M given $\mathbf{d}$. If we assume that the cases in $\mathcal{D}$ are independent given the model, then the *likelihood of M given $\mathcal{D}$* is

$$L(M \,|\, \mathcal{D}) = \prod_{\mathbf{d} \in \mathcal{D}} P(\mathbf{d}|M).$$

Often the log is taken, and it is then called the *log-likelihood*:

$$LL(M \,|\, \mathcal{D}) = \sum_{\mathbf{d} \in \mathcal{D}} \log_2 P(\mathbf{d}|M).$$

If we have to choose among several models for describing the data, then the *principle of maximum likelihood* advises us to choose a model of maximal likelihood given the data. This means that if we want to estimate the conditional probabilities, then our possible models $M_{\boldsymbol{\theta}}$ agree on the structure but differ with respect to the parameters $\boldsymbol{\theta}$. So we choose a parameter estimate $\hat{\boldsymbol{\theta}}$ that maximizes the likelihood:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} L(M_{\boldsymbol{\theta}} \,|\, \mathcal{D}) = \arg\max_{\boldsymbol{\theta}} LL(M_{\boldsymbol{\theta}} \,|\, \mathcal{D}).$$

In what follows we shall use $\hat{\boldsymbol{\theta}}$ to denote a maximum likelihood estimate for the parameters $\boldsymbol{\theta}$.

Example 6.1. We have tossed a thumbtack 100 times. It has landed pin up 80 times, and we look for the best estimate of the probability for *pin up*.

The situation is that we have a family of models, one for each possible value of θ, the probability of *pin up*. Let M_θ denote the model with $P(\textit{pin up}) = \theta$, then by assuming independent tosses, the likelihood of M_θ given the data is

$$P(\mathcal{D} \,|\, M_\theta) = \prod_{d \in \mathcal{D}} P(d \,|\, M_\theta) = \mu \cdot \theta^{80}(1-\theta)^{20},$$

where μ is a binomial factor independent of θ. By setting the derivative $\frac{d}{d\theta}P(\mathcal{D} \,|\, M_\theta)$ equal to zero it is easy to see that the likelihood is maximal for $\theta = 0.8$, so $\hat{\theta} = 0.8$.

In general, you get a maximum likelihood estimate as the fraction of positive counts over the total number of counts. This also holds for variables with more than two states. If you want to find a maximum likelihood estimate for the parameters in a Bayesian network model, then this can be done by finding maximum likelihood estimates for each conditional probability distribution in the model. That is, for each conditional probability distribution, e.g., $P(A = a \,|\, B = b, C = c)$, you simply calculate

$$\frac{N(A=a,B=b,C=c)}{N(B=b,C=c)},$$

where $N(A=a,B=b,C=c)$ is the number of cases in the database for which $A=a,B=b,C=c$.

The principle of maximal likelihood therefore supports the intuition of using frequencies as estimates, and to achieve a maximum likelihood estimate you just count. We did so in Section 3.2.4, where Table 3.10 was the result of 10, 000 words transmitted.

6.1.2 Bayesian Estimation

When you have a sparse database, maximum likelihood estimation has some drawbacks. Consider Table 6.1, which is the result of collecting 100 transmitted words. If you do maximum likelihood parameter estimation using this table, the outcomes with zero counts would be given zero probability and they are thereby doomed impossible, a rather strong assumption based on only 100 cases.

		Last three letters							
		aaa	*aab*	*aba*	*abb*	*baa*	*bba*	*bab*	*bbb*
First two letters	*aa*	2	2	2	2	5	7	5	7
	ab	3	4	4	4	1	2	0	2
	ba	0	1	0	0	3	5	3	5
	bb	5	6	6	6	2	2	2	2

Table 6.1. The table shows the number of five-letter words ($T_1T_2T_3T_4T_5$) transmitted over a channel. For example, the word *abaab* has appeared four times, whereas *bbabb* has appeared six times.

An alternative to the principle of maximum likelihood is *Bayesian estimation*: start with a prior distribution, and use experience to update the distribution. The approach can be illustrated with a Bayesian network, where each parameter for estimation is made explicit through a node. For the thumbtack experiment, a model for three tosses would be as in Figure 6.1. The conditional probabilities $P(pin\ up|\theta)$ are θ, and the prior distribution $f(\theta)$ is (as always) up to you. If you have no idea at all, a common approach is to use the uniform distribution $f(\theta)=1$, $0\leq\theta\leq 1$.

Now assume that we have performed one experiment with the result *pin up*. Using Bayes' rule we get

$$f_p(\theta|pin\ up)=\frac{P(pin\ up|\theta)f(\theta)}{P(\mathit{1}\ up)}=\frac{\theta f(\theta)}{P(pin\ up)}$$

for the posterior frequency function f_p. If we let $f(\theta)=1$, we get

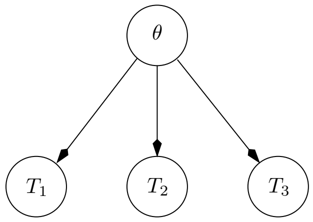

Fig. 6.1. A Bayesian network model for estimating the parameter given the outcome of three tosses.

$$f_p(\theta|pin\ up) = \frac{\theta}{P(pin\ up)}.$$

As usual, $P(pin\ up)$ is calculated as the normalization factor:

$$\int_0^1 \theta\, d\theta = \frac{1}{2},$$

so

$$f_p(\theta|pin\ up) = 2\theta.$$

This yields a distribution of the posterior for θ given *pin up*, and the best single estimate is the mean value of this distribution:

$$\int_0^1 \theta(2\theta)\, d\theta = \frac{2}{3}.$$

Next, assume that we get a toss with pin down. Then we have for the new posterior distribution

$$\begin{aligned} f_{p_2}(\theta|pin\ down, pin\ up) &= \frac{P(pin\ down, pin\ up|\theta)f(\theta)}{P(pin\ up, pin\ down)} \\ &= \mu P(pin\ down|\theta)P(pin\ up|\theta)f(\theta) \\ &= \mu P(pin\ down|\theta)\theta \cdot 1 = \mu(1-\theta)\theta, \end{aligned}$$

where μ is the normalization constant

$$\frac{1}{\mu} = P(pin\ down, pin\ up) = \int_0^1 (1-\theta)\theta\, d\theta = \frac{1}{6}.$$

The posterior distribution $f_{p2}(\theta|pin\ down, pin\ up)$ can now be written as

$$f_{p_2}(\theta|pin\ down, pin\ up) = 6(1-\theta)\theta,$$

and the single best estimate for θ is

$$\int_0^1 \theta 6(1-\theta)\theta\, d\theta = \frac{1}{2}.$$

Theorem 6.1. *Let X be a binary variable* (yes, no)*, and assume that we have performed a number of independent experiments out of which n turned up* yes *and m turned up* no. *Let θ be the probability for* yes. *Then, starting with the even prior distribution for θ, the posterior distribution is*

$$f_p(\theta) = \mu\theta^n(1-\theta)^m,$$

where μ is a normalization constant. The Bayesian estimate for θ is $\frac{n+1}{n+m+2}$.

Parameters estimated through Bayesian estimation are called *maximum a posteriori parameters*.

The theorem can be proved by induction along the lines described above. Moreover, the theorem can be interpreted so that an even prior distribution corresponds to adding two virtual experiments to the data (one for *yes* and one for *no*) and then counting frequencies.

This procedure also holds for distributions over more than two states. To pursue the Bayesian approach, assume for example that you wish to estimate $P(T_2 \mid T_1)$ from Table 6.1. First you marginalize out the other variables to obtain Table 6.2(a).

(a)

		T_1	
		a	b
T_2	a	32	17
	b	20	31

(b)

		T_1	
		a	b
T_2	a	33	18
	b	21	32

Table 6.2. (a) Counts of the first two letters from Table 6.1. (b) The table obtained by adding 1 to all counts in (a).

Next, add 1 to all cells (Table 6.2(b)), and you get the conditional probability table in Table 6.3.

		T_1	
		a	b
T_2	a	$\left(\frac{33}{54}\right)$	$\left(\frac{18}{50}\right)$
	b	$\left(\frac{21}{54}\right)$	$\left(\frac{32}{50}\right)$

Table 6.3. The result of a Bayesian approach for estimating the conditional probability table $P(T_2 \mid T_1)$

6.2 Incomplete Data

In the previous section we saw how the probability parameters in a Bayesian network can be estimated from a complete data set, i.e., a data set in which each case specifies a value for each of the variables. In practice, however, we are often faced with situations in which the data is incomplete. For example, some values may be accidentally missing (for example due to faulty sensor readings), some values may have been intentionally removed, and, in the more extreme case, some variables may simply not be observable (such variables are also called *latent variables* or *hidden variables*). If only some of the cases in the database contain missing values, then you could be tempted to simply throw these cases away and estimate the probability parameters using the remaining (complete) database. This approach, however, may have a serious drawback: Besides the risk of ending up with a very small database, we may unintentionally bias the parameter estimates. For example, assume that we have two binary variables A and B, and we are given a database with 20 cases over A and B. Assume also that the database contains an equal number of cases with $A = a_1$ and $A = a_2$, but when $A = a_2$, then the value of B is missing in 5 of the cases (B is not missing in any of the other cases). Now if we want to find the maximum likelihood estimate for θ, the probability of a_1, using the entire database, then (recall that $P^{\#}$ is the notation for frequency counts)

$$P^{\#}(a_1) = \hat{\theta} = \frac{N(a_1)}{N(a_1) + N(a_2)} = \frac{10}{10 + 10} = \frac{1}{2}.$$

However, if we throw away the cases that contain missing values, then the maximum likelihood estimate would be

$$P^{\#}(a_1) = \hat{\theta}' = \frac{N'(a_1)}{N'(a_1) + N'(a_2)} = \frac{10}{10 + 5} = \frac{2}{3}.$$

The difference in the two estimates is caused by A's influence on B's "missingness." On the other hand, if A does not have an influence on whether the value of B is missing in the database, then we can (if the database is large enough) safely throw away the cases with missing values without affecting the maximum likelihood estimate of A.

The example above illustrates that in order to deal with missing data we need to take into account *how* the data is missing. Consider the incomplete data set as having been produced from a complete data set by a process that hides some of the data.

- If the probability that a particular value is missing depends only on the observed values, then the data is said to be *missing at random* (MAR).
- If this probability is also independent of the observed values, then the data is said to be *missing completely at random* (MCAR).
- If the data is neither MAR nor MCAR, then the process that generated the missing data is said to be *nonignorable.*

In the definitions of MAR and MCAR, the probability that a value is missing is independent of that specific value. In particular, when we have hidden/latent variables, then the data is MCAR, since the variables are unobserved regardless of the values of any of the variables.

To give a few examples. Consider first an exit poll performed during an election, where an extreme right-wing party, ER, is running for parliament. If we expect people who vote for ER to be more likely than others to refuse to answer how they have voted, then the data is neither MCAR nor MAR. This also means that when estimating the parameters, we cannot disregard the underlying process causing the missing data. As another example, assume that we have a database containing the results of two tests. The results of both tests can be either *positive* or *negative*, but whereas the first test is always performed, the second test is performed only as a "backup test" when the result of the first test is *negative*. In this situation the pattern of "missingness" is dependent only on the observed values, hence the data is MAR. Finally, consider a monitoring system equipped with sensors whose values are continuously recorded and stored in a database. The recording system, however, is not completely stable, and sometimes a sensor value is not stored properly (i.e., it will be missing in the database). In this situation, the process causing the data to become missing is independent of all the sensor values, and the data is MCAR.

Today, the majority of the methods used for parameter estimation assume the data to be MAR, and in the remainder of this chapter we shall make the same assumption.

One approach to finding the maximum likelihood parameters could be to simply solve the corresponding likelihood equations. Unfortunately, this approach is not feasible in practice, since an incomplete case may cause the parameters to become dependent. The same holds if we were to consider the *maximum a posteriori parameters* $\boldsymbol{\theta}^*$:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta} \mid \mathcal{D}). \tag{6.1}$$

Instead, researchers have focused on approximative methods for doing parameter estimation.

6.2.1 Approximate Parameter Estimation: The EM Algorithm

One of the most popular algorithms for doing parameter estimation is the Expectation-Maximization (EM) algorithm. The EM algorithm is a general algorithm for finding maximum likelihood estimates for a set of parameters $\boldsymbol{\theta}$ when one is faced with an incomplete data set. The algorithm basically alternates between a so-called *expectation step* and a *maximization step*: loosely speaking, in the expectation step we "complete" the data set by using the current parameter estimates $\hat{\boldsymbol{\theta}}$ to calculate expectations for the missing values, and in the maximization step we use the "completed" data set to find a new maximum likelihood estimate $\hat{\boldsymbol{\theta}}'$ for the parameters. This estimate is then used to complete the data set in the next iteration of the algorithm. The algorithm continues either for a predetermined number of iterations or until the algorithm has converged.

Example 6.2. Consider the Bayesian network representation M of the simplified insemination problem described in Section 3.1.3 (page 55), and assume that we have the database in Table 6.4.

Cases	*Pr*	*Bt*	*Ut*
1.	*?*	*pos*	*pos*
2.	*yes*	*neg*	*pos*
3.	*yes*	*pos*	*?*
4.	*yes*	*pos*	*neg*
5.	*?*	*neg*	*?*

Table 6.4. A database consisting of five cases covering the variables *Pr*, *Bt*, and *Ut*. The *?* indicates that the value of the corresponding variable is missing.

When using the EM algorithm for learning the probability parameters based on this database, we first specify some initial "guesses" for the probability distributions for M, i.e., $P_0(Pr)$, $P_0(Bt \,|\, Pr)$ and $P_0(Ut \,|\, Pr)$. For the sake of simplicity we let all three probability distributions be even although you would usually start off with random distributions. Now, had the database been complete, then in order to find a new estimate for, say, the distribution $P(Pr = yes)$, we would count the number of cases $N(Pr = yes)$ with $Pr = yes$:

$$P_1^{\#}(Pr = yes) = \frac{N(Pr = yes)}{N}.$$

From the database we see that cases 2, 3, and 4 contain $Pr = yes$, and they therefore contribute with the value 1 to $N(Pr = yes)$. However, for cases 1 and 5 the value for Pr is missing. So to find the contribution from these two cases we use the probability of seeing $Pr = yes$: case 1 therefore contributes with $P_0(Pr = y \,|\, Bt = Ut = pos) = 0.5$ and case 5 contributes with $P_0(Pr = y \,|\, Bt = neg) = 0.5$. What we are actually calculating here is the expected value for $N(Pr = yes)$, denoted by $\mathbb{E}[N(Pr = yes)]$:

$$\begin{aligned}
\mathbb{E}[N(Pr = y)] =& P_0(Pr = y \,|\, Bt = Ut = pos) + 1 + 1 + 1 \\
&+ P_0(Pr = y \,|\, Bt = neg) = \frac{1}{2} + 1 + 1 + 1 + \frac{1}{2} = 4; \\
\mathbb{E}[N(Pr = n)] =& P_0(Pr = n \,|\, Bt = Ut = pos) + 0 + 0 + 0 \\
&+ P_0(Pr = n \,|\, Bt = neg) = \frac{1}{2} + 0 + 0 + 0 + \frac{1}{2} = 1.
\end{aligned}$$

In general, the expected value of $N(Pr = yes)$ is given by

$$\mathbb{E}[N(Pr = yes)] = \sum_{i=1}^{N} P_0(Pr = yes \mid \mathbf{d}_i).$$

We can now use the expected counts to calculate a new estimate for $P(Pr)$, but before we come that far we should also calculate the counts necessary for finding new estimates for the remaining probabilities. To estimate, say, $P(Ut = pos \mid Pr = yes)$, we need estimates for $P(Ut = pos, Pr = yes)$ and $P(Pr = yes)$:

$$\begin{aligned} P_1^{\#}(Ut = pos \mid Pr = yes) &= \frac{P^{\#}(Ut = pos, Pr = yes)}{P^{\#}(Pr = yes)} \\ &= \frac{\left[\dfrac{N(Ut = pos, Pr = yes)}{N}\right]}{\left[\dfrac{N(Pr = yes)}{N}\right]} \\ &= \frac{N(Ut = pos, Pr = yes)}{N(Pr = yes)}. \end{aligned}$$

Here $N(Ut = pos, Pr = yes)$ denotes the number of cases containing both $Ut = pos$ and $Pr = yes$. However, as for Pr, we cannot find $N(Ut = pos, Pr = yes)$ when there are missing values, so again we use the expected value/count

$$\mathbb{E}[N(Ut = pos, Pr = yes)] = \sum_{i=1}^{N} P(Ut = pos, Pr = yes \mid \mathbf{d}_i).$$

For the database above we get

$$\begin{aligned} \mathbb{E}[N(Ut = pos, Pr = yes)] = \, & P(Ut = pos, Pr = yes \mid Bt = pos, Ut = pos) + 1 \\ & + P(Ut = pos, Pr = yes \mid Bt = pos, Pr = yes) \\ & + 0 + P(Ut = pos, Pr = yes \mid Bt = neg) \\ = \, & \frac{1}{2} + 1 + \frac{1}{2} + 0 + \frac{1}{4} = 2.25. \end{aligned}$$

These counts are sufficient for finding new estimates for the probability parameters in the network (see Section 6.1). For example,

$$\begin{aligned} P_1^{\#}(Pr = yes) &= \frac{\mathbb{E}[N(Pr = yes)]}{N} = \frac{4}{5} = 0.8, \\ P_1^{\#}(Ut = pos \mid Pr = yes) &= \frac{\mathbb{E}[N(Ut = pos, Pr = yes)]}{\mathbb{E}[N(Pr = yes)]} = \frac{2.25}{4} = 0.5625. \end{aligned}$$

When a new estimate has been found for all the probabilities, the procedure starts over again, but this time you should use the newly found probability estimates when calculating the expected counts. The procedure continues until the probabilities no longer change or until another termination criterion is met. In the special case that the database is complete, the algorithm converges after one iteration and returns the maximum likelihood estimates for the parameters.

Calculation of Family Counts

In the example above, we saw that in order to find a new estimate for a conditional probability distribution $P(X \mid \mathrm{pa}(X))$ we should calculate the expected counts for the family $\{X\} \cup \mathrm{pa}(X)$ of variables. That is, for a specific configuration of the family we calculate the expected number of cases that contain this configuration. Intuitively, we can consider the following three situations:

1. If a case is inconsistent with the configuration (i.e., the case and the configuration disagrees on at least one value), then it counts as 0.
2. If a case contains the entire configuration, then it counts as 1.
3. If the value for a variable is missing in a case, then it contributes with a fractional count corresponding to the conditional probability of seeing the configuration.

The situations 1 and 2 are in fact special cases of situation 3.

From a computational point of view, the calculation of the expected counts is the main difficulty of the EM-algorithm: when a case does not contain a value for all the variables in question, then we need to calculate the conditional probability distribution for these variables given that particular case. We shall consider two situations: First, assume that we are interested in a specific configuration $\mathrm{fa}(A) = \mathbf{a}$ for a family of variables, and let $\mathbf{d}$ be a case with a missing value for exactly one variable, X, in $\mathrm{fa}(A)$. If $\mathbf{a}$ specifies $X = x$, then the probability for $\mathbf{a}$ given $\mathbf{d}$ is equal to the probability $P(X = x|\mathbf{d})$, which in turn can be calculated by a single propagation in the Bayesian network. Second, and more generally, assume that $\mathbf{d}$ contains missing values for a set of variables $\mathcal{X} \subseteq \mathrm{fa}(A)$ in the family. In this situation the probability for $\mathbf{a}$ can be read directly from the joint probability $P(\mathcal{X} \mid \mathbf{d})$, but this is not immediately provided by the Bayesian network. Fortunately, in order to calculate this probability we can exploit the junction tree architecture (see Section 4.4). In particular, the construction of the underlying junction tree ensures that each family of variables is contained in at least one clique, say V, having variables $\mathcal{V}$. Hence, after a single propagation of the evidence corresponding to case $\mathbf{d}$, all the required probabilities can be read directly from the potentials associated with V and its neighboring separators. Specifically, from Theorem 4.5 we see that if V is a clique with the set of potentials Φ_V and with k neighboring separators containing the V-directed sets of potentials $\Phi_1, \ldots, \Phi_k$, then

$$P(\mathcal{V}, \mathbf{d}) = \prod_{\phi_V \in \Phi_V} \phi_V \prod_{\phi_1 \in \Phi_1} \phi_1 \cdots \prod_{\phi_k \in \Phi_k} \phi_k .$$

From this joint probability we can find the required probability $P(\mathcal{X}, \mathbf{d})$ by marginalizing out the irrelevant variables:

$$P(\mathcal{X}, \mathbf{d}) = \sum_{\mathcal{V} \setminus \mathcal{X}} P(\mathcal{V}, \mathbf{e}).$$

We return to our previous example. In order to calculate all the expected counts, we use the junction tree structure shown in Figure 6.2.

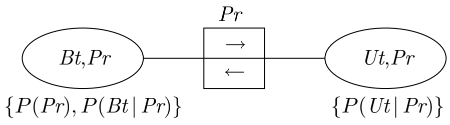

Fig. 6.2. A junction tree representation of the simplified insemination problem.

In particular, when calculating the contribution from case 5, we perform a full propagation with the evidence $Bt = neg$, and we get the annotated junction tree in Figure 6.3.

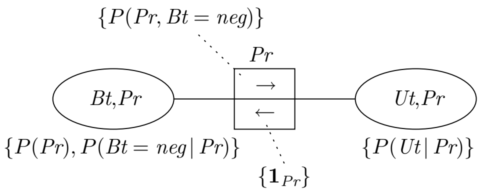

Fig. 6.3. A junction tree representation of the simplified insemination problem after inserting and propagating the evidence $Bt = neg$.

The required probability, e.g., $P(Ut, Pr \mid Bt = neg)$, can now be calculated directly from the potential in the clique containing Ut and the potential in the separator directed toward that clique:

$$\begin{aligned} P(Ut, Pr, Bt = neg) &= P(Ut \mid Pr) P(Pr, Bt = neg), \\ P(Bt = neg) &= \sum_{Ut, Pr} P(Ut, Pr, Bt = neg), \\ P(Ut, Pr \mid Bt = neg) &= \frac{P(Ut, Pr, Bt = neg)}{P(Bt = neg)}. \end{aligned}$$

Similarly, if we use the junction tree to calculate the contribution from case 5 to the expected counts for the family $\{Bt, Pr\}$, then we need $P(Bt, Pr \mid Bt = neg) = P(Pr \mid Bt = neg)$. This probability can be found using the same method as above:

$$\begin{aligned} P(Pr, Bt = neg) &= P(Bt = neg \mid Pr) P(Pr) \mathbf{1}_{Pr}, \\ P(Bt = neg) &= \sum_{Pr} P(Pr, Bt = neg), \\ P(Pr \mid Bt = neg) &= \frac{P(Pr, Bt = neg)}{P(Bt = neg)}. \end{aligned}$$

The EM-Algorithm for Bayesian Networks

We describe the algorithm more formally. Assume that we have a model structure B over the variables $\mathcal{U} = \{X_1, \ldots, X_n\}$, and let θ_{ijk} denote the parameter corresponding to the conditional probability $P(X_i = k \mid \mathrm{pa}(X_i) = j)$, i.e., the conditional probability for variable X_i being in its kth state given the jth configuration of the parents of X_i. Using this notation we can find a maximum likelihood estimate, $\hat{\theta}_{ijk}$, for the parameters θ_{ijk} given a data set $\mathcal{D} = \{\mathbf{d}_1, \ldots, \mathbf{d}_m\}$ with m cases as follows:

Algorithm 6.1 [The EM algorithm]

1. *Choose an* $\epsilon > 0$ *to regulate the stopping criterion.*
2. *Let* $\boldsymbol{\theta}^0 = \{\theta_{ijk}\}$*, where* $1 \leq i \leq n$*,* $1 \leq k \leq |\mathrm{sp}(X_i)| - 1$*, and* $1 \leq j \leq |\mathrm{sp}(\mathrm{pa}(X_i))|$*, be some initial estimates of the parameters (chosen arbitrarily).*
3. *Set* $t := 0$*.*
4. *Repeat:*
   E-step: For each $1 \leq i \leq n$ *calculate the table of expected counts:*

$$\underset{\boldsymbol{\theta}^t}{\mathbb{E}}[N(X_i, \mathrm{pa}(X_i)) \mid \mathcal{D}] = \sum_{\mathbf{d} \in \mathcal{D}} P(X_i, \mathrm{pa}(X_i) \mid \mathbf{d}, \boldsymbol{\theta}^t).$$

   M-step: Use the expected counts as if they were actual counts to calculate a new maximum likelihood estimate for all θ_{ijk}*:*

$$\hat{\theta}_{ijk} = \frac{\mathbb{E}_{\boldsymbol{\theta}^t}[N(X_i = k, \mathrm{pa}(X_i) = j) \mid \mathcal{D}]}{\sum_{h=1}^{|\mathrm{sp}(X_i)|} \mathbb{E}_{\boldsymbol{\theta}^t}[N(X_i = h, \mathrm{pa}(X_i) = j) \mid \mathcal{D}]}.$$

   Set $\boldsymbol{\theta}^{t+1} := \hat{\boldsymbol{\theta}}$ *and* $t := t + 1$*.*
   Until $|\log_2 P(\mathcal{D} \mid \boldsymbol{\theta}_t) - \log_2 P(\mathcal{D} \mid \boldsymbol{\theta}_{t-1})| \leq \epsilon$*.*

□

The EM-algorithm has been generalized for estimating the *maximum a posteriori* parameters (or penalized likelihood) instead of the maximum likelihood parameters. In this approach, virtual counts are added to both the denominator and numerator in the M-step, hence the method follows the idea of the Bayesian estimation method for complete data (see Section 6.1.2). As before, the virtual values can be interpreted as counts from a virtual database.

6.2.2 *Why We Cannot Perform Exact Parameter Estimation

When we have access to a complete database we can find the exact maximum likelihood parameters by simply counting frequencies in the database, or we can express the posterior probability distribution of the parameters in closed form. However, we are not that lucky when working with incomplete data. For example, assume that we have a probability distribution $P(\mathcal{U} \,|\, \boldsymbol{\theta})$ and that we get a single case $\mathbf{d}$ that specifies a configuration $\mathbf{x}$ over $\mathcal{X} \subset \mathcal{U}$; the variables $\mathcal{Y} = \mathcal{U} \setminus \mathcal{X}$ are therefore not observed. In order to find an estimate for the maximum likelihood parameters we should maximize the following expression with respect to $\boldsymbol{\theta}$:

$$P(\mathbf{x} \,|\, \boldsymbol{\theta}) = \sum_{\mathcal{Y}} P(\mathbf{x} \,|\, \mathcal{Y}, \boldsymbol{\theta}) P(\mathcal{Y} \,|\, \boldsymbol{\theta}).$$

That is, we maximize a sum having one term for each configuration of the unobserved variables. When performing the maximization we cannot consider the terms independently, since $P(\mathcal{Y} \,|\, \boldsymbol{\theta})$ will, in general, depend on all the parameters involved. Moreover, we have such a weighted sum for each case in the database; hence the number of terms may become intractably large.

6.3 Adaptation

When constructing a Bayesian network, you will almost always be uncertain of the correctness of the conditional probabilities specified, whether they are specified manually or learned from data. Usually you would allow each probability to range within an interval, and a number in this interval is then chosen. This type of uncertainty is called *second-order uncertainty*.

Second-order uncertainty raises two questions:

- Does the second-order uncertainty have an impact on the conclusions from the model?
- Are there systematic ways of reducing the second-order uncertainty?

The first question was discussed in Section 3.4 and was addressed in Section 5.6. In this section, we address the second question. We will look at a situation in which certain parameters are open for modification.

When a system is at work, you repeatedly get new cases, and you would like to learn from these cases. The situation may be that you are fairly certain of the structure of the network. However, the conditional probabilities are dependent on a context that varies from place to place, and you want to build a system that automatically adapts to the particular context in which it is placed.

In Figure 6.4(a), the variable A is directly influenced by B and C, and the strength is modeled by $P(A \,|\, B, C)$. The uncertainty in $P(A \,|\, B, C)$ may be modeled explicitly by introducing an extra parent, T, for A (Figure 6.4(b)). The variable T can be considered as a type variable. To reflect the frequencies of the context types, a prior distribution $P(T)$ is given.

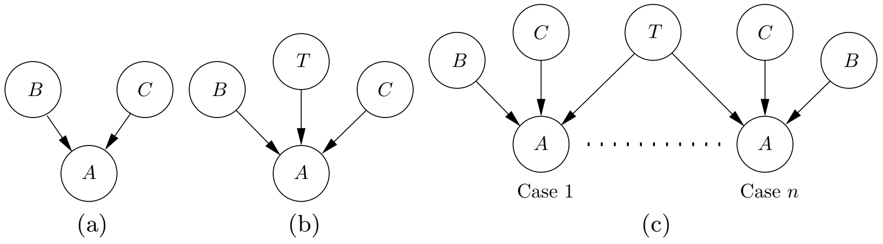

Fig. 6.4. Adaptation through a type variable T. The distribution of T is updated by *Case 1* and used in the next case.

When a case, $\mathbf{e}$, is entered into the network, the propagation will yield a new distribution $P^*(T) = P(T \mid \mathbf{e})$, and we may say that the change of the distribution for T reflects what has been learned from the case. Now $P^*(T)$ can be used as a new prior distribution when we get the next case. All variables whose tables are dependent on the context will be children of T. The way $P(T)$ is updated can also be made explicit in the network structure as shown in Figure 6.4(c). The network contains a copy of the variables for each case that will be considered, and when the ith case arrives, the corresponding variables are instantiated and $P^*(T) = P(T \mid \mathbf{e}_1, \ldots, \mathbf{e}_{i-1})$ is updated to $P(T \mid \mathbf{e}_1, \ldots, \mathbf{e}_i)$.

Example 6.3. Consider again the milk test problem described in Section 3.2.1, and assume that the farmer is not always as careful as he ought to be when performing the test. When this is the case, the risk of getting a false positive or a false negative is ten times as high as it otherwise would have been. Let us initially assume that there is an 80% chance that the farmer performs the test carefully.

One way of modeling this situation is to introduce a type variable *Type* (with states *careful* and *careless*) representing how the farmer performs the test (see Figure 6.5).

The probability $P(\mathit{Inf})$ is as before, and the conditional probability distribution $P(\mathit{Test} \mid \mathit{Inf}, \mathit{Type} = \mathit{careful})$ is as specified in Section 3.2.1. The probability distributions $P(\mathit{Test} \mid \mathit{Inf}, \mathit{Type} = \mathit{careless})$ and $P(\mathit{Type})$ can be derived from the description above, i.e., $P(\mathit{Type}) = (0.8, 0.2)$ and $P(\mathit{Test} \mid \mathit{Inf}, \mathit{Type} = \mathit{careless})$ are as specified in Table 6.5.

Now assume that a test is performed and the result is negative. When updating the probabilities with this piece of evidence you get $P^*(\mathit{Type}) = P(\mathit{Type} \mid \mathit{Test} = \mathit{neg}) = (0.815, 0.185)$. This probability distribution represents our updated belief in how the farmer performs the test. That is, the next time

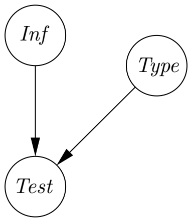

Fig. 6.5. The type variable *Type* models whether the farmer performs the milk test properly.

	$Inf = yes$	$Inf = no$
$Test = pos$	0.9	0.1
$Test = neg$	0.1	0.9

Table 6.5. The table shows the conditional probability distribution $P(Test \mid Inf, Type = careless)$.

you get new evidence you should use this conditional probability distribution (i.e., $P^*(Type)$) as the prior distribution for the variable *Type*.

Finally, it should be noted that you have to be a bit careful when working with several type variables. To illustrate the problem, assume that we get the case $A = a$ for the Bayesian network shown in Figure 6.6. When inserting this piece of evidence, we see, from d-separation, that S and T become dependent. Hence we cannot use their updated marginal distributions as prior distributions for the next case (by doing so we would have to assume that they are independent, which we have just seen is not the case). That is, in order for the above procedure to work correctly with several type variables, the evidence from a case should d-separate the type variables.

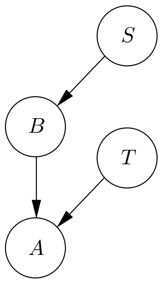

Fig. 6.6. A Bayesian network augmented with two type variables, S and T.

6.3.1 Fractional Updating

If the uncertainty of the conditional probabilities cannot be modeled explicitly through type variables, statistical methods can be used. The statistical task is first to specify a prior probability distribution over the parameters, and then iteratively update this distribution as new cases are entered. The correct approach for updating this distribution is basically the same as the task of learning exact parameter estimates from a database, but as we also saw in Section 6.2.2, this is infeasible in practice when we have missing values. Instead, approximative techniques are usually applied.

Consider $P(A \mid B, C)$, and let all variables be ternary. Under the assumptions of global and local independence, we may now think of $P(A \mid b_i, c_j) = (x_1, x_2, x_3)$ as a distribution established through a number of past cases in which (B, C) was in state (b_i, c_j). We can then express our certainty of the distribution by a fictitious sample size s. The larger the sample size, the smaller the second-order uncertainty, so we work with a *sample size* s, a set of counts (n_1, n_2, n_3) such that $s = n_1 + n_2 + n_3$, and

$$P(A \mid b_i, c_j) = \left(\frac{n_1}{s}, \frac{n_2}{s}, \frac{n_3}{s}\right).$$

That is, s represents the number of cases with (b_i, c_j), and n_1 is the number of these cases that also include a_1.

Let us first consider a couple of simple cases before we take the general case.

1. We get a new case $\mathbf{e}$ with $B = b_i$ and $C = c_j$ and with $A = a_1$. Then $n_1 := n_1 + 1$ and $s := s + 1$, and the probabilities are updated as follows:
$$x_1 := \frac{(n_1+1)}{(s+1)}; \qquad x_2 := \frac{n_2}{(s+1)}; \qquad x_3 := \frac{n_3}{(s+1)}.$$
2. We get a new case $\mathbf{e}$ with $B = b_i$ and $C = c_j$, but for A we have only a distribution $P(A \mid \mathbf{e}) = P(A \mid b_i, c_j, \mathbf{e}) = (y_1, y_2, y_3)$. Then we cannot work with integer counts, and we update $n_k := n_k + y_k$ and $s := s + 1$. Accordingly, we get
$$x_1 := \frac{(n_1+y_1)}{(s+1)}; \qquad x_2 := \frac{(n_2+y_2)}{(s+1)}; \qquad x_3 := \frac{(n_3+y_3)}{(s+1)}.$$
3. We get a new case $\mathbf{e}$ with $A = a_1$, but for B and C we have only $P(B = b_i, C = c_j \mid \mathbf{e}) = z$. As before, we cannot work with integer counts, so instead we update with a fractional count:
$$x_1 := \frac{(n_1+z)}{(s+z)}; \qquad x_2 := \frac{n_2}{(s+z)}; \qquad x_3 := \frac{n_3}{(s+z)}.$$

In general, we may get a case with $P(b_i, c_j \,|\, \mathbf{e}) = z$ and $P(A \,|\, b_i, c_j, \mathbf{e}) = (y_1, y_2, y_3)$. To update the counts, we use these distributions; because the sample size is increased only with z we take $n_k := n_k + zy_k$, and we get

$$x_k := \frac{(n_k + zy_k)}{(s + z)} = \frac{n_k + P(a_k, b_i, c_j \,|\, \mathbf{e})}{s + P(b_i, c_j \,|\, \mathbf{e})}.$$

This scheme is known as *fractional updating*. Unfortunately, the scheme has a serious drawback, namely that it tends to overestimate the count of s, thereby overestimating our certainty of the distribution. Assume for example that $e = \{B = b_i, C = c_j\}$. Then the case tells us nothing about $P(A \,|\, b_i, c_j)$, but nevertheless fractional updating will add a count of 1 to s and take it as a confirmation of the present distribution:

$$x_k := \frac{n_k + P(a_k \,|\, b_i, c_j)}{s + 1} = \frac{n_k + \frac{n_k}{s}}{s + 1} = \frac{n_k}{s}.$$

In Section 6.3.6 we shall return to this issue and consider another approximative updating method, which does not have the same drawback as mentioned above.

6.3.2 Fading

It is often a problem for fractional updating that the initial counts are kept when the system is trying to adapt to the environment. Particularly, when the conditional probabilities in the environment change over time, the accumulated counts will prevent the system from following the changes. Also, because fractional updating has a tendency to overestimate counts, vacuous counts will build up and make the parameters too resistant to change. Therefore, to keep the flexibility of parameters, it may be a good policy to prevent the sample size from growing unbounded.

An idea for solving this problem is the following: For example, let a ternary variable X have sample size s and counts (n_1, n_2, n_3), and assume that we get a count of 1 for x_1. Now, instead of increasing n_1 by one, we first multiply the counts by a fading factor, $q \in (0, 1)$. Hence, we get

$$s := sq + 1; n_1 := n_1 q + 1; n_2 := n_2 q; n_3 := n_3 q.$$

If we assume that all counts are of value 1, the influence from the past will fade away exponentially. In the limit where $s \to \infty$, we get a sample size s^*, where

$$s^* = \frac{1}{(1 - q)}.$$

The number s^* is called the *effective sample size*, and it represents a steady-state situation. If $s = s^*$ and we get a new count, we have

$$s := s^* q + 1 = \frac{q}{(1 - q)} + 1 = \frac{1}{(1 - q)} = s^*.$$

Instead of declaring a fading factor, you may declare an effective sample size s^*, and the fading factor is then

$$q^* = \frac{(s^* - 1)}{s^*}.$$

This idea can be used for each distribution $P(X \mid \mathrm{pa}(X) = \pi)$ that we wish to adapt to the evidence. The effective sample size need not be the same for all distributions. The effective sample size to declare is dependent on how resistant to change you wish the distribution to be. The higher the resistance, the higher the effective sample size.

Fading can be implemented such that the effective sample size is preserved. In other words, if the sample size for a distribution is equal to the declared effective sample size s^*, then it will not be changed in adapting to a new case.

Let $P(X \mid \pi)$ be declared with an effective sample size s^*, and assume we have $P(\pi \mid \mathbf{e}) = y$ for a case. Then fractional updating yields a new count of y. To preserve the sample size in the steady-state situation we have to adapt the fading factor q to the count y:

$$s^* q + y = s^*.$$

Hence

$$q = \frac{(s^* - y)}{s^*}.$$

Note that if $P(\pi \mid \mathbf{e}) = 1$, then $q = q^*$, and if $P(\pi \mid \mathbf{e}) = 0$, then $q = 1$.

6.3.3 *Specification of an Initial Sample Size

Frequently, the uncertainty of a parameter is expressed as an interval $[x, y]$. To exploit the technique for adaptation, the second-order uncertainty expressed by this interval will be translated to an initial sample size and a set of counts. The specification of the interval $[x, y]$ for $t = P(A = a)$ can be interpreted as, "I expect the value of t to be somewhere in the middle of the interval, and I am 90% sure that the value is in the interval." In other words, you have a distribution of t with mean close to $\frac{1}{2}(x + y)$ and with 90% of the density mass inside $[x, y]$.

As an example, take the interval $[0.3, 0.4]$ for the state a of the binary variable A. We interpret the interval as before, and assume that the distribution is the result of s samples out of which n were in state a. The distribution for t is a beta distribution, $\mathrm{Beta}(n_1, n_2)$, with mean $\mu = \frac{n_1}{s}$ and with variance $\sigma^2 = \frac{\mu(1-\mu)}{(s+1)}$, where $s = n_1 + n_2$ (see Figure 6.7 for examples). It holds that at least 90% of the probability mass lies in the interval $[\mu - 3\sigma, \mu + 3\sigma]$, so we seek values for s and n such that $\mu \approx 0.35$ and $\sigma \approx 0.0167$, and we get $n_1 = 285.16$ and $s = 814.73$.

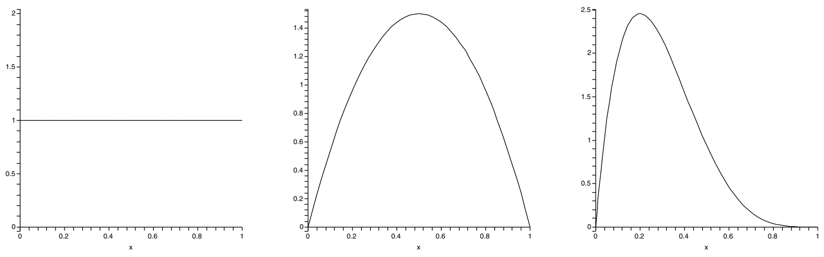

Fig. 6.7. The figure shows the density functions for the three beta distributions Beta$(1,1)$, Beta$(2,2)$, and Beta$(2,5)$.

6.3.4 Example: Strings of Symbols

Consider the transmission of symbols example from Section 3.2.4 with the model from Figure 3.18. Assume that every tenth word is sent through an error-correcting code, so that you know for certain the word transmitted. You wish to adapt the parameters of the model to the words actually transmitted and received.

First, you can use the coded words to adapt the distribution of the error rates: $P(R_i \mid T_i)$. Choose the effective sample size 100 for all parameters. This gives the fading factor 0.99. Also, let the initial sample be 100. The counts are given in Table 6.6.

	$T = a$	$T = b$
$R = a$	80	15
$R = b$	10	80
$R = c$	10	5

Table 6.6. Initial counts for $P(R \mid T)$.

Whenever a coded word is received, you have five cases (excluding the redundancy bits in the code). Assume that **baaba** was sent but **baaca** received. This means that the distribution $P(R \mid a)$ is modified three times and the distribution $P(R \mid b)$ is changed twice. For $P(R \mid a)$ we get the faded counts $((80 \cdot 0.99 + 1) \cdot 0.99 + 1) \cdot 0.99 + 1, 10 \cdot 0.99^3, 10 \cdot 0.99^3) = (80.6, 9.7, 9.7)$, and for $P(R \mid b)$ we get the faded counts $(15 \cdot 0.99^2, (80 \cdot 0.99 + 1) \cdot 0.99, 5 \cdot 0.99^2 + 1) = (14.7, 79.4, 5.9)$.

The noncoded words cannot be used for adaptation of $P(R \mid T)$, but they can be used for modifying $P(T_1)$ as well as $P(T_{i+1} \mid T_i)$. Assume that we receive the word $e = baaca$. Let us concentrate on modifying $P(T_2 \mid T_1)$. Let the initial sample size be 50 for $T_1 = a$ and 150 for $T_1 = b$. From Table 3.11, we infer the count table as given in Table 6.7.

	$T_1 = a$	$T_1 = b$
$T_2 = a$	30	60
$T_2 = b$	20	90

Table 6.7. Initial counts for $P(T_2 \mid T_1)$.

The model from Exercise 3.13 yields $P(T_1 \mid \mathbf{e}) = (0.13, 0.87)$, $P(T_2 \mid T_1 = a, \mathbf{e}) = (0.81, 0.19)$, and $P(T_2 \mid T_1 = b, \mathbf{e}) = (0.66, 0.34)$. The fading factors are $(100 - 0.13)/100 = 0.9987$ and $(100 - 0.87)/100 = 0.9913$. We get for $P(T_2 \mid a)$ the counts $(30 \cdot 0.9987 + 0.13 \cdot 0.81, 20 \cdot 0.9987 + 0.13 \cdot 0.19) = (30.07, 20.00)$ and for $P(T_2 \mid b)$ we get $(60 \cdot 0.9913 + 0.87 \cdot 0.66, 90 \cdot 0.9913 + 0.87 \cdot 0.34) = (60.05, 89.5)$.

Note that the sample size increases for the part with initial sample size smaller than the effective sample size and decreases for the part with initial sample size larger than the effective sample size.

6.3.5 Adaptation to Structure

As for the parameters in a model, it may happen that the structure of the model does not fit the cases you meet. If you use incremental adaptation of parameters, you will often experience that the changes in parameter values to a large degree will compensate for a slightly incorrect structure. Anyway, the structural inaccuracy may be so substantial that parameter adjustments cannot compensate. Unfortunately, no handy method for incremental adaptation of structure has been constructed. The reason is that structural changes are performed in jumps, and the justification for a jump is based on accumulated experience rather than a single case.

Basically, there are two ways out: you can accumulate the cases and run a batch learning algorithm (see Chapter 7) now and then, or you can work concurrently with several models. The second way is similar to the "expert disagreement approach."

Assume that you have three alternative models M_1, M_2, M_3 with initial normalized weights w_1, w_2, w_3; these weights can be interpreted as the probabilities for the models, $P(M_1)$, $P(M_2)$, and $P(M_3)$. A case with evidence $\mathbf{e}$ is entered into all models, and propagation yields $P(A \mid M_i, \mathbf{e})$ as well as $P(\mathbf{e} \mid M_i)$, where A is any variable. Then we can calculate new weights for the models

$$w_i := P(M_i \mid \mathbf{e}) = \frac{P(\mathbf{e} \mid M_i) P(M_i)}{P(\mathbf{e})} = \frac{P(\mathbf{e} \mid M_i) w_i}{\sum_j w_j P(\mathbf{e} \mid M_j)},$$

as well as the probability for the variable A:

$$P(A \mid \mathbf{e}) = w_1 P(A \mid M_1, \mathbf{e}) + w_2 P_2(A \mid M_2, \mathbf{e}) + w_3 P_3(A \mid M_3, \mathbf{e}).$$

6.3.6 *Fractional Updating as an Approximation

As we saw in Section 6.3.1, fractional updating has a serious drawback, namely that it tends to overestimate the sample size. To overcome this problem an alternative updating method (called *incremental updating*) has been proposed. Both fractional updating and incremental updating have their origins in the same problem: exact updating of the probability parameters is intractable, since it requires us to keep track of a mixture of Dirichlet distributions, where the number of mixture components may grow exponentially in the number of cases. More specifically, given evidence $\mathbf{e}$, both updating methods look for an approximation of the posterior distribution $P(\boldsymbol{\theta}|\mathbf{e})$, which determines the conditional probability distributions in the network.

In order to illustrate the updating method, we will first revisit the initial problem and show some of the derivations that underlie both fractional updating and incremental updating. Based on this, we will consider where the two updating methods differ.

Consider again the conditional probability distribution $P(A \mid B, C)$, where all variables are ternary. We set $P(A = a_k \mid B = b_i, C = c_j, \boldsymbol{\theta}) = \theta_{ijk}$ (see Figure 6.8 for a graphical representation) such that $\boldsymbol{\theta} = \{\theta_{ijk}\}$ and $1 \leq i \leq 3$, $1 \leq j \leq 3$, and $2 \leq k \leq 3$; the parameter θ_{ij1} is given by $1 - (\theta_{ij2} + \theta_{ij3})$. We will sometimes use the shorthand notation $\boldsymbol{\theta}_{ij} = \{\theta_{ij1}, \theta_{ij2}, \theta_{ij3}\}$, and we also assume that the prior distribution for $\boldsymbol{\theta}_{ij}$ follows a Dirichlet distribution with hyperparameters (n_1, n_2, n_3), denoted by $\mathrm{Dir}[\boldsymbol{\theta}_{ij}|n_1, n_2, n_3]$.

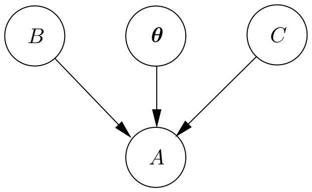

Fig. 6.8. An explicit representation of the parameter $\boldsymbol{\theta}$, which determines the conditional probability distribution $P(A = a_k \mid B = b_i, C = c_j)$.

Now assume that we have the simple case with evidence $\mathbf{e} = \{A = a_2, B = b_i, C = c_j\} \cup \mathbf{e}'$. Since A, B, and C constitute the Markov blanket for $\boldsymbol{\theta}$, we can disregard $\mathbf{e}'$ when updating the distribution for the parameters $\boldsymbol{\theta}_{ij}$, i.e., $f(\boldsymbol{\theta}_{ij}|a_2, b_i, c_j, \mathbf{e}') = f(\boldsymbol{\theta}_{ij}|a_2, b_i, c_j)$. Moreover, due to the choice of prior distribution for $\boldsymbol{\theta}$, we have

$$f(\boldsymbol{\theta}_{ij}|a_2, b_i, c_j) = \mathrm{Dir}[\boldsymbol{\theta}_{ij}|n_1, n_2 + 1, n_3].$$

As we did in the thumbtack problem (Section 6.1.2), we can similarly find a single point estimate for $P(A = a_k \mid B = b_i, C = c_j)$ by calculating the expectation of θ_{ijk} given $\mathbf{e} = \{A = a_2, B = b_i, C = c_j\} \cup \mathbf{e}'$:

$$\begin{aligned}
P'(A = a_k \mid B = b_i, C = c_j) &= \int_{\boldsymbol{\theta}_{ij}} \theta_{ijk} \mathrm{Dir}[\boldsymbol{\theta}_{ij} | n_1, n_2 + 1, n_3] d\boldsymbol{\theta}_{ij} \\
&= \begin{cases} \frac{n_k + 1}{n_1 + n_2 + n_3 + 1} & \text{for } k = 2, \\ \frac{n_k}{n_1 + n_2 + n_3 + 1} & \text{otherwise.} \end{cases}
\end{aligned}$$

These updating rules are identical to those for fractional updating.

Consider now the more general situation in which the evidence does not necessarily include A, B, and C. In this case, we first express the posterior distribution $f(\boldsymbol{\theta}_{ij} \mid \mathbf{e})$ as follows (recall that $f(\boldsymbol{\theta} \mid A, B, C) = f(\boldsymbol{\theta} \mid A, B, C, \mathbf{e})$):

$$f(\boldsymbol{\theta} \mid \mathbf{e}) = \sum_A \sum_B \sum_C f(\boldsymbol{\theta} \mid A, B, C) P(A, B, C \mid \mathbf{e}).$$

From the assumption of local parameter independence (see Section 6.3.1) we can derive that $f(\boldsymbol{\theta}_{ij}) = f(\boldsymbol{\theta}_{ij} \mid A, B = b_{i'}, C = c_{j'})$, for $i' \neq i$ or $j' \neq j$. This allows us to decompose the above expression into two parts, one with $j' = j$ and $i' = i$ and the other with $j' \neq j$ and $i' \neq i$:

$$\begin{aligned}
f(\boldsymbol{\theta}_{ij} \mid \mathbf{e}) = &\sum_A f(\boldsymbol{\theta}_{ij} \mid A, B = b_i, C = c_j) P(A, B = b_i, C = c_j \mid \mathbf{e}) \\
&+ \sum_{j' \neq j} \sum_{i' \neq i} \sum_A f(\boldsymbol{\theta}_{ij}) P(A, B = b_{i'}, C = c_{j'} \mid \mathbf{e}) \\
= &\sum_A f(\boldsymbol{\theta}_{ij} \mid A, B = b_i, C = c_j) P(A, B = b_i, C = c_j \mid \mathbf{e}) \\
&+ f(\boldsymbol{\theta}_{ij})(1 - P(B = b_i, C = c_j \mid \mathbf{e})).
\end{aligned}$$

As we also used above, we have that, for example,$f(\boldsymbol{\theta}_{ij} \mid A = a_2, B = b_i, C = c_j) = \mathrm{Dir}[\boldsymbol{\theta}_{ij} | n_1, n_2 + 1, n_3]$; hence the above expression can be rewritten as

$$\begin{aligned}
f(\boldsymbol{\theta}_{ij} \mid \mathbf{e}) = &\mathrm{Dir}[\boldsymbol{\theta}_{ij} | n_1 + 1, n_2, n_3] P(A = a_1, B = b_i, C = c_j \mid \mathbf{e}) \\
&+ \mathrm{Dir}[\boldsymbol{\theta}_{ij} | n_1, n_2 + 1, n_3] P(A = a_2, B = b_i, C = c_j \mid \mathbf{e}) \\
&+ \mathrm{Dir}[\boldsymbol{\theta}_{ij} | n_1, n_2, n_3 + 1] P(A = a_3, B = b_i, C = c_j \mid \mathbf{e}) \\
&+ \mathrm{Dir}[\boldsymbol{\theta}_{ij} | n_1, n_2, n_3](1 - P(B = b_i, C = c_j \mid \mathbf{e})).
\end{aligned} \tag{6.2}$$

Note that the last term models the situation in which the specified parent configuration is not observed, and if it is observed then the term contributes with zero.

This equation readily generalizes to a variable A with r states and parent configuration π:

$$\begin{aligned}
f(\boldsymbol{\theta}_\pi \mid \mathbf{e}) = &\sum_{k=1}^{r} \mathrm{Dir}[\boldsymbol{\theta}_\pi | n_1, \ldots, n_k + 1, \ldots, n_r] P(A = a_k, \mathrm{pa}(A) = \pi \mid \mathbf{e}) \\
&+ \mathrm{Dir}[\boldsymbol{\theta}_\pi | n_1, \ldots, n_r](1 - P(\mathrm{pa}(A) = \pi \mid \mathbf{e})).
\end{aligned} \tag{6.3}$$

Unfortunately, there is a computational problem with this expression, namely that the number of mixture components may grow exponentially in the number of cases that we process. This problem has led to the development of approximate updating methods such as fractional updating and incremental updating. Both of these methods approximate the mixture above using a single Dirichlet distribution, but there is a difference in how they estimate the parameters.

Fractional Updating Revisited

In fractional updating, equation (6.3) is approximated with a single Dirichlet distribution. The hyperparameters for this approximate distribution are formed by taking the linear combination (as defined by the mixture) of the corresponding hyperparameters in the mixture (disregarding the last term). For example, for the first hyperparameter n_1' in equation (6.2) we get

$$\begin{aligned} n_1' &= (n_1+1)P(A=a_1, B=b_i, C=c_j|\mathbf{e}) \\ &\quad + n_1 P(A=a_2, B=b_i, C=c_j|\mathbf{e}) \\ &\quad + n_1 P(A=a_3, B=b_i, C=c_j|\mathbf{e}) \\ &= n_1 + P(A=a_1, B=b_i, C=c_j|\mathbf{e}). \end{aligned}$$

That is, the mixture in equation (6.2) is approximated by

$$\begin{aligned} f'(\boldsymbol{\theta}_{ij}\,|\,\mathbf{e}) = \mathrm{Dir}[&n_1 + P(A=a_1, B=b_i, C=c_j\,|\,\mathbf{e}), \\ &n_2 + P(A=a_2, B=b_i, C=c_j\,|\,\mathbf{e}), \\ &n_3 + P(A=a_3, B=b_i, C=c_j\,|\,\mathbf{e})], \end{aligned}$$

and the new estimate for $P(A=a_k\,|\,, B=b_i, C=c_j)$ is then given by the mean value of θ_{ijk}:

$$\begin{aligned} P'(A=a_k\,|\,, B=b_i, C=c_j) = \int_{ij} \theta_{ijk}\mathrm{Dir}[&n_1 + P(A=a_1,, B=b_i, C=c_j\,|\,\mathbf{e}), \\ &n_2 + P(A=a_2,, B=b_i, C=c_j\,|\,\mathbf{e}), \\ &n_3 + P(A=a_3,, B=b_i, C=c_j\,|\,\mathbf{e})]d\boldsymbol{\theta}_{ij}. \end{aligned}$$

Hence

$$\begin{aligned} P'(A=a_k\,|\,, B=b_i, C=c_j) &= \frac{n_k + P(A=a_k,, B=b_i, C=c_j\,|\,\mathbf{e})}{n_1+n_2+n_3+P(B=b_i, C=c_j\,|\,\mathbf{e})} \\ &= \frac{n_k + P(A=a_k,, B=b_i, C=c_j\,|\,\mathbf{e})}{s+P(B=b_i, C=c_j\,|\,\mathbf{e})}, \end{aligned}$$

and by comparing this result with the updating rule presented in Section 6.3.1 we see that they are identical. Thus, the intuitive appeal of fractional updating that we saw in Section 6.3.1 rests on a mathematical foundation.

The Incremental Updating Rule

Analogously to fractional updating, when doing incremental updating we also estimate the mixture of Dirichlet distributions in equation (6.3) with a single Dirichlet distribution. However, the hyperparameters for the approximate Dirichlet distribution are determined by equating the means and average variance of the mixture to the means and the average variance of the approximating distribution. To be more specific, let θ^*_{ik} denote the mean of $\theta_{\cdot\cdot k}$ in the ith component in equation (6.3). The mean for the kth parameter in the mixture is then (θ^*_{0k} denotes the mean of $\theta_{\cdot\cdot k}$ in the last term)

$$\theta^*_k = \sum_{i=1}^{r} \theta^*_{ik} P(A = a_i, \pi \,|\, \mathbf{e}) + \theta^*_{0k}(1 - P(B = b_j, C = c_j|\mathbf{e})).$$

Thus estimating the mixture with a single Dirichlet distribution having the parameters $(s\theta^*_1, \ldots, s\theta^*_r)$ will provide the correct means for the parameters θ_k. The value for s is found by setting the average variance of the approximating distribution

$$\tilde{v} = \sum_{i=1}^{r} \theta^*_i \frac{\theta^*_i(1 - \theta^*_i)}{s + 1},$$

equal to the average-mean-weighted variance of the mixture. This gives the following updating value for s:

$$s = \frac{\sum_{k=1}^{r} \theta^{*2}_k (1 - \theta^*_k)}{\sum_{k=1}^{r} \theta^*_k v_k} - 1,$$

where v_k is the variance of θ^*_k in the mixture. Although this updating rule does not have the same intuitive appeal as fractional updating, it has the property that the sample size will not increase when no relevant evidence is entered. In fact, it is actually possible for the sample size to decrease if the evidence does not reflect an event with high prior probability.

6.4 Tuning

We have a Bayesian network BN. For this network, we have some evidence $\mathbf{e}$, and for a particular variable A we have $\mathbf{x} = P(A \,|\, \mathbf{e}) = (x_1, \ldots, x_n)$. We may have a prior request $\mathbf{y} = (y_1, \ldots, y_n)$ for $P(A \,|\, \mathbf{e})$, so we want to tune the network such that $P(A \,|\, \mathbf{e}) = \mathbf{y}$. Assume that the structure of BN is fixed, but for the conditional probabilities we have some freedom described by a set of modifiable parameters $\mathbf{t} = (t_1, \ldots, t_m)$ with an initial set of values $\mathbf{t}_0$; to emphasize that we consider a subset of the parameters we use t_i to represent a parameter rather than θ_{ijk} as we previously have used. We want to set the parameters so that $P(A \,|\, \mathbf{e})$ is sufficiently close to $\mathbf{y}$. One way to measure how close the two distributions are would be to use the Euclidean distance:

Definition 6.1. *Let* $\mathbf{x} = (x_1, \ldots, x_n)$ *and* $\mathbf{y} = (y_1, \ldots, y_n)$ *be two probability distributions. Then the* Euclidean distance *between* $\mathbf{x}$ *and* $\mathbf{y}$ *is (although we do not take the square root):*

$$\text{dist}\,(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} (x_i - y_i)^2.$$

The Euclidean distance measure is a *metric*, meaning that:

1. $\text{dist}(\mathbf{x}, \mathbf{y}) = 0$ if and only if $\mathbf{x} = \mathbf{y}$.
2. $\text{dist}(\mathbf{x}, \mathbf{y}) \leq \text{dist}(\mathbf{x}, \mathbf{z}) + \text{dist}(\mathbf{z}, \mathbf{y})$.
3. $\text{dist}(\mathbf{x}, \mathbf{y}) = \text{dist}(\mathbf{y}, \mathbf{x})$.

Another distance measure frequently used is the Kullback-Leibler divergence:

Definition 6.2. *Let* $\mathbf{x} = (x_1, \ldots, x_n)$ *and* $\mathbf{y} = (y_1, \ldots, y_n)$ *be two probability distributions. Then the* Kullback-Leibler divergence *between* $\mathbf{x}$ *and* $\mathbf{y}$ *is:*

$$\text{KL}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} x_i \log_2 \left(\frac{x_i}{y_i} \right),$$

where $0 \log_2(0/y_i) = 0$ *and* $x_i \log_2(x_i/0) = \infty$.

Note that the Kullback-Leibler divergence does not satisfy property 3 above so it is not a metric. In the remainder of this section we shall consider only the Euclidean distance.

If $(t_1, \ldots, t_n)$ are parameters in the Bayesian network BN (parameters are entries in conditional probability tables, see also Section 5.7) over the universe $\mathcal{U}$, then $P(\mathcal{U})$ is a function of $(t_1, \ldots, t_n)$, as are also $P(A \,|\, \mathbf{e})$ and $P(\mathbf{e})$. In the following, we assume proportional scaling, and we also assume that there is at most one parameter per distribution.

The task is to set the parameters such that the distance is as small as possible. If the parameters cannot be set in such a way that the distance is close to zero, then it is an indication of an incorrect structure.

If it is possible to determine $\text{dist}(\mathbf{x}, \mathbf{y})$ as a function of $\mathbf{t}$, you might be so fortunate that the problem can be solved directly. However, usually the problem cannot be solved directly even when the function is known, and a *gradient descent* method can be used:

1. Calculate $\mathbf{grad}\,\text{dist}(\mathbf{x}, \mathbf{y})$ with respect to the parameters $\mathbf{t}$.
2. Give $\mathbf{t}_0$ a displacement $\triangle\mathbf{t}$ in the direction opposite to the direction of the gradient $\mathbf{grad}\,\text{dist}\,(\mathbf{x}, \mathbf{y})\,(\mathbf{t}_0)$; that is, choose a step size $\alpha > 0$ and let $\triangle\mathbf{t} = -\alpha\,\mathbf{grad}\,\text{dist}\,(\mathbf{x}, \mathbf{y})\,(\mathbf{t}_0)$.
3. Iterate this procedure until the gradient is close to $\mathbf{0}$.

From the definition of the Euclidean distance measure, we see that

$$\frac{\partial}{\partial t}\operatorname{dist}(\mathbf{x},\mathbf{y}) = \sum_i 2(x_i - y_i)\frac{\partial x_i}{\partial t}.$$

The y_i's are known, and the x_i's are available through updating in BN, so what we need are **grad** $x_i(\mathbf{t})$ for all i. If the variable A is binary, we have $\mathbf{x} = (x, 1-x)$, $\mathbf{y} = (y, 1-y)$, and

$$\operatorname{dist}(\mathbf{x},\mathbf{y}) = 2(x-y)^2$$

and

$$\mathbf{grad}\operatorname{dist}(\mathbf{x},\mathbf{y}) = 4(x-y)\,\mathbf{grad}\,x,$$

From these formulas, we see that the gradient is $\mathbf{0}$ if and only if either x is independent of all the parameters or $x = y$.

6.4.1 Example

Let BN be the Bayesian network in Figure 6.9 with initial probabilities from Table 6.8. Let C be the information variable and A the variable of interest. Assume also that the parameters are $t = P(\neg a)$ and $s = P(\neg c \mid \neg b)$. Initially, we have $\mathbf{t}_0 = (0.5, 0.4)$.

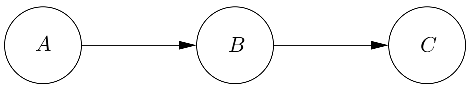

Fig. 6.9. A small Bayesian network for illustration.

$B \setminus A$	a	$\neg a$
b	1	0.3
$\neg b$	0	0.7

$C \setminus B$	b	$\neg b$
c	1	0.6
$\neg c$	0	0.4

Table 6.8. Parameters for the network in Figure 6.9, $P(A) = (0.5, 0.5)$.

Assume that we require $P(A \mid c) = (0.4, 0.6) = (y, 1-y)$. Through updating, we get $x = P(a \mid c) = 0.58$. We calculate $P(a \mid c)$ as a function of $\mathbf{t}$:

$$P(A,c) = \sum_B P(A)P(B \mid A)P(c \mid B) = (1-t, t-0.7ts),$$
$$P(a \mid c) = \frac{P(a,c)}{\sum_A P(A,c)}.$$

We get

$$P(a \mid c) = x(t,s) = \frac{(1-t)}{(1-0.7ts)}.$$

The request is

$$\frac{1-t}{1-0.7ts} = 0.4,$$

which yields

$$s = \frac{t-0.6}{0.28t} = \frac{25}{7} - \frac{15}{7t}.$$

The set of parameters t meeting the request is shown in Figure 6.10.

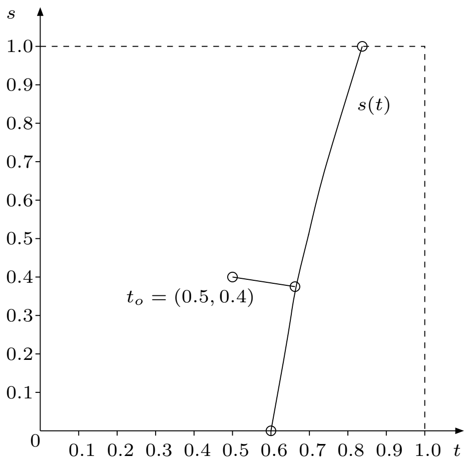

Fig. 6.10. The graph of $s(t)$ consists of the parameter pairs (t,s) meeting the request $P(a \mid c) = 0.4$.

Out of the infinite number of parameter pairs $(t, s(t))$, we choose one. If we do not wish to choose either of the extremes $(0.6, 0)$ and $(\frac{5}{6}, 1)$, it would be natural to choose the point closest to $\mathbf{t}_0 = (0.5, 0.4)$. This point is characterized by the property that the normal contains $\mathbf{t}_0$ (see Figure 6.10). Through standard calculations, we get the following equation in t:

$$t^4 - \frac{1}{2}t^3 + \frac{666}{98}t - \frac{225}{49} = 0.$$

A root is $t = 0.668$, and we get $s = 0.364$. For this very simple example, it was possible to calculate the closest parameter setting meeting the request.

The situation need not be much more complex before a direct calculation becomes intractable.

The gradient descent method will in this example go as follows:

$$\begin{aligned}\mathbf{grad}\, x(t) &= \frac{1}{(1-0.7ts)^2}(0.7s-1,(1-t)0.7t),\\ \mathbf{grad}\, x(t_0) &= (-0.97, 0.24).\end{aligned}$$

Formula (6.4) yields

$$\mathbf{grad}\,\mathrm{dist}\,(\mathbf{x},\mathbf{y}) = 4(0.58-0.4)(-0.97,0.24) = (-0.70, 0.18).$$

Using a step size of 0.2, we get

$$\triangle\mathbf{t} = (0.14, -0.036)$$

and

$$\mathbf{t}_1 = (0.640, 0.364); \quad P^1(a\,|\,c) = 0.43.$$

The process is repeated:

$$\begin{aligned}\mathbf{grad}\, x(t_1) &= (-1.06, 0.23),\\ \mathbf{grad}\,\mathrm{dist}\,(x,y) &= (-0.13, 0.03),\\ \mathbf{t}_2 &= (0.686, 0.358); \quad P^2(a\,|\,c) = 0.380.\end{aligned}$$

Repeating once more yields

$$\mathbf{t}_3 = (0.672, 0.361); \quad P^3(a\,|\,c) = 0.395.$$

6.4.2 Determining grad dist(x, y) as a Function of t

The gradient descent method seems to require that we be able to calculate $\mathbf{x}$ and $\mathbf{grad}\,\mathbf{x}$ as a function of the parameters $\mathbf{t}$. It was possible for the preceding small example, but the method used will in general be intractable.

Instead, the results form Section 5.7 can be used. By using proportional scaling we have

$$x = \frac{\alpha t+\beta}{at+b}.$$

This yields

$$\frac{\partial x}{\partial t} = \frac{\alpha(at+b)-a(\alpha t+\beta)}{(at+b)^2} = \frac{\alpha b - a\beta}{(at+b)^2},$$

where the constants can be found as described in Section 5.7.

6.5 Summary

Maximum Likelihood Estimation

For each case $\mathbf{d} \in \mathcal{D}$, the probability $P(\mathbf{d}|M)$ is called the *likelihood* of M given $\mathbf{d}$. If we assume that the cases in $\mathcal{D}$ are independent given the model, then the *likelihood of M given $\mathcal{D}$* is

$$L(M \,|\, \mathcal{D}) = \prod_{\mathbf{d} \in \mathcal{D}} P(\mathbf{d}|M).$$

The parameters $\boldsymbol{\theta}$ maximizing the likelihood are called the maximum likelihood parameters (and denoted by $\hat{\boldsymbol{\theta}}$):

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} L(M_{\boldsymbol{\theta}} \,|\, \mathcal{D}) = \arg\max_{\boldsymbol{\theta}} LL(M_{\boldsymbol{\theta}} \,|\, \mathcal{D}),$$

where

$$LL(M \,|\, \mathcal{D}) = \sum_{\mathbf{d} \in \mathcal{D}} \log_2 P(\mathbf{d}|M).$$

If the database does not contain missing values, then the likelihood of a Bayesian network is maximized by the (local) maximum likelihood estimates for the conditional probability tables, say $P(A \,|\, \mathrm{pa}(A))$, in the network:

$$\frac{N(A, \mathrm{pa}(A))}{N(\mathrm{pa}(A))}.$$

Bayesian Estimation

Let X be a binary variable (yes, no), and assume that we have performed a number of independent experiments out of which n turned up *yes* and m turned up *no*. Let θ be the probability for *yes*. Then, starting with the even prior distribution for θ, the posterior distribution is

$$f_p(\theta) = \mu\theta^n(1-\theta)^m,$$

where μ is a normalization constant. The Bayesian estimate for θ is $\frac{n+1}{n+m+2}$.

This result can be interpreted so that an even prior distribution corresponds to adding two virtual experiments to the data (one for *yes* and one for *no*) and then counting frequencies. The procedure generalizes to distributions over variables with more than two states.

Incomplete Data

- If the probability that a particular value is missing depends only on the observed values, then the data is said to be *missing at random* (MAR).
- If this probability is also independent of the observed values, then the data is said to be *missing completely at random* (MCAR).
- If the data is neither MAR nor MCAR, then the process that generated the missing data is said to be *nonignorable*.

The EM algorithm

To find an estimate for the maximum likelihood parameters when the data is incomplete, you may run the EM algorithm; note that you are guaranteed only to find a local maximum likelihood estimate.

1. Choose an $\epsilon > 0$ to regulate the stopping criterion.
2. Let $\boldsymbol{\theta}^0 = \{\theta_{ijk}\}$, where $1 \leq i \leq n$, $1 \leq k \leq |\mathrm{sp}(X_i)| - 1$, and $1 \leq j \leq |\mathrm{sp}(\mathrm{pa}(X_i))|$, be some initial estimates of the parameters (chosen arbitrarily).
3. Set $t := 0$.
4. Repeat:
   E-step: For each $1 \leq i \leq n$ calculate the table of expected counts:
   $$\mathop{\mathbb{E}}_{\boldsymbol{\theta}^t}[N(X_i, \mathrm{pa}(X_i)) \,|\, \mathcal{D}] = \sum_{\mathbf{d} \in \mathcal{D}} P(X_i, \mathrm{pa}(X_i) \,|\, \mathbf{d}, \boldsymbol{\theta}^t).$$
   M-step: Use the expected counts as if they were actual counts to calculate a new maximum likelihood estimate for all θ_{ijk}:
   $$\hat{\theta}_{ijk} = \frac{\mathbb{E}_{\boldsymbol{\theta}^t}[N(X_i = k, \mathrm{pa}(X_i) = j) \,|\, \mathcal{D}]}{\sum_{h=1}^{|\mathrm{sp}(X_i)|} \mathbb{E}_{\boldsymbol{\theta}^t}[N(X_i = h, \mathrm{pa}(X_i) = j) \,|\, \mathcal{D}]}.$$
   Set $\boldsymbol{\theta}^{t+1} := \hat{\boldsymbol{\theta}}$ and $t := t + 1$.
   Until $|\log_2 P(\mathcal{D} \,|\, \boldsymbol{\theta}_t) - \log_2 P(\mathcal{D} \,|\, \boldsymbol{\theta}_{t-1})| \leq \epsilon$.

The probabilities required in the E-step are easily calculated using junction tree propagation.

Adaptation

Adaptation through type variables: The second-order uncertainty can be characterized as uncertainty about which table out of $\mathbf{t}_1, \ldots, \mathbf{t}_m$ is the correct one for $P(A \,|\, pa(A))$.

Add a type variable T with states $t_1, \ldots, t_m$ and with A as child. The prior probability $P(t_1, \ldots, t_m)$ reflects your belief in the various tables. Put $P(A \,|\, pa(A), t_i) = \mathbf{t}_i$.

Whenever a case e has been processed, the probability $P(t_1, \ldots, t_n \,|\, e)$ is used as the new prior for the next case.

Fractional updating: Assume that the second-order uncertainty obeys both the global and local independence requirements. For each parent configuration π, choose a fictitious sample size n expressing the present certainty of $P(A \,|\, \pi)$. This yields a fictitious sample size $n_a = nP(a \,|\, \pi)$ for the configuration (a, π).

When a case has been processed, it yields $P(a, \pi \,|\, e)$. Add $P(a, \pi \,|\, e)$ to n_a. Thereby the sample is increased by $P(\pi \,|\, e)$.

Warning: fractional updating reduces the second-order uncertainty too quickly.

Fading: Instead of counting up with n_a, first multiply the counts for π by a fading factor. A fading factor q can be established from an effective sample size s^*

$$q = \frac{(s^* - P(\pi \mid e))}{s^*}.$$

The alternative model approach: If there is explicit uncertainty in the model – that is, if there are alternative models $M_1, \ldots, M_m$ – they can be weighted initially and run in parallel. After each case, the weights are modified.

Tuning

The set of parameters $\mathbf{t}$ open for modification; $\mathbf{x}(\mathbf{t})$ the current distribution in the model; $\mathbf{y}$ the target distribution.

1. Calculate $\mathbf{grad}\,\mathrm{dist}(\mathbf{x}, \mathbf{y})$ with respect to the parameters $\mathbf{t}$.
2. Give $\mathbf{t}_0$ a displacement $\triangle\mathbf{t}$ in the direction opposite to the direction of the gradient $\mathbf{grad}\,\mathrm{dist}\,(\mathbf{x}, \mathbf{y})\,(\mathbf{t}_0)$; that is, choose a step size $\alpha > 0$ and let $\triangle\mathbf{t} = -\alpha\,\mathbf{grad}\,\mathrm{dist}\,(\mathbf{x}, \mathbf{y})\,(\mathbf{t}_0)$.
3. Iterate this procedure until the gradient is close to $\mathbf{0}$.

We have

$$\frac{\partial}{\partial t}\mathrm{dist}\,(\mathbf{x}, \mathbf{y}) = \sum_i 2(x_i - y_i)\frac{\partial x_i}{\partial t}.$$

Because $P(e)(t) = \alpha t + \beta$, we know that $x_i(t)$ is the ratio of two linear functions, and the partial derivatives can be calculated for all parameters through two propagations (Chapter 4).

6.6 Bibliographical Notes

The characterization of the different ways in which data may be missing/incomplete was suggested by Rubin (1976). With outset in incomplete data, the EM algorithm was proposed by Dempster *et al.* (1977) for learning maximum likelihood parameter estimates. Green (1990) described how the EM-algorithm can be used to find penalized maximum likelihood estimates, and Lauritzen (1995) showed how the junction tree architecture can be exploited in calculating the expected counts in the E-step of the algorithm.

When data arrives sequentially, the probability parameters can be adapted using fractional updating (Titterington, 1976). In some cases, however, fractional updating may overestimate the sample size and an improved version of the algorithm (known as incremental updating) was proposed by Spiegelhalter and Lauritzen (1990). Later this algorithm was extended by Olesen *et al.* (1992) to also allow for fading.

The tuning method was proposed by Jensen (1999), based on work by Russell *et al.* (1995) and Castillo *et al.* (1996).

6.7 Exercises

Exercise 6.1. Consider Example 6.1. Prove that the maximum likelihood estimate for the model given the data is $\theta = 0.8$.

Exercise 6.2. In the thumbtack experiment, let the nonnormalized prior distribution for θ be

$$f(\theta) = \begin{cases} \theta \text{ if } \theta \leq 1/2 \\ (1-\theta) \text{ if } 1/2 \leq \theta \leq 1 \end{cases}$$

(i) What is the normalization constant?

We have performed one experiment resulting in *up*.

(ii) What is the functional part of f_p, the posterior distribution for θ?
(iii) What is normalization constant for f_p?
(iv) What is the posterior Bayesian estimate?

Exercise 6.3. Consider the data in Table 6.1 and a Bayesian network consisting of two nodes T_1 and T_2, with T_1 being a parent of T_2. What are the maximum likelihood parameter estimates for the model given the data? What are the Bayesian parameter estimates for the model given the data?

Exercise 6.4. Prove the distribution part of Theorem 6.1.

Exercise 6.5. Establish a Bayesian estimate of the conditional probability $P(\mathbf{a}|\mathbf{b})$ from the counts in Table 6.1.

Exercise 6.6. Characterize the type (MAR, MCAR, or nonignorable) of missingness that underlies the database for the variables A and B described in the beginning of Section 6.2.

Exercise 6.7. Without taking the size of the database into account, when would it be safe to throw away cases with missing values, i.e., should the data be MAR, MCAR, or neither of the two?

Exercise 6.8. ^{E}

(i) Update the remaining probabilities in Example 6.2.
(ii) Use the updated probabilities to perform another iteration of the EM-algorithm.

Exercise 6.9. Refer back to the example of EM parameter estimation in Example 6.2. What are the estimated parameters after a full iteration? And after two full iterations? What are the maximum likelihood parameter estimates using only the complete cases? What are the Bayesian parameter estimates using only the complete cases?

Exercise 6.10. ^{E} Consider the model in Exercise 3.28.

(i) What happens when you adapt to the following sequence of (A, B) states: $\langle(n, y)(n, y)(y, n)\rangle$?
(ii) Process a sequence of cases with $A = y$ in which the states of B are

$$(n, y, n, y, y, n, n, y, y, y).$$

What are your beliefs in the experts now, and what is $P(B \mid A)$?

Exercise 6.11. You have the same model as in Exercise 6.10, but $P(B \mid A)$ is the one in Table 6.9.

$B \setminus A$	y	n
y	0.75	0.4
n	0.25	0.6

Table 6.9. Table for Exercise 6.11.

For $P(B \mid A = y)$, you have an initial sample size of 12.

(i) Perform fractional updating from the sequence in Exercise 6.10 (*iii*).
(ii) Perform fractional updating on the same sequence but with fading factor 0.9.

Exercise 6.12. The network from Example 6.4.1 in its initial state has sample sizes $s_t = 25$, $s_s = 10$, and $s_u = 25$ for the three parameters. It now receives 20 cases with $C = c$ out of which 10 have $A = a$ (the rest have $A = \neg a$). For the cases with $A = a$, all cases have $B = b$, and in the rest, 4 had $B = \neg b$.

1. Adapt the network without fading.
2. Adapt the network with effective sample sizes 25, 10, and 25 for t, s, and u, respectively.
3. Adapt the network to the same cases but without the information on B.

Exercise 6.13. Perform the calculations of Example 6.4.1 by use of a direct representation of the parameters t, s.

Exercise 6.14. Assume that in Example 6.4.1 we require $P(A \mid c) = (0.5, 0.5)$, and assume that $t = 0.6$ is fixed. Use the technique from Example 6.4.1 to tune the parameters s and u.

Exercise 6.15. Let D be a child of C, and let C have parents A and B, all variables being binary. $P(A)$ and $P(B)$ have even distributions; $P(D \mid c) = (0.1, 0.9)$, $P(D \mid \neg c) = (0.6, 0.4)$, and $P(c \mid A, B)$ are as specified in Table 6.10. Tune the parameters t, s to the prescribed behavior $P(a \mid d) = 0.8$.

$B \setminus A$	a	$\neg a$
b	$1 - ts$	$1 - s$
$\neg b$	$1 - t$	0

Table 6.10. The conditional probability table $P(C = c \mid A, B)$ for Exercise 6.15.