Chapter 4

Parameter Learning

This chapter focuses on learning the parameters of a Bayesian network from incomplete data. Iterative search methods such as Expectation-Maximization (EM) perform a local search, and therefore do not necessarily find the global maximum. To avoid getting stuck in poor local maxima, many improvements have been proposed, typically based on changing the score function or the search procedure. Based on the voting perspective from the previous chapter, we investigate how the use of a quantized parameter space can help to avoid poor local maxima, and speed up learning. At the same time, it can increase the robustness against overfitting when training on small data sets, without requiring parameter priors. We show how this approach can easily be incorporated in a general EM procedure.

4.1 Introduction

Parameter learning for Bayesian networks is the process of finding adequate parameter values, given a fixed network structure. Typically, the dependency structure can be found with the help of human domain experts. Finding precise parameter values is difficult for humans, so we often use methods to learn the parameters from a data set of representative cases. These algorithms typically attempt to maximize some score. This maximization is not trivial, since the data cases are often incomplete.

Typical scores are data likelihood and model posterior probability. In the case of a complete data set, finding the maximizing model is trivial. However, no direct solution exists for the maximum likelihood or model posterior parameter values from incomplete data. In these cases, we require a search method that iteratively improves the score of the Bayesian network. One such search algorithm is the expectation-maximization (EM) algorithm [DLR77, Lau95]. A major disadvantage of these methods is that they perform a local search, and thus find maxima
in the locality of the starting point of the search. These local maxima can be quite poor.

Approaches to overcome this disadvantage have been proposed. The most common solution is to use many restarts from random initializations, and keep the best solution that was found. Some more advanced approaches focus on changing the search procedure to allow certain non-improving steps [GL97, KGV83], and others on changing the search surface or score function to guide the search to better maxima [UN98, ENFS02]. Section 4.2 gives a more detailed description of some approaches. One downside of methods capable of finding models that fit the training data very well, is that they become more likely to overfit on small data sets. A small data set is more likely to badly represent the true distribution. Thus, if a model is learned that fits the training data very well, it might not be a good generalization of the true application domain.

In this chapter we investigate a third type of approach based on a quantization of the parameter values. In other words, we allow the parameters to take only a finite number of possible values, instead of any value from a continuous space. The intuition behind this is that by choosing an appropriate quantization of the space, we are discarding unnecessary and potentially harmful details. This simplifies the search surface and can smooth out local maxima. The learning procedure can consist of two phases, one phase that searches in the quantized space, and a second phase that refines the solution in the full parameter space. Higher precision details are learned separately, after a good solution in the quantized space has been found. Which quantization we choose and the reasons why certain details can be smoothed out is further discussed in Section 4.2.1.

A quantized parameter space approach can have several advantages. Firstly, the number of allowed parameter values directly influences the number of local maxima, which is elaborated in see Section 4.1.1. Less local maxima implies more stable results, and less need for random restarts, speeding up the search procedure. Secondly, in a space with less possible values, reaching any maximum is likely to take less steps. Hence, convergence is likely to be reached faster. Thirdly, the quantization can facilitate to avoid very extreme parameter values, which are often unwanted. Similar to smoothing CPTs by using parameter priors, this can help prevent overfitting.

This seems to imply that the quantization approach can be effective in the case of small training data sets. In such a context, typically many local maxima are present that do not correspond to maxima in the ‘true’ space. Quantization can lower the number of local maxima and disallowing extreme parameter values. Therefore the quantized search becomes more robust against overfitting. It focuses on learning a rough model first, without committing itself to finding precise values in the first phase of learning.

Most approaches that aim to counteract overfitting, such as using priors on parameters or adding a penalization term to the score function, use problem-specific prior or external knowledge about which parameter values are (un)wanted.
4.1 Introduction

Our approach does not require such knowledge.

According to the perspective from the previous chapter, even a coarse or quantized model can give accurate state estimation, under certain conditions. For this we need an inference algorithm that can handle the coarse representation. Therefore, an interesting experiment is to couple quantized learning to the voting algorithm from Chapter 3. Quantized learning can produce a model that satisfies the accuracy conditions, but has precise parameter values that are not optimal. If the voting algorithm can achieve good state estimation accuracy despite this sub-optimality, this is a validation for our perspective on model grounding.

4.1.1 Rationale

To illustrate how the number of local maxima is influenced by the quantization level of the search space, we performed a small experiment. We took an X-shaped network, consisting of a central node with 2 parents and two child nodes, as shown in Figure 4.1(a). All variables had two states. The conditional probabilities were initialized to random values, and a data set was sampled from the network. The value of the central node was made hidden in all data cases.

If we had to learn the network parameters from this data set, we would be searching for the optimal values of the CPTs of the central node and its two children. This amounts to 8 parameters, as the other 8 are their complements. In the quantized space, parameters can only take values that are a multiple of a certain step size. For example, values \( \{0.05, 0.1, \ldots, 0.95\} \) for a step size of 0.05. To get an estimate of the number of local maxima under different quantization levels, we performed a large number of random-restart hill-climbs. This simple hill climbing considers all neighbors of the current parameter estimate in the quantized space and selects the one with the greatest data likelihood.

We stored the location of every hilltop that was found in this way, and counted how often certain maxima were found. Varying the step size and repeating the process led to the results shown in Figure 4.1(b). A clear trend can be seen; a higher step size leads to less local maxima. In other words, a coarser quantization of the space contains less local maxima.

A key question is whether a maximum in the quantized space is close, in terms of quality, to a maximum in the full space. The answer depends on the particular quantization. Here, the voting perspective from the previous chapter comes in. Under that perspective, we defined parameters to be accurate if they are support-equivalent to the true distribution (they produce the same supports, Definition 5). Large intervals of parameter values are support-equivalent. Within each interval, the CPTs are similar with respect to state estimation quality.

The approach in this chapter is to have one quantized point per support-equivalence interval. The closeness within each interval is therefore not defined in terms of likelihood, but in terms of state estimation accuracy. The quantized learning phase searches among the quantized CPTs for the one that is support-
equivalent to the true distribution, which is represented by the training data set. In the second learning phase, the quantized maximum is refined by a search in the complete parameter space. If the second phase converges to a maximum that lies in the same support-equivalence interval, this maximum is also support-equivalent to the true distribution. Therefore, this maximum is accurate in a state estimation sense, and close to the quantized maximum.

In other words, the quantized value in each support-equivalence interval is close to the other values in the interval, in a state estimation sense. Therefore it is a good representative of the values in the interval, and a search in the quantized space makes sense.

See Figure 4.2 for an illustration. The solid curve represents an example likelihood surface, for different parameter values. The circles are the quantized values. The vertical dotted lines bound the intervals of values that are support-equivalent. The horizontal dashed lines show the likelihood of parameter values corresponding to the circles. One would like to find the parameter value that corresponds to the global likelihood maximum (the triangle in the figure). In this figure, the maximum likelihood parameter value can be found by starting a search from the quantized maximum (the middle circle). Hence, the quantized maximum will be support-equivalent to the global maximum.
4.2 Quantized EM

In Section 2.3.3, the standard EM algorithm for learning the parameters of a Bayesian network was discussed. A disadvantage of the algorithm is that it only performs a local search of the parameter space, and any found maximum could be a local maximum of quite poor quality. Several improvements for avoiding poor local maxima have been proposed. Two types can be distinguished, one that alters the search procedure in order to escape local maxima, and one that adapts the likelihood landscape to increase the chance of ending up in a (near) global peak.

**Changing the Search Procedure**

The most trivial method for improving the quality of the output of the EM algorithm is to do multiple restarts from random initializations of the parameters. The thought behind this is that even though EM is doing a local search, at least one of the random starting positions should be in the locality of a (near) global solution. This is generally considered the baseline method.

Tabu search is a local iterative improvement method that puts constraints on the space of possible parameter values in the next iteration [GL97]. The most basic version maintains a set of states that the algorithm is not allowed to visit in the next iteration. Simulated annealing is a modification that allows downhill steps with some probability [KGV83]. This probability becomes smaller as the quality of the current solution improves and as a temperature parameter decreases.

---

Figure 4.2: Schematic view of a likelihood landscape (solid curve) with support-equivalent parameter intervals (dotted boundaries).
over time. These two methods are not trivial to apply to BN parameter learning and are not often used.

**Changing the Score Function**

The deterministic annealing approach attempts to find global maxima by smoothing the likelihood landscape [RGF92, UN98]. The degree of smoothing is regulated by a temperature parameter, which is decreased over time. This lets the landscape gradually return to its original unsmoothed shape. The thought behind this method is that the global maximum of the maximally smoothed landscape is easily found. By making the assumption that the global maximum at each annealing step is close to the global maximum of the previous step, one can show that the final outcome is the global maximum of the unannealed landscape. [UN98] shows that the EM algorithm can be adapted to this method by a simple change to the E-step.

Data perturbation is a method that influences the likelihood landscape by assigning weights to data cases [ENFS02]. Changing the weights in a sensible manner can help escape poor local maxima. Higher weights are given to the data cases which have low likelihood under the current model. Again a temperature parameter is used to make the distribution converge to a uniform distribution over time. The idea is that poor maxima are based on outliers in the data set, whereas good maxima are based on common cases in the data. This method can also be implemented by changing the E-step.

### 4.2.1 Parameter Space Quantization

The quantization approach can be seen as a third type. It does not alter the search procedure, as it will still take maximizing steps. Nor does it change the score function. The choice for the particular quantization is based on the perspective of voting on factor supports from the previous chapter.

One of the consequences of this perspective is that the state estimation of a variable is strongly dependant on the factor supports it receives from its neighboring nodes. If all factors support the true state of a variable, the state estimation will be accurate. It was also observed that many different CPTs can cause the same supports, since the supports are given by inequality relations.

These inequality relations define intervals of parameter values which are similar in terms of state estimation accuracy. As explained in Section 4.1.1, a good quantization of the parameter space is one where each interval is represented by exactly one quantized value, and the quantized value is representative of all values in its interval, possibly including the global maximum.

To achieve this quantization, we map all CPTs in an interval to one single quantized CPT. In other words, all support-equivalent CPTs (recall Definition 5) map to one point in the quantized space. We first define the following function.
For any node $X$, it returns the combined state of the parents of $X$, which are most likely given $X = x_j$:

$$
\text{support}(x_j) = \arg \max_k p(X = x_j | \text{Pa}_x = k). \quad (4.1)
$$

The CPT of a node $X$ consists of $J$ by $K$ entries denoted by $\theta_{ijk}$ (as shown in Figure 4.3). Each child state $j$ corresponds to one row, and each combined-parent state $k$ to one column. The state support($x_j$) is thus the maximum entry in a row.

The mapping consists of two parts. First, the supported parent states should remain the maximum in its row. This ensures that the supports do not change and we stay in the same interval. However, these entries are replaced by a ‘high’ value denoted by $\alpha$. Second, we smooth out everything else, by replacing all other entries by a ‘low’ value $\beta_k$, such that columns still sum to 1. In this way, any CPT in an interval is changed into the same quantized CPT representing this interval.

As for the value of $\alpha$ and the $\beta_k$, there are several restrictions. There can be between 0 and $J - 1$ $\alpha$’s in a column. No probabilities should be set to 0, since extreme values are unwanted. Columns must sum to 1, and $\alpha$’s must be the maximum entries in each row. The maximum value a $\beta_k$ can take is $1/J$, namely when a column contains no $\alpha$’s. Since $\alpha$ must be greater than any $\beta_k$, it follows that $1/J < \alpha$. In the case that a column contains $J - 1$ $\alpha$’s, the column should still be able to sum to 1. Hence, $(J - 1)\alpha < 1$, and thus $\alpha < 1/(J - 1)$. $\alpha$ can be freely chosen from that interval.

The mapping from normal CPT parameters $\theta_i$ to quantized $\tilde{\theta}_i$ for some variable $X$, is then as follows, for all $j$ and $k$:

$$
\tilde{\theta}_{ijk} = \begin{cases} 
\alpha & \text{if } k = \text{support}(x_j) \\
\beta_k & \text{otherwise}
\end{cases} \quad (4.2)
$$

$\beta_k$ depends on the number of $\alpha$’s in its column $k$. If $|\alpha|_k$ denotes the number of

<table>
<thead>
<tr>
<th>Pa_i</th>
<th>1</th>
<th>...k...</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>j</td>
<td>...θ_{ijk}...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Algorithm 1: Quantized EM

/* quantized phase */
while not converged do
    E[Count] ← E-step;
    \( \Theta \) ← M-step;
    \( \Theta \) ← Replace each \( \theta_i \in \Theta \) with \( \tilde{\theta}_i \) using (4.2);
end

/* refining phase */
while not converged do
    Regular EM iterations;
end

\( \alpha \)'s in column \( k \), then \( \beta_k \) is given by

\[
\beta_k = \frac{1 - \alpha |\alpha|_k}{J - |\alpha|_k} \tag{4.3}
\]

From the interval for \( \alpha \), it follows that \( \alpha > \beta_k > 0 \) for all \( k \), and thus the supports are unchanged. Table 4.1 shows some examples.

The quantized algorithm (Algorithm 1) consists of two phases. The first phase performs a search through the quantized space. After it has converged, the second phase continues with a regular EM search to refine the solution. The quantized search works as follows: After each EM iteration, the solution is mapped to its corresponding quantized value. This new CPT is then used for the next iteration. Essentially, it is a search for the best placement of the \( \alpha \)'s in the CPT, corresponding to a search for the best supports. From a quantized point, EM will try to make an up-hill step. If this step is large enough, a new interval will be reached (c.f. Figure 4.2), and subsequently a new quantized point.

Another observation from the previous chapter was that the more children a node has, the less important do the actual values in the children’s CPTs become. Since each child has an independent support for a particular state, the redundancy decreases the sensitivity to (small) parameter changes, with respect to state estimation. It makes it more likely to satisfy the conditions that make the majority voting exact. This also implies that all CPTs mapping to the same quantized CPT will result in a state estimation quality that is close to that of the quantized CPT. Thus, the quantized CPT becomes a better representative of its interval. Therefore, the quantization is likely to work best on networks where nodes have many conditionally independent children.

4.2.2 Convergence Properties

The convergence properties of the algorithm require some discussion. One important issue to note is that the current definition of the algorithm does not strictly
4.2 Quantized EM

<table>
<thead>
<tr>
<th>y1</th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>⇒</th>
<th>y1</th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.7</td>
<td>0.6</td>
<td>0.1</td>
<td></td>
<td>0.8</td>
<td>0.5</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>y2</td>
<td>0.3</td>
<td>0.4</td>
<td>0.9</td>
<td></td>
<td>y2</td>
<td>0.2</td>
<td>0.5</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>y1</th>
<th>x1</th>
<th>x2</th>
<th>x1</th>
<th>x2</th>
<th>⇒</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.40</td>
<td>0.20</td>
<td>0.50</td>
<td>0.35</td>
<td></td>
</tr>
<tr>
<td>y2</td>
<td>0.32</td>
<td>0.10</td>
<td>0.44</td>
<td>0.18</td>
<td></td>
</tr>
<tr>
<td>y3</td>
<td>0.28</td>
<td>0.70</td>
<td>0.06</td>
<td>0.47</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1: Some examples of the quantization. (top) CPT \( p(Y|X) \), \( \alpha \) is chosen as 0.8. (bottom) CPT \( p(Y|X,Z) \), \( \alpha = 0.45 \).

improve the solution quality in each EM-loop. In the variational view of EM [NH98] the free energy need not be maximized in every E or M-step. An increase is sufficient for guaranteeing convergence (as explained in Section 2.3.3).

In a regular M-step, the free energy is maximized within the full parameter space. To maximize in the quantized space, we should choose the quantized CPT resulting in the highest free energy. Since the free energy is proportional to the KL-divergence, we could achieve this maximization by choosing the quantized CPT with the lowest divergence to the maximizing CPT in the full space.

The mapping (4.2) defined in the previous section does not necessarily achieve this. There are (rare) situations in which the KL-divergence is not minimized. The reason for this is the fact that \( \beta_k \) depends on the number of \( \alpha \)'s in a column. See Figure 4.4 for an example. During learning this could cause the algorithm to switch back and forth between two solutions.

To solve this problem, convergence can be guaranteed by performing an (exhaustive) search among the possible quantized CPTs. The number of CPTs that needs to be considered is small, and since the search only needs to be done once per CPT per M-step, the computational burden is insignificant compared to the E-step. Another fast method is to directly compare the new quantized CPT with the one found in the previous iteration, in terms of divergence to the point in the full space. In case the new CPT does not decrease the divergence, the old one can be kept.
4.2.3 Experiments I

An experiment was conducted to compare our quantized variant of EM with regular EM and EM with parameter smoothing through parameter priors (as explained in Section 2.3.4). Like quantized EM, smoothed EM splits the learning in two phases. The first phase has uniform penalization terms added to the counts, thus smoothing the CPTs, and the second is done without penalization. As we mentioned in the introduction, the quantized algorithm’s characteristics suggest a robustness against over-fitting and the local maxima present when learning from small data sets.

We applied the algorithm on the Insurance, Hailfinder, and Water Bayesian networks. From each BN we sampled a training data set and test data set, using forward sampling. The training data set consisted of 100 samples, and the test set of 1000 samples. We made several central nodes hidden, and randomized the parameters of the network. New parameter values were then learned from the data set using the three different algorithms, using the same convergence threshold. We measured the quality of the new parameters using (i) the average log-likelihood of the test set, and (ii) the state estimation accuracy (percentage of correct state estimations) of one of the variables on the test set. The randomization of the CPTs and relearning was repeated 50 times.

The results averaged over these 50 runs are shown in Table 4.2. As can be seen, quantized EM performs at least as good as the other two methods on the three networks in likelihood and accuracy. With respect to the log-likelihood, quantized EM performs better on the Insurance and Hailfinder networks. The very low improvement in the state estimation accuracy can be explained through the perspective in the previous section; state estimation can be accurate for many different parameter values as long as the supports are adequate. Apparently the other learning methods also learn parameter values that result in adequate supports.

---

<table>
<thead>
<tr>
<th></th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_1$</th>
<th>$x_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y_1$</td>
<td>0.60</td>
<td>0.05</td>
<td>$y_1$</td>
<td>0.4</td>
<td>0.3</td>
<td>$y_1$</td>
</tr>
<tr>
<td>$y_2$</td>
<td>0.25</td>
<td>0.20</td>
<td>$y_2$</td>
<td>0.4</td>
<td>0.3</td>
<td>$y_2$</td>
</tr>
<tr>
<td>$y_3$</td>
<td>0.15</td>
<td>0.75</td>
<td>$y_3$</td>
<td>0.2</td>
<td>0.4</td>
<td>$y_3$</td>
</tr>
</tbody>
</table>

Figure 4.4: (a) Maximizing CPT in full space. (b) Subsequent support-equivalent quantized CPT obtained through (4.2). (c) Alternative quantized CPT. KL-divergence between (a) and (b) is 0.38. Between (a) and (c) it is 0.36.
4.2 Quantized EM

Table 4.2: (a) Average log-likelihood and (b) state estimation accuracy on the test set for different networks and methods.

<table>
<thead>
<tr>
<th>network</th>
<th>quantized EM</th>
<th>smoothed EM</th>
<th>regular EM</th>
</tr>
</thead>
<tbody>
<tr>
<td>water</td>
<td>-16.28</td>
<td>-16.28</td>
<td>-16.28</td>
</tr>
<tr>
<td>hailfinder</td>
<td>-35.94</td>
<td>-36.03</td>
<td>-36.14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>network</th>
<th>quantized EM</th>
<th>smoothed EM</th>
<th>regular EM</th>
</tr>
</thead>
<tbody>
<tr>
<td>insurance</td>
<td>0.63</td>
<td>0.62</td>
<td>0.62</td>
</tr>
<tr>
<td>water</td>
<td>0.78</td>
<td>0.78</td>
<td>0.78</td>
</tr>
<tr>
<td>hailfinder</td>
<td>0.41</td>
<td>0.41</td>
<td>0.40</td>
</tr>
</tbody>
</table>

Table 4.3: Average number of iterations required by three EM algorithms on different networks. The ‘+’ sign separates the average number of iterations for each learning phase of the quantized and smoothed methods.

<table>
<thead>
<tr>
<th>network</th>
<th>quantized EM</th>
<th>smoothed EM</th>
<th>regular EM</th>
</tr>
</thead>
<tbody>
<tr>
<td>insurance</td>
<td>4.2 + 7.1</td>
<td>17.0 + 15.0</td>
<td>20.0</td>
</tr>
<tr>
<td>water</td>
<td>3.0 + 3.0</td>
<td>4.7 + 3.1</td>
<td>6.0</td>
</tr>
<tr>
<td>hailfinder</td>
<td>2.9 + 3.0</td>
<td>3.3 + 8.4</td>
<td>9.3</td>
</tr>
</tbody>
</table>

Computational Cost

The computational cost per EM-iteration is of the same order of complexity for the three algorithms. A clear difference can be observed with respect to the number of iterations needed for convergence, though. Table 4.3 shows the average number of iterations for the three algorithms, given equal convergence thresholds. For the quantized and smoothed variants, the average number of iterations used by their two individual learning phases are shown, separated by a ‘+’. The quantized variant in general requires less iterations than the other two methods. The fact that quantized EM is faster than regular EM even though it consists of two phases is interesting. Apparently, the quantized phase guides the search very quickly to the neighborhood of a (local) maximum, requiring much less iterations in its second phase.

4.2.4 Experiments II

In another experiment, we combine the quantized learning phase with the voting algorithm from the previous chapter. From the experiments above, we see that
Table 4.4: Combining quantized learning with the voting algorithm. Table (a) shows the state estimation accuracy for different networks and inference methods. Table (b) shows the similarity between the voting algorithm applied on the quantized network, and exact inference on the original network.

The quantized learning phase requires very few iterations to converge. In the previous chapter we said that the state estimation can be very accurate as long as the CPTs result in the correct supports. These supports are learned by the quantized phase. Therefore, combining the two methods could result in a very fast algorithm that still scores a high state estimation accuracy.

To see under which conditions this approach works best, we apply the experiment on a number of networks with varying CPTs. The networks were simple trees with a branching factor of 4 and 3 layers, so with 1 root node and 16 leaf nodes. All variables have 3 states, and the CPT between a variable $A$ and $B$ is characterized by $p(b_1|a_1) = p(b_2|a_2) = p(b_3|a_3) = s$, and the other values are equal to $(1 - s)/2$. Thus, $s$ controls the ‘spikiness’ of the CPTs. Spikier CPTs model stochastic processes that are closer to deterministic relations, which are easier to find by learning algorithms.

We let this $s$ take on values $\{0.4, 0.6, 0.7, 0.8\}$. From each network we sampled a data set with the middle layer hidden. This set was then used to train a new network with random CPTs. We performed voting and exact state estimation on the network that was obtained after only performing the quantized phase of quantized EM. This was compared to exact inference on the original network (which generated the data set). We recorded the state estimation accuracy and similarity (as defined in Section 3.4.2) on the root node, averaged over 20 runs.

Table 4.4 shows the results. We can see that for the more spiky networks, using the voting algorithm on the quantized network results in a state estimation accuracy that is equal to the accuracy of exact inference on the original network. The similarity is also high for the more spiky networks.

The simple trees used in this experiment are well suited for the quantized and voting approaches. For comparison, we also applied the experiment on two
### 4.3 Improving Quantized EM

In the previous section we showed that quantized EM can be faster than regular EM, and find higher likelihood parameters. However, quantized phase of quantized EM can be further improved. To illustrate the significance of the quantized phase we conducted the following small experiment: We created a simple polytree BN with random parameters and sampled a training data set and test data set from it. Some of the variables were made hidden in both sets. Both regular EM and quantized EM were then used to learn the parameters of a new BN, starting from random initialization. The results of 50 random restarts are shown in Figure 4.5. The figure shows a histogram of the test set log-likelihoods of the trained networks. We see that quantized EM (black bars) on average finds better models than regular EM (gray bars). Quantized EM also has a lower variance in its output than regular EM, indicating that fewer local maxima exist in the

Table 4.5: Combining quantized learning with the voting algorithm. Table (a) shows the state estimation accuracy for different networks and inference methods. Table (b) shows the similarity between the voting algorithm applied on the quantized network, and exact inference on the original network.

<table>
<thead>
<tr>
<th></th>
<th>network</th>
<th>inference</th>
<th>conv. tree</th>
<th>insurance</th>
</tr>
</thead>
<tbody>
<tr>
<td>quantized</td>
<td>voting</td>
<td>0.32</td>
<td>0.54</td>
<td></td>
</tr>
<tr>
<td>quantized</td>
<td>exact</td>
<td>0.32</td>
<td>0.28</td>
<td></td>
</tr>
<tr>
<td>original</td>
<td>exact</td>
<td>0.41</td>
<td>0.82</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.5: Combining quantized learning with the voting algorithm. Table (a) shows the state estimation accuracy for different networks and inference methods. Table (b) shows the similarity between the voting algorithm applied on the quantized network, and exact inference on the original network.

networks for which it is less suited. The first is a ‘converging’ tree, namely a regular tree with all arc directions reversed. In the corresponding equivalent factor graph, each factor node has many neighbors, and each variable node only 2 neighbors. The second network is the Insurance network, which is quite multiply connected and does not have many conditionally independent fragments. The results are shown in Table 4.5. The scores are much lower, as expected.

Summarizing, combining quantized learning with the voting algorithm works well in a particular type of network, namely trees with spiky CPTs. On these networks, we can use a fast learning algorithm to learn a coarse, quantized model, and then use a fast inference algorithm to get accurate state estimations.
Figure 4.5: Histogram of test results comparing regular EM (gray) and quantized EM (black). The dashed line corresponds to the network trained from the golden quantized initialization.

quantized space.

However, Figure 4.5 also shows that there is still quite high variance among the models found by the quantized method. Apparently, the quantized search space still contains some local maxima. For comparison, we ran a third algorithm, performing a regular EM run using the ‘golden quantized network’ as initialization. This golden quantized network is obtained by mapping all CPTs of the generating network to quantized CPTs, using (4.2). This algorithm leads to a network whose log-likelihood (indicated by a dashed line) lies within the right-most bin (−4.35) of the histogram. In other words, if the quantized phase would find the golden quantized parameters, the refining phase is likely to lead to a good solution. Improving the quantized search phase will therefore significantly improve the overall algorithm. To achieve this, we propose to use the data perturbation method already mentioned in Section 4.2.

4.3.1 Data perturbation

As mentioned before, data perturbation assigns weights to data cases. This general approach is readily applicable to the EM algorithm. It can also easily be combined with the quantized EM search. The only change takes place in the E-step. If \( w_x \) denotes the weight for case \( x \) then the E-step becomes:

E-step (Data perturbation) Given a set of weights \( w \), for each \( i, j \) and \( k \),
compute:

\[
E[\text{Count}(N_i = j, P_a_i = k)] = \sum_{x \in D} w_x \cdot p(N_i = j, P_a_i = k | x; \Theta) \tag{4.4}
\]

The weights are regularly changed, for example when the algorithm converges, or after a fixed number of iterations. Changing the weights can help to escape from poor local maxima (also in the quantized space). In [ENFS02], the authors propose two methods for changing the weights. The first is to change the weights to a random Dirichlet distribution. A temperature parameter is used to make the distribution converge to a uniform distribution over time. A second method, that reported better performance, is to give higher weights to the data cases which have low probability under the current model. The thought behind this is that poor maxima are based on outliers in the data set, whereas good maxima are based on common cases in the data. If the weights are uniformly distributed, the algorithm is equivalent to regular EM.

A parameter \( \tau \) is used to represent the temperature. A common method to cool down \( \tau \) is to multiply it at each time step with some decay value \( \delta < 1 \). Thus, the temperature at time \( t+1 \) is given by: \( \tau^{t+1} = \tau^t \delta \).

Random reweighting samples the new weights from a Dirichlet distribution with a parameter \( \beta = 1/\tau \) that corresponds to the uniformness of the weights vector. A reweighting scheme that reported better performance was adversarial reweighting. It works by punishing data cases that have a high probability under the current model. This can be achieved by computing the gradient of the weighted log-likelihood score with respect to the weights, and taking a step in the negative direction. In order to converge to a uniform weight vector \( w^0 \) and decrease the weight changes (from \( w^t \) to \( w^{t+1} \)) over time, two extra terms are added to the score before computing the gradient. The new score function is as follows:

\[
\text{Score}_w + \beta KL(w^{t+1} \| w^0) + \gamma KL(w^{t+1} \| w^t)
\]

and the following reweighting equation is obtained after computing the gradient [ENFS02]:

\[
w^{t+1}_x = \alpha^{t+1} (w^0_x)^{\beta/\gamma} (w^t_x)^{\beta/\gamma} e^{-\frac{\eta}{\beta+\gamma} \left( \frac{\partial \text{Score}_w}{\partial w_x} \right)_{w^t_x}}.
\]

\( \alpha \) is a normalizing constant, and the other three factors control the weight cooling, the smoothness of weight changes, and the penalizing, respectively. \( \beta \) and \( \gamma \) correspond to the importance of the terms that decrease the distance to the uniform \((w^0)\) and previous \((w^t)\) weights, respectively. \( \eta \) controls the size of the gradient step. \( \beta \) and \( \gamma \) should be set proportional to \( 1/\tau \), making the penalization more important over time. We apply the data perturbation approach on the quantized EM algorithm, and the result is Algorithm 2.

One final issue involves the tuning parameters \( \beta, \gamma, \eta, \) and \( \delta \). \( \delta \) strongly influences the total running time of the algorithm. Letting \( \tau \) decay slower will
Algorithm 2: Perturbed Quantized EM

/* perturbed quantized phase */

while $\tau > \tau_{\text{min}}$ do
    Decay $\tau$;
    $w \leftarrow [\beta \gamma \eta \Theta]$ Reweighting;
    while not converged and maximum iterations not reached do
        $E[\text{Count}] \leftarrow [w]$ E-step;
        $\Theta \leftarrow$ M-step;
        Replace each $\theta_i \in \Theta$ with $\tilde{\theta}_i$ using (4.2);
    end
end

/* refining phase */

while not converged do
    Regular EM iterations;
end

generally result in a better model. Values between 0.8 and 0.95 are typically used. The relation between $\beta$ and $\gamma$ determines the importance of moving towards uniformity and the importance of making small changes, respectively. Typically, one would value the convergence to uniformity more, so the values could for example be $\beta = 5/\tau$ and $\gamma = 1/\tau$. The value of $\eta$ relative to $\beta$ and $\gamma$ controls the ‘spikiness’ of the new weight distribution. A reasonable value is $\eta = \tau(\beta + \gamma)$. These values were found after initial testing. It should be noted that the sensitivity of the outcome to the precise values of these tuning parameters was low.

4.3.2 Experiments III

In this section we compare perturbed quantized EM to quantized EM, regular EM, and EM with data perturbation. The algorithms are applied to learning the parameters of the Alarm and Insurance network. We sampled a test data set and train data sets of different sizes from the original network, and made 5 central nodes hidden. The three algorithms were applied to 50 random initializations of the networks’ CPTs. We varied the train set size so we could also investigate the robustness against overfitting. The test set size was fixed at 2000 cases. We recorded the average log-likelihood on the test and train data sets, and the state estimation accuracy on one of the variables for the test set.

Figure 4.6, Figure 4.7, and 4.8 show the outcome of the experiment in terms of log-likelihood and state estimation accuracy. On the training data set, perturbed EM consistently scores the highest likelihood of all four methods. However, it also suffers from overfitting on the smaller data sets, scoring the lowest test set likelihood. Perturbed quantized EM scores lower on the training data set, while
reaching the highest test set likelihood for smaller training sets. This shows that it is robust against overfitting. Herein lies one the strength of the algorithm; it focusses on getting the CPTs accurate (support-equivalent to the true distribution), before fine-tuning the solution.

### 4.3.3 Computational Cost

Again, the computational cost per EM iteration is of the same order of complexity for all three algorithms, as multiplication with the weights requires relatively few extra computations. The reweighting itself requires almost no time. However, applying the temperature cooling scheme causes more EM iterations. Table 4.6 shows the average number of iterations used on the train set of size 800. Both perturbed methods require more iterations than regular EM due to the temperature decay loops. The difference between the perturbed methods is caused by the faster convergence of the quantized variant. A maximum can be reached faster in the quantized space, which was also indicated by the experiments from the previous section.

### 4.4 Conclusion

This chapter presented a novel approach to parameter learning for Bayesian networks. It is based on the use of a quantized parameter space. The voting perspective from the previous chapter tells us that under certain conditions, large intervals of parameter values have similar state estimation accuracy. For this reason, we have chosen a quantization where each interval is represented by one quantized point.

The mapping from the full space to the quantized space thus maintains support-equivalence, while smoothing out all other details. The resulting algorithm (Algorithm 1) first searches for a maximum in the quantized space. After a (local) maximum is found, one regular EM run (in the full space) is used to refine the solution. Experiments show that for a small training data set, the algorithm scores at least as high as two related algorithms, on test set likelihood. Furthermore, it converges faster. Namely, about twice as fast as regular EM and three times as fast as EM with parameter smoothing. The algorithm is relatively easy to im-

<table>
<thead>
<tr>
<th>network</th>
<th>pert. quantized EM</th>
<th>quantized EM</th>
<th>pert. EM</th>
<th>regular EM</th>
</tr>
</thead>
<tbody>
<tr>
<td>alarm</td>
<td>47</td>
<td>11</td>
<td>61</td>
<td>16</td>
</tr>
<tr>
<td>insurance</td>
<td>31</td>
<td>12</td>
<td>42</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 4.6: Average number of iterations required by four EM variants on different networks.
Figure 4.6: Experimental results comparing perturbed quantized EM (solid), quantized EM (dash-dotted), regular EM (dashed), and data perturbed EM (dotted), on learning the parameters of the Alarm network. The horizontal axis shows the train set size. The vertical axis shows average log-likelihood on the train set (a) and on the test set (b).
Figure 4.7: Experimental results comparing perturbed quantized EM (solid), quantized EM (dash-dotted), regular EM (dashed), and data perturbed EM (dotted), on learning the parameters of the Insurance network. The horizontal axis shows the train set size. The vertical axis shows average log-likelihood on the train set (a) and on the test set (b).
Figure 4.8: Experimental results comparing perturbed quantized EM (solid), quantized EM (dash-dotted), regular EM (dashed), and data perturbed EM (dotted). The horizontal axis shows the train set size. The vertical axis shows the state estimation accuracy on one of the variables, for the Alarm network (a) and Insurance network (b).
plement and requires no extra tuning. The algorithm is applicable in situations where we want to speed up learning and the training data set is small.

Next, we performed state estimation by applying the voting algorithm from Chapter 3 on the network that was obtained after the quantized learning phase. The state estimation accuracy was best on tree networks with CPTs that contain clear spikes. On this type of network, the combination of the two methods offers very fast learning and state estimation. This might be useful in situations where one wants fast preliminary state estimation results, while running normal learning and exact inference in the background.

An experiment (Figure 4.5) showed that the variance in the likelihood of the models learned with quantized EM is still quite high (albeit lower than for regular EM). Data perturbation on the quantized phase can decrease this variance, and increase the quality of the learned models. The resulting method (Algorithm 2) was shown to be robust against overfitting, by scoring better on smaller training data sets, compared to existing learning methods. If we are modeling very rare events, it is likely we have only small training data sets. In this case, we can apply perturbed quantized EM to obtain better models, at the expense of more learning iterations.