Scribe Notes: 4/1/2013 and 4/3/2013
Presenter: Daniel Geschwender
Scribe: Tony Schneider
Reading: Chapter 14 of Dechter's Textbook

Probabilistic Networks

In both hard and soft CSPs, the constraints in the problem are known with certainty. However, there are many situations in which uncertainty needs to be included in the model. Probabilistic networks (as well as belief and Bayesian networks) are one way to handle this uncertainty. Although these networks are not 'pure' CSPs, many of the same techniques used in CP (especially those for optimization) can be applied to probabilistic networks.

Background on Probability

- **Single variable probability** [*P(b)*]: The probability of an event occurring
- **Joint probability** [*P(a,b)*] : The probability of two events occurring together
- **Conditional probability** [***P(a|b)***] : The probability of one event occurring knowing that another event has occurred

According to Bayes Theorem, we have the following relations:

$$P(a|b) = \frac{P(a,b)}{P(b)}$$

$$P(a|b)P(b) = P(b|a)P(a) = P(a,b)$$

Chaining Conditional Probability

A joint probability of any size can be expressed as the product of conditional probabilities, enabling transformation from one to the other. In the case of probabilistic networks:

- The conditional probabilities are known a priori, i.e., given as input as tables
- While the joint probabilities are queries (e.g., "What is the probability of events a, b, and c co-occurring?").

$$P(a_1, a_2, ..., a_n) = P(a_1|a_2, ..., a_n)P(a_2|a_3, ..., a_n)P(a_3|a_4, ..., a_n)...P(a_n)$$

Belief Networks

The graphical representation for a belief network is a directed, acyclic graph, where each edge represents the causal influence of one variable on another. Direct influences are modeled with single edges; indirect influences by paths of length two or more. In

Figure 1, the season has a direct influence on both the sprinkler system and rain, and an indirect influence on the whether it's wet or not.

Figure 2: Probabilistic Network Example

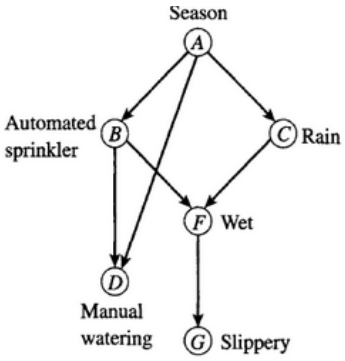

Figure 1: Conditional Probability Table Example

C:

A	P(C=0|A)	P(C=1|A)
w	1.0	0.0
sp	0.7	0.3
su	0.8	0.2
f	0.9	0.1

Each variable in the network has a corresponding *conditional probability table* (CPT), which gives the conditional probability that a variable has some value given the value of its parents (i.e., the variables that directly affect it). Figure 2 shows a conditional probability table for variable C of the corresponding belief network.

In addition to the variables, domains, conditional probability tables, and graph, an *evidence set* may be given, which is tantamount to a subset of variables that are already instantiated (i.e., variables whose domain values are set as input).

The network itself gives a probability distribution over all the variables in the network. Assigning values to variables and multiplying the corresponding probabilities (from the CPTs) will give the probability of that particular series of events occurring.

There are three main queries for probabilistic networks:

1. **Belief assessment:** Given some evidence, how are the probabilities of the other variables in the network affected?
2. **Most probable explanation (MPE):** Given some evidence, what is the most probable assignment to the other variables?
3. **Maximum a posteriori hypothesis (MAP) :** Assign a *subset* of unobserved variables to maximize their conditional probability

Belief Assessment

The idea behind belief assessment is to determine how probabilities of variables are affected given some evidence. The probabilities of variables not in evidence can be updated to reflect the new information, in a process also known as *belief updating*.

As in constraint optimization (chapter 13 of Dechter), a modified version of bucket elimination can be used to perform belief updating. The algorithm (called ELIM-BEL) is

similar to bucket elimination for constraint optimization (ELIM-OPT), but the summation in ELIM-OPT is replaced with the product, and maximization replaced by summation.

The algorithm works by setting up the buckets as you would in other bucket elimination algorithms, *with the variable whose belief you want to update at the top of the ordering*. Then, for each bucket from the bottom up:

- If the bucket contains any evidence (i.e., we know what the variable assignment is), ignore all the probabilities not pertaining to the assignment
- Otherwise generate a new CPT over the union of the scopes of other CPTs in the bucket with the bucket's variable projected out (similar to standard bucket elimination).

To generate the new CPT, every possible tuple is generated for the given scope, *multiplying* the probabilities in the existing CPTs corresponding to the generated tuple. To project out a variable, the probabilities of identical tuples are *added up* to produce the final CPT that will be placed in the lowest bucket whose variable is in the generated CPT's scope:

- *Summation* is used to project out a variable because it is equivalent to asking the question, "What is the probability of this tuple occurring, disregarding this variable?" For example, if the probability of the 2-tuples (0, 0), (0, 1), and (1,0) over the scope {A, B} are .6, .3, and .1, respectively, then projecting out variable B yields two 1-tuples, (0) and (1) with probabilities .9 and .1.
- Similarly, the *product* is used to generate tuples because each value in the generated tuple has a distinct probability of appearing. The probability of all of the values in a tuple appearing in together is their product.

Once only a single CPT remains, the CPT is normalized so that the probabilities of all values of the variable sum to 1:

$$P(x = a|e) = \frac{P(x = a, e)}{\sum_{a \in D_x} P(x = a, e)}$$

The output of ELIM-BEL is an updated CPT for a *single* variable. To get the updated CPTs for a different variable not in evidence, the algorithm needs to be run again with the to-be-updated variable at the top of the ordering. Daniel's slides[1] contain a full example of EILM-BEL (slides 22-29).

[1] http://cse.unl.edu/~choueiry/S13-921/Slides/DG-ProbabilisticNetworks.pptx

ELIM-BEL Derivation

The derivation of the algorithm is again similar to the derivation for constraint optimization. Here, if the desire were to update the belief in a variable *a* given that our evidence is *g = 1* using the network in Figure 1 above, the full computation would be:

$$P(a, g=1) = \sum_{c,b,f,d,g=1} P(g|f)P(f|b,c)P(d|a,b)P(c|a)P(b|a)P(a)$$

…which is equivalent to (pushing the summations inward):

$$P(a)\sum_{c}P(c|a)\sum_{b}P(b|a)\sum_{f}P(f|b,c)\sum_{d}P(d|a,b)\sum_{g=1}P(g|f)$$

Now, moving right to left, we can generate a function over G that only depends on f, and move it as far to the left as possible:

$$P(a)\sum_{c}P(c|a)\sum_{b}P(b|a)\sum_{f}P(f|b,c)\lambda_G(f)\sum_{d}P(d|a,b)$$

This process can be repeated for each rightmost summation until only a function over variable *a* is left:

$$P(a, g=1) = p(a)\lambda_c(a)$$

But we don't want the probability of each value of *a* **and** *g = 1*, we want the probability of *a* **given** that *g = 1*, so a final normalization step is needed:

$$P(a|g=1) = \frac{P(a)\lambda_c(a)}{\sum_{a\in D_x} P(a)\lambda_c(a)}$$

This expression stems from the probability functions given above:

$$P(a|b) = \frac{P(a,b)}{P(b)}$$

Most Probable Explanation (MPE)

The idea behind MPE is to find the most likely assignment to *all* the variables in the network given some evidence (as opposed to belief updating, which updates a *single* variable's probability). Again, this computation can be performed using a modified bucket elimination algorithm called ELIM-MPE.

ELIM-MPE works in an identical fashion to ELIM-BEL, with the exception that now the maximum probability is used when projecting variables out of a CPT. Further, the output of the BE portion of the algorithm will be a probability, but in order to find the

corresponding assignment, the solution must be generated by going through the ordering from first to last and finding the correct value assignments.

An important note is that the functions in buckets with evidence can be considered separately from one another (i.e., rather than taking the product and generating a higher arity CPT, it is possible to generate two separate functions to keep the arity lower). The examples provided in the book (and slides 41-55) use the product as in belief updating.

Maximum a Posteriori Hypothesis (MAP)

MAP queries give the most probable assignment to a *subset* of the non-evidence variables given some evidence (as opposed to the most probable assignment to all non-evidence variables). As Shant pointed out on Piazza, MAP essentially has three sets of variables:

1. the variables in evidence,
2. the variables for which we want to find the assignment with the highest probability, and
3. a third set of variables whose assignments influence the second set.

This situation makes MAP more difficult than MPE, because the variables in the second set must be instantiated before the third (which curtails many optimization opportunities).

ELIM-MAP is an algorithm (one more!) that uses bucket elimination to generate the MAP assignment. The input for the algorithm is the same for ELIM-MPE, but the algorithm additionally requires the subset of variables for which we're interested in finding the most probable assignment. The subset of variables must come first in the provided ordering. The initialization procedure is the same as ELIM-MPE, as is the treatment of variables with evidence.

There are two key differences between ELIM-MAP and ELIM-MPE:

- In ELIM-MAP, if the variable *is not* in the subset of variables in which we're interested, the new CPT is generated by taking the product of all functions and projecting out the bucket's variable by ***summing*** (like in belief updating).
- In ELIM-MAP, if the variable *is* in the subset of variables in which we're interested, the new CPT is generated by taking the product of all functions and projecting out the bucket's variable by ***maximizing*** (as in MPE).

These two rules make ELIM-MAP a more general algorithm than both ELIM-BEL and ELIM-MPE:

- if the subset is all of the variables in the network, the algorithm is equivalent to ELIM-MPE,
- while if the subset is empty, the algorithm is equivalent to ELIM-BEL (with the exception that the last step—generating the assignment—is omitted).

Complexity of Elimination Algorithms

Because all of the ELIM-* algorithms are based on bucket elimination, the complexity is dominated by the time and space required to process each bucket. Specifically, the time and space complexity are exponential in the number of variables in a bucket, meaning that the overall complexity is bounded by the size of the largest bucket (and hence, by the induced width given some ordering).

As previously mentioned, if a bucket contains evidence, then taking the product to generate a new CPT is unnecessary because we already know what the assignment for the bucket will be. Instead, we can take the graph with some fixed ordering, create the induced graph, and remove any nodes that have evidence. This method provides a tighter bound on the induced width of the graph called the *adjusted induced width.*

Question (Robert): Can we just use the evidence first (i.e., instantiate any variables with evidence immediately)?

Answer: You can, and doing so may generate simpler functions (preventing unnecessary calculations), but it may incidentally increase the adjusted induced width of the graph (see Figure 3 below, where the left graph eliminates the variable B after moralizing the graph, while the right graph instantiates B up front and creates an induced graph with a higher width).

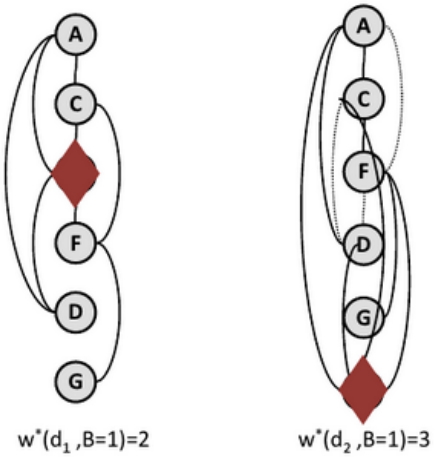

Figure 3: Adjusted Induced Width

Hybrid of Elimination & Conditioning

Because all of the previous elimination algorithms are based on bucket elimination, they require an exponential amount of memory. This requirement is prohibitive for most problems, so some compromise between search and elimination is required in order to trim down the amount of space required by the algorithms.

As a preliminary, full search in a probabilistic network would consist of a traversal of a tree of variable assignments. At each leaf, the joint probability can be calculated for the given path as shown in Figure 4:

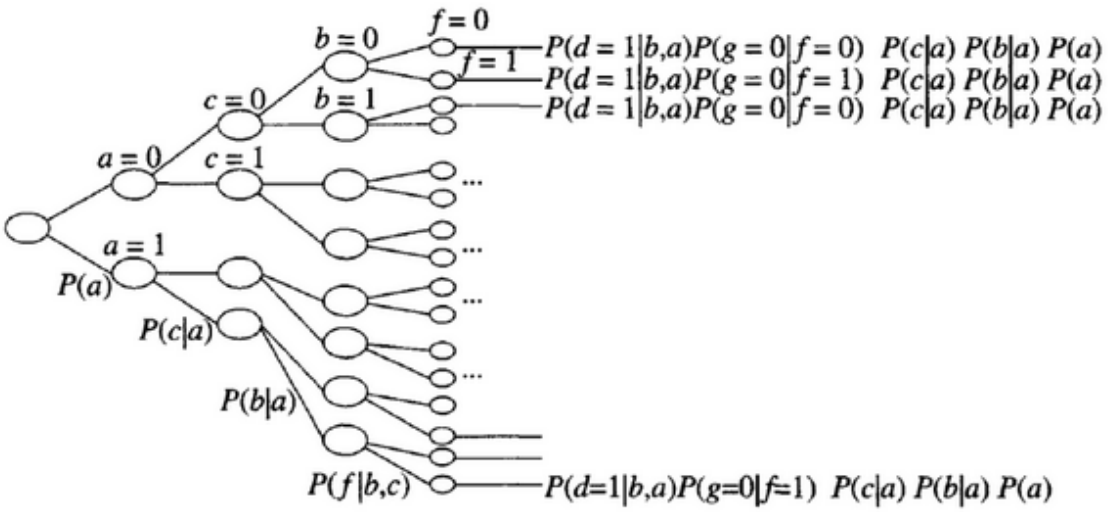

Figure 4: Probabilistic Network Search Tree for P(a, G=0, D=1) given network in Figure 1

In hybrid search, the idea is to search only over some subset of variables *Y,* while other variables will be handled with the corresponding elimination algorithm, using the assignments in *Y* as evidence.

There are two approaches to hybrid search:

1. Use a static selection for *Y* (i.e., *Y* is known ahead of time or given as input)
2. Use a dynamic selection for *Y*

Consider a hybrid for belief updating:

- *The static case* proceeds in a straightforward manner: For each assignment of values to the variables in *Y*, get the output from ELIM-BEL using the union of the provided evidence and the current assignment as input, and sum the outputs of each call to ELIM-BEL together to obtain the updated CPT.
- *The dynamic case* selects the variables in *Y* based on some bound of the degree of a variable (i.e., any variable with a degree higher than the bound belongs in *Y*, while elimination is performed on the other variables). This operation limits the number of CPTs to be considered in each bucket of the elimination algorithm.

The space complexity for hybrid search is

$$O(n \cdot exp(w^*(d, Y \cup E)))$$

where *n* is the number of variables, *w** is the induced width of the graph along some ordering *d*, and *Y U E* is the set of evidence given to the elimination algorithm. Similarly, the time complexity is

$$O(n \cdot exp(w^*(d, Y \cup E) + |Y|))$$

The additional |Y| is due to the time required to perform search over those variables.

A final note: if *E*U*Y* is a cycle-cutset of the moral graph (i.e., the removal of those variables leaves a graph with no cycles), then there will be some ordering of the graph such that its adjusted induced width is 1, leaving a tree and reducing ELIM-COND-BEL (the hybrid algorithm) to the cycle-cutset algorithm.

Note: All figures and equations are taken from Daniel's slides.