Lesson 6 - Bayes Nets

Bayes Rule

P(A|B) = P(B|A) P(A) / P(B)

Bayes rule can be represented via a simple Bayes Network (with nodes A and B and an arc from A to B)

A -> B

P(A) is known as is P(B | A) and P(B | not A). Typically this is useful when A is not observable but B is related to A and is observable.

Diagnostic Reasoning: P(A | B) and P(A | not B)

In some cases we care for both P(A | B) and P(not A | B) in which case can calculate:

P'(A | B) = P(B | A) P(A) P'(not A | B) = P(B | not A) P(not A)

note P' is not an actual probability

P(A | B) = \eta P'(A | B) P(not A | B) = \eta P'(not A | B)

\eta = (P'(A | B) + P'(not A | B)) ^ -1

Conditionally Independent

Conditional Independence implies that

P(B_2 | A, B_1) = P(B_2 | A)

This can be used in Bayes nets like C -> T_1, C -> T_2

P(T_2 = + | T_1 = +) = P(t_2 | t_1, C) P(C | t_1) + P(t_2 | t_1, not C) P(not C | t_1) = P(t_2 | C) P(C | t_1) + P(t_2 | not C) P(not C | t_1)

note that conditional independence doesn't imply independence and independence does not imply conditional independence.

Value of Bayes Networks

Normally to determine P(A, B, C, D, E) you would need to know 2^5 - 1 probabilities (in general for k binary variables you need 2^k - 1). However, with a Bayes net you can generally know the probability with a significantly smaller number of probabilities. For an arbitrary node you need to know 2^m probabilities where m is the number of inputs into the node

D Separation

a:b, b:c

- c dependent on a
- c independent of a given b

a:b, a:c

- b dependent on c
- b independent of c given a

a:c, b:c

- a independent of b
- a dependent on b given c

a:c, b:c, c:...d

- a dependent on b given d

Summary of Bayes Nets

- Graph structure
- Compact Representation
- Conditional Independence

Probabilistic Inference

Probabilistic inference utilizes probability theory, Bayes nets, and independence to generate inferences

A common problem in inference can be thought of similar to a function in programming: given certain inputs return an output. However rather than input and output variables they are called evidence and query variables. Variables that are neither evidence or query are hidden variables

Furthermore, the output is not a single number for each query but rather, it is a probability distribution or the posterior distribution given the evidence: P(Q_1, Q_2, ...| E_1 = e_1, E_2 = e_2, ...)

Another questions is the most likely explanation? argmax_q P(Q_1 = q_1, Q_2 = q_2, ...| E_1 = e_1, E_2 = e_2, ...) or which q values are maximal given evidence variables.

Unlike many programming functions which are single directions, Bayes nets can go in both directions we can provide the evidence and ask which query is most likely or we can give the query and ask which evidence most likely caused the query, or any other direction.

Enumeration

P(+b | +j, +m) = P(+b, +j, +m) / P(+j, +m)

In order to do enumeration we: 1. Re-write the conditional probability with unconditional probabilities. 2. We enumerate all the atomic probabilities based on the hidden variables and calculate the sum of products. 3. Express the sum of products in terms of the probabilities given in the Bayes Net

```
P(+b | +j, +m)
= P(+b, +j, +m) / P(+j, +m)

P(+b, +j, +m)
= \sum_e \sum_a P(+b, +j, +m, e, a)
= \sum_e \sum_a P(+b) P(e) P(a | +b, e) P(+j | a) P (+m | a)
= f(+e, +a) + f(+e, -a) + f(-e, +a) + f(-e, -a)
```

Speeding up Enumeration

Enumeration is pretty efficient and grows exponentially with the number of hidden nodes, arcs and possible values for nodes (if nodes aren't boolean). There are several techniques for speeding up enumeration

Pulling out terms

In some cases the terms of the enumeration can be rearranged. This reduces the number of terms that need to be calculated or looked up in the inner most loop of the summation of products. However this doesn't reduce the overall number of sums needed.

```
\sum_e \sum_a P(+b) P(e) P(a | +b, e) P(+j | a) P (+m | a)
= P(+b) \sum_e P(e) \sum_a P(a | +b, e) P(+j | a) P (+m | a)
```

Maximizing independence

The structure of a Bayes net determines how fast it is to do inference on. Inference on a linear network of n nodes can be done in O(n) time. Inference on a complete network of nodes can be done in O(2^n) time if all nodes are boolean. The more arcs in the network the more calculations.

In general a Bayes net contains less arcs if if it is written in the **causal direction**

Variable Elimination

In an equation like \sum_e \sum_a P(+b) P(e) P(a | +b, e) P(+j | a) P (+m | a) we join up the whole joint distribution before we sum over the hidden variables which is slow because we repeat a lot of work. Variable elimination is an alternative technique.

It's still NP-hard however it works faster than inference by enumeration in most practical cases. It requires an algebra for manipulating factors which are just names for multidimensional arrays.

Given a net r: t, t: l

p(+l) = \sum_r \sum_t P(r) P(t | r) P(+l | t)

Variable elimination requires

1. Joining Factors - A factor is a multidimensional matrix such as P(R), P(T | R), P(L | T). Joining the factors combines 2 factors such as P(R) and P(T | R) to create a table P(T, R)
2. Summing out or marginalizing - Based on the factor for P(T, R) you can sum over the terms for +t and -t in order to create a factor P(T) that can be used in a network of just t: l

Repeating and marginalizing factors can be repeated several times and if we go about choosing variables to eliminate efficiently it can be faster than enumerating over the entire joint probability.

Approximate inference by sampling

Simulate the phenomena (either by simulating the underlying phenomenon or by utilizing joint probability distributions) and determine the probability distribution based on the sampling distribution. If we only do a few counts then there can be more random variation that prevents samples from converging. This has an advantage in that it is computationally easier to come up with an approximate distribution.

With an infinite number of samples the procedure computes the true joint probability distribution. Therefore the sampling method is **consistent**.

In order to handle conditional probability we can use **rejection sampling** where we continue to generate random samples however we eliminate the samples that don't match the conditional probabilities that we are interested in. This procedure is also **consistent**.

In **likelihood weighting** we fix the conditional variables to be what we want however this is **inconsistent**. We can fix this by assigning a probability to each sample and weighing them correctly. A running product of weights is calculated based on the conditional probability whenever we assign a value fixed by a conditional query probability. This produces a **consistent** result.

Gibbs sampling

There are still issues with rejection sampling. Downstream nodes from the conditional probability will get good values based on the conditional evidence that is constrained and weighted. However upstream nodes will not.

A technique called Gibbs Sampling takes all evidence into account. It uses Markov Chain Monte Carlo or MCMC.

In Gibbs we initialize the variables to random values keeping the evidence values fixed. Each iteration through the loop we select on non-evidence variables and re-sample it based on all of the other variables. We repeat this