Bayes Nets

CPS 170
Ron Parr

Conditional Independence

• Suppose we know the following:
  – The flu causes sinus inflammation
  – Allergies cause sinus inflammation
  – Sinus inflammation causes a runny nose
  – Sinus inflammation causes headaches

• How are these connected?

Causal Structure

Knowing sinus separates the variables from each other.

Notation Reminder

• $P(A|B)$ is a conditional prob. distribution
  – It is a function!
  – $P(A=true|B=true)$, $P(A=true|B=false)$,
    $P(A=false|B=True)$, $P(A=false|B=true)$
• $P(A|B)$ is a probability distribution, function
• $P(a|B)$ is a function, not a distribution
• $P(a|b)$ is a number

Conditional Independence

• We say that two variables, A and B, are conditionally independent given C if:
  – $P(A|BC) = P(A|C)$

• How does this help?

• We store only a conditional probability table (CPT) of each variable given its parents

• Naive Bayes (e.g. Spam Assassin) is a special case of this!

Getting More Formal

• What is a Bayes net?
  – A directed acyclic graph (DAG)
  – Parents chosen such that
    $P(x_1, ..., x_n) = \prod P(x_i | \text{parents}(x_i))$

  – Given the parents, each variable is independent of non-descendants
  – For each node $X_i$, store $P(X_i | \text{parents}(X_i))$
  – Represent as table called a CPT
Real Applications of Bayes Nets

- Diagnosis of lymph node disease
- Used by robots to identify meteorites to study
- Study the human genome: Alex Hartemink et al.
- Many other applications...

Space Efficiency

- Entire joint as 32 (31) entries
  - P(H|S), P(N|S) have 4 (2)
  - P(S|AF) has 8 (4)
  - P(A) has 2 (1)
  - Total is 20 (10)
- This can require exponentially less space
- Space problem is solved for “most” problems

Atomic Event Probabilities

\[ P(x_1 \ldots x_n) = \prod_i P(x_i \mid \text{parents}(x_i)) \]

Note that this is guaranteed true if we construct net incrementally, so that for each new variable added, we connect all influencing Variables as parents

Working Smarter I

\[ P(h | SANF) = P(h | SF) \cdot P(S | AF) \cdot P(AF) \]

\[ = P(h | S) \cdot P(N | S) \cdot P(S | AF) \cdot P(AF) \]

\[ = P(h | S) \cdot P(N | S) \cdot P(S | AF) \cdot P(A) \cdot P(F) \]

Working Smarter II

\[ P(h) = \sum_{SANF} P(h | SANF) \]

\[ = \sum_{SANF} P(h | N | SAF) \cdot P(SAF) \]

\[ = \sum_{SANF} P(h | S) \sum_{AF} P(S | AF) \cdot P(A) \cdot P(F) \]

Doing Things the Hard Way

\[ P(f | h) = \frac{P(fh | SANF)}{P(h)} = \frac{\sum_{SANF} P(fh | SANF)}{\sum_{SANF} P(h | SANF)} \]

defn. of conditional probability marginalization

Doing this naively, we need to sum over all atomic events defined over these variables. There are exponentially many of these.
**Checkpoin**

- BNs can give us an exponential reduction in the space required to represent a joint distribution.
- Storage is exponential in largest parent set.
- Claim: Parent sets are often reasonable.
- Claim: Inference cost is often reasonable.
- Question: Can we quantify the relationship between structure and inference cost?

**Computational Efficiency**

\[
\sum_{S_{ANF}} P(h|S_{ANF}) = \sum_{S_{ANF}} P(h|S)P(N|S)P(S|AF)P(A)P(F)
\]

\[
= \sum_S P(h|S)\sum_{N} P(N|S)\sum_{P(S|AF)} P(A)P(F)
\]

The distributive law allows us to decompose the sum.

Potential for an exponential reduction in computation costs.

**What Is a Bayes Net, Really?**

- A Bayes net is a data structure (with associated algorithms) for fast manipulation of probability distributions.
- Bayes nets solve computational problems.
- Real problems are solved by is not a method.
  - Modeling distribution of features is a method.
  - Exploiting independence is a method.
- Bayes nets represent; they do not solve.
- Q: How often can a net solve a computational efficiency problem?

**Now the Bad News…**

- In full generality: Inference is NP-hard.
- Decision problem: Is \( P(X_i) > 0.5 \)?
- We reduce from 3SAT.
- 3SAT variables map to BN variables.
- Clauses become variables with the corresponding SAT variables as parents.

**Reduction**

\[
(\overline{X}_1 \lor X_2 \lor X_3) \land (\overline{X}_2 \lor X_3 \lor X_4) \land ...
\]

Problem: What if we have a large number of clauses? How does this fit into our decision problem framework?

**And Trees**

We could make a single variable which is the AND of all of our clauses, but this would have CPT that is exponential in the number of clauses.
Is BN Inference NP Complete?

- Can show that BN inference is \#P hard
- \#P is counting the number of satisfying assignments
- Idea: Assign variables uniform probability
- Probability of conjunction of clauses tells us how many assignments are satisfying

Checkpoint

- BNs can be very compact
- Worst case: Inference is intractable
- Hope that worst is case:
  - Avoidable
  - Easily characterized in some way

(Undirected) Trees

- Are the structures from our reduction trees?

Dealing With General DAGs

- Q: How does graphical structure relate to our ability to push in summations over variables?
  - A:
    - We relate summations to graph operations
    - Summing out a variable =
      - Removing node(s) from DAG
      - Creating new replacement node
    - Relate graph properties to computational efficiency

Variable Elimination

Recall that in variable elimination for CSPs, we eliminated variables and created new supervariables

BN Variable Elimination

- The same trick applies to Bayes nets
- Observation:
  - Every variable elimination ordering corresponds to a rearrangement of the summation in the marginalization computation
Another Example Network

```
P(c) = 0.5
P(s | c) = 0.1
P(s | T) = 0.5
P(r | c) = 0.8
P(r | T) = 0.2

P(w | sr) = 0.99
P(w | sT) = 0.9
P(w | TR) = 0.9
P(w | TT) = 0.0
```

Marginal Probabilities

```
Suppose we want P(W):
P(W) = \sum_{c,s,r} P(C)P(S | C)P(R | C)P(W | RS)
P(W) = \sum_{s,r} P(W | RS) \sum_{c} P(S | C)P(C)P(R | C)
```

Eliminating Cloudy

```
P(C) = 0.5
P(s | c) = 0.5 * 0.1 * 0.8 + 0.5 * 0.5 * 0.2 = 0.4
P(s | T) = 0.5 * 0.1 * 0.2 + 0.5 * 0.5 * 0.8 = 0.32
P(r | c) = 0.5 * 0.9 * 0.8 + 0.5 * 0.5 * 0.2 = 0.65
P(r | T) = 0.5 * 0.9 * 0.2 + 0.5 * 0.5 * 0.8 = 0.55

P(W) = \sum_{s,r} P(C)P(S | C)P(R | C)P(W | RS)
P(W) = \sum_{s} P(W | RS) \sum_{c} P(S | C)P(C)P(R | C)
```

Eliminating Sprinkler/Rain

```
P(s) = 0.09
P(s | r) = 0.21
P(r | r) = 0.41
P(r | T) = 0.29

P(w | sr) = 0.99
P(w | sT) = 0.9
P(w | TR) = 0.9
P(w | TT) = 0.0

P(W) = \sum_{s,r} P(W | RS)P(RS)
P(W) = 0.99 * 0.99 + 0.21 * 0.9 + 0.41 * 0.9 + 0.29 * 0
P(W) = 0.6471
```

Dealing With Evidence

```
Suppose we have observed that the grass is wet.
What is the probability that it has rained?
P(R | W) = \alpha P(RW)
= \alpha \sum_{c,s} P(CSRW)
= \alpha \sum_{c,s} P(C)P(S | C)P(R | C)P(W | RS)
= \alpha \sum_{c} P(R | C)P(C)\sum_{s} P(S | C)P(R | C)
```

Is there a more clever way to deal with w?

The Variable Elimination Algorithm

```
Eliminate(bn, query)
If bn.vars = query
return bn
Else
x = pick_variable(bn)
newbn.vars = bn.vars - x
newbn.vars = newbn.vars - neighbors(x)
newbn.vars = newbn.vars + newvar
newbn.vars[newvar].function =
\sum_{x \in X.vars(neighbor(x))} P(x).function
return(elim(newbn, query))
```
Efficiency of Variable Elimination

- Exponential in the largest domain size of new variables created
- Equivalently: Exponential in largest function created by pushing in summations
- Linear for trees
- Almost linear for almost trees 😊
- (See examples on board…)

Beyond Variable Elimination

- Variable elimination must be rerun for every new query
- Possible to compile a Bayes net into a new data structure to make repeated queries more efficient
  - Note that inference in trees is linear
    - Define a cluster tree where
      - Cluster = set of original variables
      - Can infer original proba from cluster proba
- For networks w/o good elimination schemes
  - Sampling
  - Variational methods

Facts About Variable Elimination

- Picking variables in optimal order is NP hard
- For some networks, there will be no elimination ordering that results in a poly time solution (Must be the case unless P=NP)
- Polynomial for trees
- Need to get a little fancier if there are a large number of query variables or evidence variables

Bayes Net Summary

- Bayes net = data structure for joint distribution
- Can give exponential reduction in storage
- Variable elimination:
  - simple, elegant method
  - efficient for many networks
- For some networks, must use approximation