Bayes Nets

CPS 170

Ron Parr

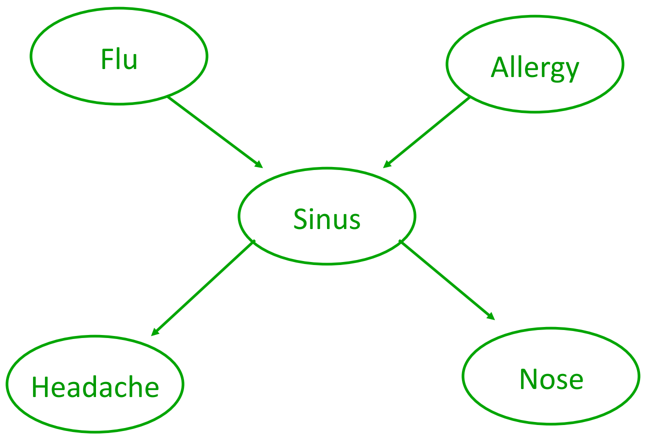
Modeling Distributions

- Suppose we knew $P(X_1 \ldots X_n)$ for all features
  - Can answer any classification question optimally
    - Let $Y=X_i$
    - $P(Y|X_1 \ldots X_n \backslash X_i)$
  - Can answer many clustering type questions
    - $P(X_i X_j)$? (How often do two features co-occur)
    - $P(X_1 \ldots X_n)$ (How typical is an instance?)
- To do correctly we need joint probability distribution
- Unwieldy for discrete variables
- Use *independence* to make this tractable

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.

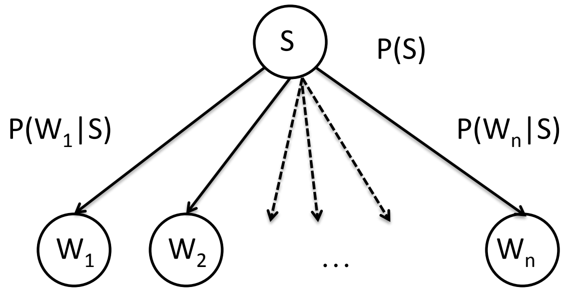
Conditional Independence

- We say that two variables, A and B, are conditionally independent given C if:
  - P(A|BC) = P(A|C)
  - P(AB|C) = P(A|C)P(B|C)
- How does this help?
- We store only a conditional probability table (CPT) of each variable given its parents
- Naïve Bayes (e.g. Spam Assassin) is a special case of this!

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

Naïve Bayes Spam Filter



We will see later why this is a particularly convenient representation. (Does it make a correct assumption?)

Getting More Formal

- What is a Bayes net?
  - A directed acyclic graph (DAG)
  - Given the parents, each variable is independent of non-descendents
  - Joint probability decomposes:

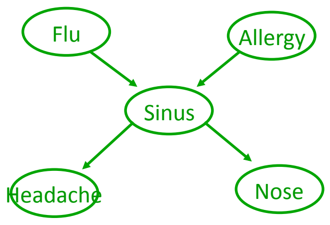
$$P(x_1..x_n) = \prod_i P(x_i \mid \text{parents}(x_i))$$

  - For each node X_i, store $P(X_i|\text{parents}(X_i))$
  - Represent as table called a CPT

Real Applications of Bayes Nets

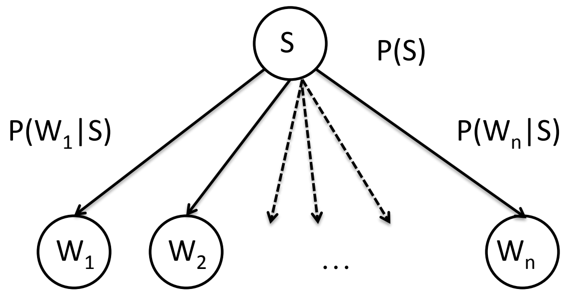
- Diagnosis of lymph node disease
- Used in Microsoft office and Windows
  - http://research.microsoft.com/en-us/groups/mlas/
- Used by robots to identify meteorites to study
- Study the human genome:Alex Hartemink et al.
- Many other applications…

Space Efficiency



- Entire joint distribution 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

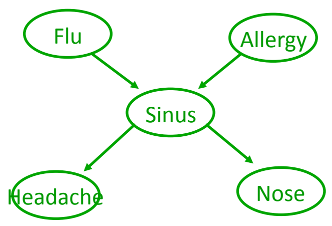
Naïve Bayes Space Efficiency



Entire Joint distribution has 2^{n+1} $(2^{n+1}-1)$ numbers vs. 4n+2 (2n+1)

Atomic Event Probabilities

$$P(x_1 .. x_n) = \prod_i P(x_i \mid \text{parents}(x_i))$$



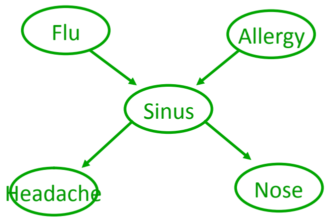
Note that this is guaranteed true if we construct net incrementally, so that for each new variable added, we connect all influencing variables as parents (prove it by induction)

Doing Things the Hard Way

$$P(f \mid h) = \underbrace{\frac{P(fh)}{P(h)}}_{\text{defn. of conditional probability}} = \underbrace{\frac{\sum_{SAN} P(fhSAN)}{\sum_{SANF} P(hSANF)}}_{\text{marginalization}}$$

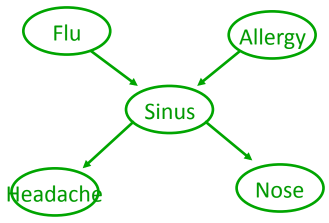
Doing this naïvely, we need to sum over all atomic events defined over these variables. There are exponentially many of these.

Working Smarter I



$$P(hSANF) = \prod_x p(x \mid parents(x))$$
$$= P(h \mid S)P(N \mid S)P(S \mid AF)P(A)P(F)$$

Working Smarter II



$$
\begin{aligned}
P(h) &= \sum_{SANF} P(hSANF) \\
&= \sum_{SANF} P(h \mid S)P(N \mid S)P(S \mid AF)P(A)P(F)) \\
&= \sum_{NS} P(h \mid S)P(N \mid S)\sum_{AF} P(S \mid AF)P(A)P(F)) \\
&= \sum_{S} P(h \mid S)\sum_{N} P(N \mid S)\sum_{AF} P(S \mid AF)P(A)P(F))
\end{aligned}
$$

Potential for exponential reduction in computation.

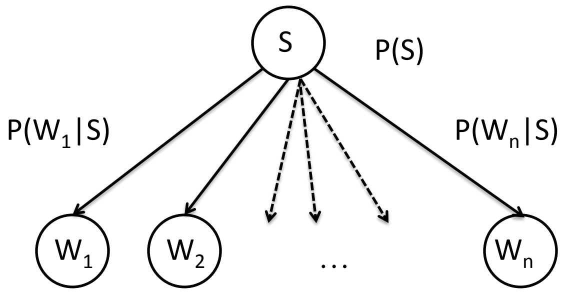
Computational Efficiency

$$
\sum_{SANF} P(hSANF) = \sum_{SANF} P(h \mid S)P(N \mid S)P(S \mid AF)P(A)P(F)
$$
$$
= \sum_{S} P(h \mid S)\sum_{N} P(N \mid S)\sum_{AF} P(S \mid AF)P(A)P(F)
$$

The distributive law allows us to decompose the sum.
AKA: Sum-product algorithm

Potential for an exponential reduction in computation costs.

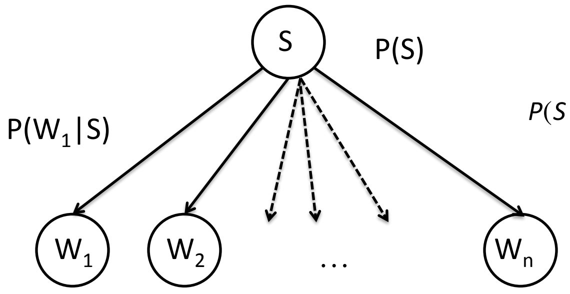
Naïve Bayes Efficiency



Given a set of words, we want to know which is larger: $P(s|W_1 \ldots W_n)$ or $P(\neg s|W_1 \ldots W_n)$.

Use Bayes Rule: $P(S|W_1 \ldots W_n) = \frac{P(W_1 \ldots W_n|S)P(S)}{P(W_1 \ldots W_n)}$

Naïve Bayes Efficiency II



$$P(S|W_1 \ldots W_n) = \frac{P(W_1 \ldots W_n|S)P(S)}{P(W_1 \ldots W_n)}$$

Observation 1: We can ignore $P(W_1 \ldots W_n)$

Observation 2: P(S) is given

Observation 3: $P(W_1 \ldots W_n|S)$ is easy: $P(W_1 \ldots W_n|S) = \prod_{i=1}^{n} P(W_i|S)$

Checkpoint

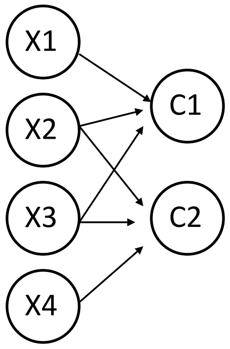
- BNs can give us an exponential reduction in the space required to represent a joint distribution.
- Storage is exponential in largest parent set.
- Claim: Parent sets are often reasonable.
- Claim: Inference cost is often reasonable.
- Question: Can we quantify relationship between structure and inference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

Now the Bad News…

- In full generality: Inference is NP-hard
- Decision problem: Is P(X)>0?
- We reduce from 3SAT
- 3SAT variables map to BN variables
- Clauses become variables with the corresponding SAT variables as parents

Reduction

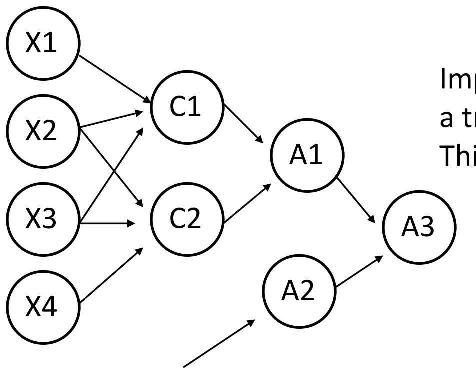
$$(\overline{X}_1 \vee X_2 \vee X_3) \wedge (\overline{X}_2 \vee X_3 \vee X_4) \wedge \ldots$$



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.



Implement as a tree of ANDs. This is polynomial.

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

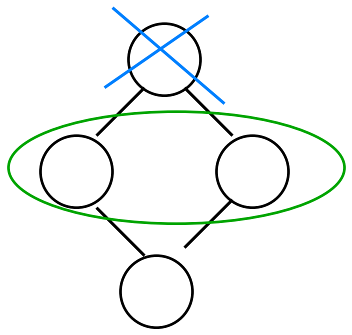
- BNs can be very compact
- Worst case: Inference is intractable

- Hope that worst is case:
  - Avoidable
  - Easily characterized in some way

Clues in the Graphical Structure

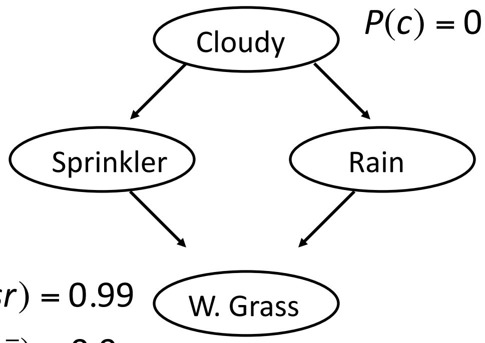
- Q: How does graphical structure relate to our ability to push in summations over variables?
- A:
  - We relate summations to graph operations
  - Summing out a variable =
    - Removing node(s) from DAG
    - Creating new replacement node
  - Relate graph properties to computational efficiency

Variable Elimination



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

Another Example Network



$P(c) = 0.5$

$P(s \mid c) = 0.1$

$P(s \mid \bar{c}) = 0.5$

$P(r \mid c) = 0.8$

$P(r \mid \bar{c}) = 0.2$

$P(w \mid sr) = 0.99$

$P(w \mid s\bar{r}) = 0.9$

$P(w \mid \bar{s}r) = 0.9$

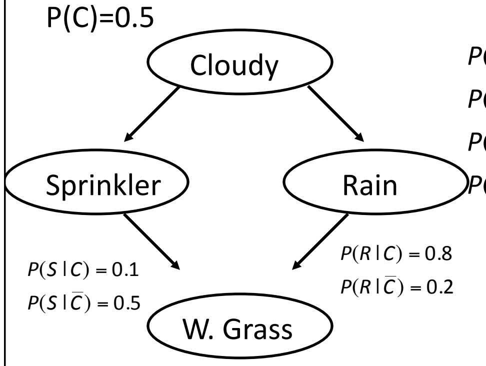
$P(w \mid \bar{s}\bar{r}) = 0.0$

Marginal Probabilities

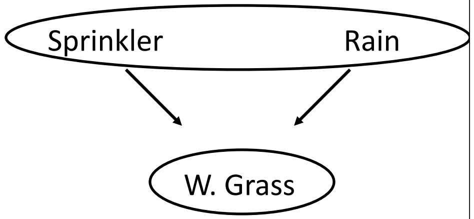
Suppose we want P(W):

$$\begin{aligned} P(W) &= \sum_{CSR} P(CSRW) \\ &= \sum_{CSR} P(C)P(S \mid C)P(R \mid C)P(W \mid RS) \\ &= \sum_{SR} P(W \mid RS) \sum_{C} P(S \mid C)P(C)P(R \mid C) \end{aligned}$$

Eliminating Cloudy



$$P(sr) = 0.5*0.1*0.8 + 0.5*0.5*0.2 = 0.09$$
$$P(s\bar{r}) = 0.5*0.1*0.2 + 0.5*0.5*0.8 = 0.21$$
$$P(\bar{s}r) = 0.5*0.9*0.8 + 0.5*0.5*0.2 = 0.41$$
$$P(\bar{s}\bar{r}) = 0.5*0.9*0.2 + 0.5*0.5*0.8 = 0.29$$



$$\begin{aligned} P(W) &= \sum_{CSR} P(CSRW) \\ &= \sum_{CSR} P(C)P(S \mid C)P(R \mid C)P(W \mid RS) \\ &= \sum_{SR} P(W \mid RS) \sum_{C} P(S \mid C)P(C)P(R \mid C) \end{aligned}$$

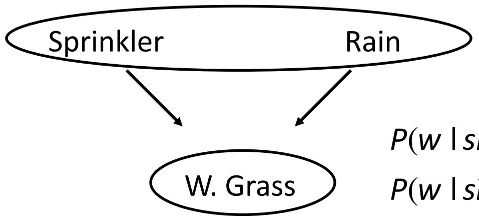
Eliminating Sprinkler/Rain

$P(sr) = 0.09$

$P(s\bar{r}) = 0.21$

$P(\bar{s}r) = 0.41$

$P(\bar{s}\bar{r}) = 0.29$



$P(w \mid sr) = 0.99$

$P(w \mid s\bar{r}) = 0.9$

$P(w \mid \bar{s}r) = 0.9$

$P(w \mid \bar{s}\bar{r}) = 0.0$

$$\begin{aligned} P(w) &= \sum_{SR} P(w \mid RS)P(RS) \\ &= 0.09*0.99 + 0.21*0.9 + 0.41*0.9 + 0.29*0 \\ &= 0.6471 \end{aligned}$$

Dealing With Evidence

Suppose we have observed that the grass is wet?
What is the probability that it has rained?

$$P(R \mid W) = \alpha P(RW)$$
$$= \alpha \sum_{CS} P(CSRW)$$
$$= \alpha \sum_{CS} P(C)P(S \mid C)P(R \mid C)P(W \mid RS)$$
$$= \alpha \sum_{C} P(R \mid C)P(C) \sum_{S} P(S \mid C)P(W \mid RS)$$

Is there a more clever way to deal with w?

Turning our Summation Trick into an Algorithm

- What happens when we "sum out" a variable?
  - All CPTs that reference this variable get pushed to the right of the summation
  - A new function defined over the union of these variables replaces these CPTs
- We call this "variable elimination"
- Analogous to Gaussian elimination in many ways

The Variable Elimination Algorithm

```
Elim(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 =
      $\sum_{X} \prod_{Y \in X \cup neighbors(X)}$ bn.vars(Y).function
  return(elim(newbn, query))
```

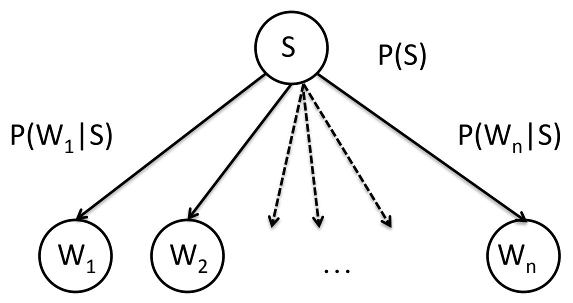
Can also sum out variables that are "hidden" $\longrightarrow$

Efficiency of Variable Elimination

- Exponential in the largest domain size of new variables created (just as in CSPs)
- Equivalently: Exponential in largest function created by pushing in summations (sum-product algorithm)

- Linear for trees
- Almost linear for almost trees ☺

Naïve Bayes Efficiency



Another way to understand why Naïve Bayes is efficient:
It's a tree!

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

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
  - Recall that inference in trees is linear
  - Define a "cluster tree" where
    - Clusters = sets of original variables
    - Can infer original probs from cluster probs
- For networks w/o good elimination schemes
  - Sampling (discussed briefly)
  - Variational methods (not covered in this class)
  - Loopy belief propagation (not covered in this class)

Sampling

- A Bayes net is an example of a **generative model** of a probability distribution
- Generative models allow one to generate samples from a distribution in a natural way
- Sampling algorithm:
  - While some variables are not sampled
    - Pick variable x with no unsampled parents
    - Assign this variable a value from p(x|parents(x))

Comments on Sampling

- Sampling is the easiest algorithm to implement
- Can compute marginal or conditional distributions by counting

- Problem: How do we handle observed values?
  - Rejection sampling: Quit and start over when mismatches occur
  - Importance sampling: Use a reweighting trick to compensate for mismatches

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

- BNs are a major success story for modern AI
  - BNs do the “right” thing (no ugly approximations)
  - Exploit structure in problem to reduce storage/computation
  - Not always efficient, but inefficient cases are well understood
  - Work and used in practice