Bayes Nets

CPS 271
Ron Parr

Modeling Distributions

• Suppose we knew \( P(X_1, X_n) \) for all features
  - Can answer any classification question optimally
  • Let \( Y = X_i \)
  - Can answer many clustering type questions
  • \( P(X|Y_i, Y) \) (How often do two features co-occur)
  • \( P(Y_i, X_n) \) (How typical is an instance?)

• To do correctly we need joint probability distribution

• Unwieldy for discrete variables

• Use independence to make this tractable

Where This Is Going

• Want: Some clever data structures and algorithms to circumvent the combinatorial explosion in the size of the joint distribution

• Note: BNs are NOT a learning method

• First: Understand how to use these data structures

• Relevance to machine learning:
  - Very useful to assume/have such structures
  - Learning of parameters
  - Learning of structure

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:
  - \( P(A|BC) = P(A|C) \)

• 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

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:
    \[ P(x_1, ..., x_n) = \prod_i P(x_i | \text{parents}(x_i)) \]
  - For each node \( X \), store \( P(X | \text{parents}(X)) \)
  - Represent as table called a CPT

Real Applications of Bayes Nets

- Diagnosis of lymph node disease
- Used in Microsoft office and Windows
- 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

\[ P(x_1, ..., x_n) = \prod_i P(x_i | \text{parents}(x_i)) \]

Doing Things the Hard Way

\[ P(f \mid h) = \frac{P(fh) \sum_{\text{SAN}} P(fhSAN)}{P(h) \sum_{\text{SAN}} P(hSANF)} \]

defn. of conditional probability

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

Flu

Allergy

Sinus

Headache

Nose

\[ P(h\text{SANF}) = \prod_{i} p(x_{i} \text{ parents}(x)) \]

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

Working Smarter II

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

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

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

Potential for exponential reduction in computation.

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?

Computational Efficiency

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

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

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

Potential for an exponential reduction in computation costs.

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

\[ (\overline{X}_{1} \lor X_{2} \lor X_{3}) \land (\overline{X}_{2} \lor X_{3} \lor X_{4}) \land \ldots \]

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

Another Example Network

Suppose we want P(W):

\[
P(W) = \sum_{CSR} P(CSRW) \\
= \sum_{CSR} P(C)P(S|C)P(R|C)P(W|RS) \\
= \sum_{SR} P(W|RS) \sum_{C} P(S|C)P(C)P(R|C)
\]
Eliminating Cloudy

\[ P(C) = 0.5 \]

\[ P(C) = 0.5 \times 0.1 + 0.5 \times 0.2 = 0.09 \]

\[ P(C) = 0.5 \times 0.1 + 0.5 \times 0.2 = 0.21 \]

\[ P(C) = 0.5 \times 0.9 \times 0.8 + 0.5 \times 0.2 = 0.41 \]

\[ P(C) = 0.5 \times 0.9 \times 0.2 + 0.5 \times 0.8 = 0.29 \]

\[ P(W) = \sum_{W \in \text{RS}} P(C) \times P(S \mid C) \times P(R \mid C) \times P(W \mid RS) \]

\[ = \sum_{W \in \text{RS}} P(C) \times P(S \mid C) \times P(R \mid C) \times P(W \mid RS) \]

\[ = \sum_{W \in \text{RS}} P(C) \times P(S \mid C) \times P(R \mid C) \times P(W \mid RS) \]


Eliminating Sprinkler/Rain

\[ P(s) = 0.09 \]

\[ P(s) = 0.21 \]

\[ P(r) = 0.41 \]

\[ P(r) = 0.29 \]

\[ P(w | s, r) = 0.99 \]

\[ P(w | s, r) = 0.9 \]

\[ P(w | r) = 0.9 \]

\[ P(w | r) = 0.0 \]

\[ P(w) = \sum_{w} P(w | RS) \times P(RS) \]

\[ = 0.09 \times 0.99 + 0.21 \times 0.9 + 0.41 \times 0.9 + 0.29 \times 0 \]

\[ = 0.6471 \]

Dealing With Evidence

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

\[ P(R \mid W) = \alpha P(RW) \]

\[ = \alpha \sum_{C} P(CSRW) \]

\[ = \alpha \sum_{C} P(C) \times P(S \mid C) \times P(R \mid C) \times P(W \mid RS) \]

\[ = \alpha \sum_{C} P(R \mid C) \times P(C) \sum_{S} P(S \mid C) \times P(W \mid RS) \]

Is there a more clever way to deal with w?

Efficiency of Variable Elimination

- Exponential in the largest domain size of new variables created
- Equivalently: Exponential in largest function created by pushing in summations (sum-product algorithm)
- 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:
    - Clusters = sets of original variables
    - Can infer original probs from cluster probs
- 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
- Q: Why is this interesting for machine learning?
  - A1: Very useful data structure!
  - A2: Often necessary to assume structure (even if it isn’t quite right)
  - A3: Learning/discovering structure can be very useful