INF367A: Probabilistic machine learning

Lecture 4: Bayesian networks - inference

Pekka Parviainen

University of Bergen

28.1.2020

Outline

What is inference?

Computational efficiency

Inference in simple networks

Variable elimination

Inference

- Inference corresponds to using the distribution to answer questions about the environment
- Examples
  - $P(Guilty \,|\, Evidence)$
  - $P(Diagnosis \,|\, Symptoms)$
- Observe some evidence $E = e$, what is the probability of some interesting event $X = x$ (query variable) given the observations?

Computation

- Given a BN for variables $X_1, X_2, \ldots, X_n$, how to compute $P(X_2 \mid X_1)$?
- Recall

$$P(X_2 \mid X_1) = \frac{P(X_1, X_2)}{P(X_1)}$$

- Reduces to computing $P(X_1, X_2)$ and $P(X_1)$
- Marginalization

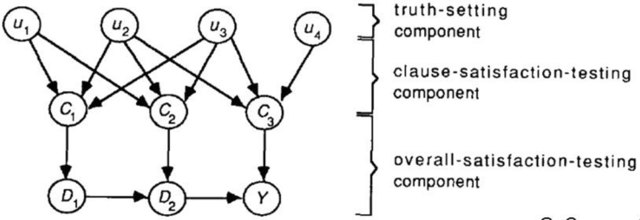
$$\begin{aligned} P(X_1) &= \sum_{X_2} \cdots \sum_{X_n} P(X_1, X_2, \ldots, X_n) \\ &= \sum_{X_2} \cdots \sum_{X_n} \prod_{i=1}^{n} P(x_i \mid pa(X_i)) \end{aligned}$$

- How hard can it be?

Hardness of exact inference

- We can reduce satisfiability to Bayesian network inference
  - Decision problem: $P(Y) > 0$?

$$Y = \underbrace{(u_1 \vee u_2 \vee u_3)}_{C_1} \wedge \underbrace{(\neg u_1 \vee \neg u_2 \vee u_3)}_{C_2} \wedge \underbrace{(u_2 \vee \neg u_3 \vee u_4)}_{C_3}$$



G. Cooper, 1990

Hardness of exact inference

- The decision problem is NP-complete
- Computing $P(Y)$ is at least as hard as counting satisfying assignments
- Thus, Bayesian networks inference is $\#$P-hard in general

Hardness of approximate inference

- Let ρ denote our estimate of $P(X)$
- Absolute error: $|P(X) - \rho| \leq \epsilon$
- Relative error: $\frac{\rho}{1+\epsilon} \leq P(X) \leq \rho(1 + \epsilon)$
- Approximate inference w.r.t. both absolute and relative error is NP-hard

Good news

- The exists efficient algorithms for singly-connected graphs (e.g. trees)
- In general, inference time grows exponentially with respect to the tree-width of the network (not defined at this course) but only linearly w.r.t. the number of variables
  - That is, inference is fixed-parameter tractable w.r.t. tree-width

Inference in a "simple" Bayesian network: Naive Bayes classifier

- A probabilistic classifier that applies Bayes' theorem
- Naive assumption: all features are conditionally independent given the class
- $P(C, x_1, x_2, \ldots, x_d) = P(C) \prod_{i=1}^{d} P(x_i \mid C)$

$$
\begin{tikzcd}
 & & C \arrow[dll] \arrow[dl] \arrow[d] \arrow[drr] & & \\
x_1 & x_2 & x_3 & \cdots & x_d
\end{tikzcd}
$$

Naive Bayes classifier

- Prediction:
  - Calculate through joint probability

$$\begin{aligned} P(C \mid x_1, x_2, \ldots, x_d) &\propto P(C, x_1, x_2, \ldots, x_d) \\ &= P(C) \prod_{i=1}^{d} P(x_i \mid C) \end{aligned}$$

- Predict class label $C^* = \arg\max_C P(C \mid x_1, x_2, \ldots, x_d)$
- Because of wrong independence assumptions, Naive Bayes is often poorly calibrated: Probabilities $P(C \mid x_1, x_2, \ldots, x_d)$ are way off, but $\arg\max_C P(C \mid x_1, x_2, \ldots, x_d)$ is still often correct

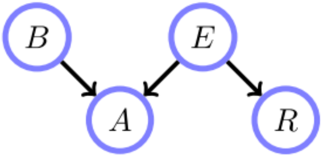
Naive Bayes classifier

- Prediction with partially observed data:
  - Assume that we have observed a subset A of features (and not B, $A \cup B = [d]$)
  - Let $\mathbf{x}_A$ denote feature values of the features in set A

$$
\begin{aligned}
P(C \,|\, \mathbf{x}_A) &\propto P(C, \mathbf{x}_A) \\
&= \sum_{\mathbf{x}_B} P(C) \prod_{i \in A} P(x_i \,|\, C) \prod_{j \in B} P(x_j \,|\, C) \\
&= P(C) \prod_{i \in A} P(x_i \,|\, C) \sum_{\mathbf{x}_B} \prod_{j \in B} P(x_j \,|\, C) \\
&= P(C) \prod_{i \in A} P(x_i \,|\, C) \prod_{j \in B} \sum_{x_j} P(x_j \,|\, C) \\
&= P(C) \prod_{i \in A} P(x_i \,|\, C)
\end{aligned}
$$

Inference in general Bayesian networks

Computation - example (1/3)



- A: 'Alarm is on', B: 'There's a burglar in the house', E:'There's an earthquake', R:'Radio reports of an earthquake'
- Compute $P(B = 1|A = 1)$,the probability that there's a burglar, given the alarm is on.
- Conditional probabilities:

$P(A = 1\|B, E)$	B	E
0.9999	1	1
0.99	1	0
0.99	0	1
0.0001	0	0

$P(R = 1\|E)$	E
1	1
0	0

$$P(E = 1) = 0.000001$$

$$P(B = 1) = 0.01$$

Computation - example (2/3)

$$
\begin{aligned}
P(B=1|A=1) &\stackrel{1}{=} \frac{P(B=1,A=1)}{P(A=1)} \\
&\stackrel{2}{=} \frac{\sum_e \sum_r P(B=1,A=1,E=e,R=r)}{\sum_b \sum_e \sum_r P(B=b,A=1,E=e,R=r)} \\
&\stackrel{3}{=} \frac{\sum_e \sum_r P(A=1|B=1,E=e)P(B=1)P(E=e)P(R=r|E=e)}{\sum_b \sum_e \sum_r P(A=1|B=b,E=e)P(B=b)P(E=e)P(R=r|E=e)}
\end{aligned}
$$

1: definition of conditional probability, 2: marginalization, 3: factorization of the joint distribution according to the BN

Computation - example (3/3)

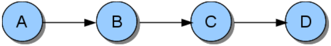
By reordering to simplify computations, we get further that

$$\ldots = \frac{\sum_e P(A=1|B=1,E=e)P(B=1)P(E=e)\sum_r p(R=r|E=e)}{\sum_b\sum_e P(A=1|B=b,E=e)P(B=b)P(E=e)\sum_r P(R=r|E=e)},$$

and, because $\sum_r P(R=r|E=e)=1$, we finally get

$$\ldots = \frac{\sum_e P(A=1|B=1,E=e)P(B=1)P(E=e)}{\sum_b\sum_e P(A=1|B=b,E=e)P(B=b)P(E=e)} \approx 0.99.$$

Variable elimination

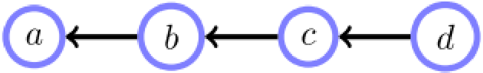


$$
\begin{aligned}
P(D) &= \sum_{A,B,C} P(A,B,C,D) \\
&= \sum_C \sum_B \sum_A P(A)P(B \mid A)P(C \mid B)P(D \mid C) \\
&= \sum_C \sum_B P(C \mid B)P(D \mid C) \sum_A P(A)P(B \mid A) \\
&= \sum_C P(D \mid C) \sum_B P(C \mid B) \sum_A P(A)P(B \mid A)
\end{aligned}
$$

Variable elimination

- Idea: eliminate one variable at a time.
- Usually, each step depends on a limited number of variables.
- Time (and space) complexity of the algorithm depends on the structure of the network and the elimination order.

Variable elimination - a simple example (1/2)



- Compute marginal $P(a=0)$ in the given graph

$$P(a=0) = \sum_{b\in\{0,1\}} \sum_{c\in\{0,1\}} \sum_{d\in\{0,1\}} P(a=0,b,c,d)$$

$$= \sum_{b\in\{0,1\}} \sum_{c\in\{0,1\}} \sum_{d\in\{0,1\}} P(a=0|b)P(b|c)P(c|d)P(d)$$

Naive computation: $2^{n-1}-1=7$ additions (8 terms)

Variable elimination - a simple example (2/2)

- A more efficient approach is to eliminate one variable at a time:

$$P(a=0) = \sum_{b\in\{0,1\}} \sum_{c\in\{0,1\}} P(a=0|b)P(b|c) \underbrace{\sum_{d\in\{0,1\}} P(c|d)p(d)}_{\gamma_d(c)}$$

$$= \sum_{b\in\{0,1\}} P(a=0|b) \underbrace{\sum_{c\in\{0,1\}} P(b|c)\gamma_d(c)}_{\gamma_c(b)}$$

$$= \sum_{b\in\{0,1\}} P(a=0|b)\gamma_c(b)$$

Computational cost: $2 \times (n-1) - 1 = 5$ additions

- *Variable elimination*: eliminate variables starting from the end of the chain (or a leaf of a tree)

Variable elimination with evidence

- $P(X \mid E = e) \propto P(X, E = e)$
- Compute $P(X = x, E = e)$ for all values of X and normalize in the end:

$$P(X = x \mid E = e) = \frac{P(X = x, E = e)}{\sum_x P(X = x, E = e)}$$

Practical hints

- Computation simplifies according to the following principles:
- You can ignore all nodes that are d-separated from X by E
  - $P(X, E = e)$ is off by a constant factor (same for all values of X)
  - Normalizing takes care of it
- You can ignore all nodes that are non-ancestors of X and E
  - *Sink* is a node without children
  - Eliminating a sink leads to a factor with value 1
  - You can continue eliminating sinks until all sinks are either in X or E

Effect of the elimination order

- Elimination order can have an enormous effect on the speed
  - In worst case, inference can be exponentially slower
- Finding an optimal elimination order is NP-hard
- Fortunately, heuristics work often well

Further readings

- Bishop 8.4