Probabilistic graphical models
CPSC 532c (Topics in AI)
Stat 521a (Topics in multivariate analysis)

Lecture 6

Kevin Murphy

Wednesday 29 September, 2004

Administrivia

- Discussion section on Thursday, 3.30-4, in 304 (this week only).

Types of probabilistic inference

- There are several kinds of queries we can make.
- Suppose the joint is $P(Y, E, W) = P(Y, W) \times P(E|Y, W)$.
- Conditional probability queries (sum-product):

$$P(Y|E = e) \propto \sum_w P(Y, W) \times P(e|Y, W)$$

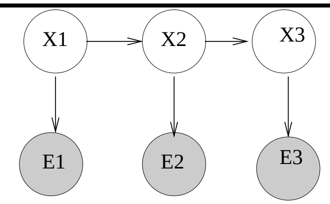
- Most probable explanation (MPE) queries (max-product, MAP):

$$(y, w)^* = \arg\max_y \max_w P(Y, W) \times P(e|Y, W)$$

- Maximum A Posteriori (MAP) queries (max-sum-product, marginal MAP)

$$y^* = \arg\max_y \sum_w P(Y, W) \times P(e|Y, W)$$

Inference in Hidden Markov Models (HMM)



- Conditional probability queries, e.g. estimate current state given past evidence (online filtering)

$$P(X_t|e_{1:t}) = \sum_{x_{1:t-1}} P(x_{1:t-1}, X_t|e_{1:t})$$

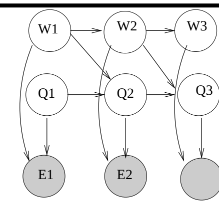
- Most probable explanation (MPE) queries, e.g., most probable sequence of states (Viterbi decoding)

$$x^*_{1:t} = \arg\max_{x_{1:t}} P(x_{1:t}|e_{1:t})$$

Word-error rate vs bit error rate

- Note: Most probable sequence of states not necessarily equal to sequence of most probable states.
- e.g., $X_1 \to X_2$
  $P(X_1) \begin{pmatrix} 0.4 \\ 0.6 \end{pmatrix}$ $P(X_2|X_1) \begin{pmatrix} 0.1 & 0.9 \\ 0.5 & 0.5 \end{pmatrix}$ $P(X_1, X_2) \begin{pmatrix} 0.04 & 0.36 \\ 0.3 & 0.3 \end{pmatrix}$
  $$\arg\max_{x_1} P(X_1) = 1, \quad \arg\max_{x_1} \max_{x_2} P(X_1, X_2) = (0, 1)$$
- Viterbi decoding minimizes word error rate
  $$x^*_{1:t} = \arg\max_{x_{1:t}} P(x_{1:t}|e_{1:t})$$
- To minimize bit error rate, use most marginally likely state
  $$\begin{aligned} P(X_t|y_{1:t}) &= \sum_{x_{1:t-1}} P(x_{1:t-1}, X_t|e_{1:t}) \\ x^*_t &= \max_x P(X_t = x|e_{1:t}) \end{aligned}$$

MAP vs Marginal MAP



- Consider a Dynamic Bayes Net (DBN) for speech recognition, where W = word and Q = phoneme.
- Most likely sequence of *states* (Viterbi/ MAP, max-product):
  $$\arg\max_{q_{1:t}, w_{1:t}} P(q_{1:t}, w_{1:t}|e_{1:t})$$
- Most likely sequence of *words* (Marginal MAP, max-sum-product):
  $$\arg\max_{w_{1:t}} \sum_{q_{1:t}} P(w_{1:t}, q_{1:t}|e_{1:t})$$
- Max-product often used as computationally simpler approximation to max-sum-product (or can use A^* decoding).

Complexity of exact inference

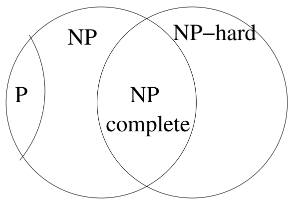
- Determinining if $P_B(X = x) > 0$ for some (discrete) variable X and some Bayes net B is NP-complete.
- What does this mean?
- Roughly: The best algorithm for exact inference (in discrete-state models) probably takes exponential time, in the worst case.
- More formally: we need a review of basic computational complexity theory.

Decision problems

- Defn: a **decision problem** is a task of the form: does there exist a solution which satisfies these conditions?
- Example: boolean satisfiability:
  $$(q_1 \vee \neg q_2 \vee q_3) \wedge \left(\neg q_q \vee q_2 \vee \neg q_3\right)$$
  is satisfiable ($q_1 = q_2 = q_3$=true)
- **3-SAT** is boolean satisfiability where $\phi = C_1 \wedge C_2 \ldots \wedge C_n$, and every clause C_i has 3 literals.

P vs NP

- Defn: A decision problem Π is in P if it can be solved in polynomial time.
- Defn: Π is in NP if it can be solved in polynomial time using a non-deterministic oracle (i.e., you can verify its guesses in polytime).
- Defn: Π is NP-hard if $\forall \Pi' \in NP.\ \exists T \in P.\ \Pi' \xrightarrow{T} \Pi$.
- Defn: Π is NP-complete if it is NP-hard and in NP.
- Conjecture: $P \neq NP$

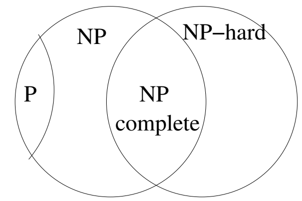


Proving NP-completeness

- Thm: 3-SAT is NP-complete.
- To show Π is NP-hard, it suffices to find a transformation $T \in P$ from another NP-hard problem Π' (e.g., 3-SAT) since

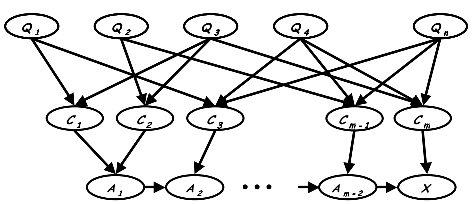
$$NP \xrightarrow{T'} \Pi' \xrightarrow{T} \Pi$$

- To show Π is NP-complete, show it is NP-hard and that you can check (oracular) guesses in poly-time.



Exact inference in discrete Bayes nets is NP-complete

- Thm: the decision problem "Is $P_B(X_i = x) > 0$?" is NP-complete.
- Proof. To show in NP: Given an assignment $X_{1:n}$, we can check if $X_i = x$ and then check if $P(X_{1:n}) > 0$ in poly-time. To show NP-hard: we can encode any 3SAT problem as a polynomially sized Bayes net, as shown below.
- $P(X = 1|q_{1:n}) > 0$ iff $q_{1:n}$ is a satisfying assignment.



Complexity of approximate inference

- Defn: An estimate ρ has absolute error ϵ for $P(y|e)$ if $|P(y|e) - \rho| \leq \epsilon$.
- Defn: An estimate ρ has relative error ϵ for $P(y|e)$ if

$$\frac{\rho}{1+\epsilon} \leq P(y|e) \leq \rho(1+\epsilon)$$

- Thm: Computing $P(X_i = x)$ with relative error ρ is NP-hard.
- Thm: Computing $P(X_i|e)$ with absolute error for any $\epsilon \in (0, 0.5)$ is NP-hard.
- But: special cases may have error bounds.
- And: heuristics often work well.

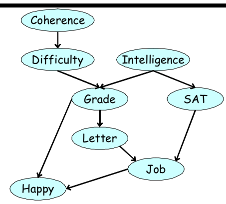
How hard is inference in practice?

- We will show later that exact inference can always be done in time $O(Nv^W)$, where W_G is the *tree-width* of the graph (to be defined later) and $v = \max_i |X_i|$ is the max number of values (states) each node can take.
- The NP-hardness proof shows that, *in the worst case*, we have $W_G \sim N$.
- But for many models used in practice, we have $W_G \sim$ constant.
- Also, for *Gaussian* graphical models, exact inference is $O(N^3)$ no matter what the graph structure is!

Exact inference in Gaussian models takes $O(N^3)$ time

- For *Gaussian* graphical models, exact inference is $O(N^3)$ no matter what the graph structure is!
- c.f., linear programming easier than integer programming.
- Lecture 3: Any undirected graphical model in which potentials have the form
$$\psi_{ij} = \exp(X_i - \mu_i)\Sigma_{ij}^{-1}(X_j - \mu_j)$$
can be converted to a joint Gaussian distribution.
- Book chap 4: any directed graphical model in which CPDs have the form
$$p(X_i|X_{\pi_i}) = \mathcal{N}(X_i; W X_{\pi_i} + \mu_i, \Sigma_i)$$
can be converted to a joint Gaussian distribution.
- Exact inference in a Gaussian graphical model = matrix inversion.

Variable elimination algorithm



- Key idea 1: push sum inside products.
- Key idea 2: use (non-serial) dynamic programming to cache shared subexpressions.

$$\begin{aligned}
P(J) &= \sum_L \sum_S \sum_G \sum_H \sum_I \sum_D \sum_C P(C, D, I, G, S, L, J, H) \\
&= \sum_L \sum_S \sum_G \sum_H \sum_I \sum_D \sum_C P(C)P(D|C)P(I)P(G|I,D)P(S|I)P(L|G)P(J|L,S)P(H|G,J) \\
&= \sum_L \sum_S \sum_G \sum_H \sum_I \sum_D \sum_C \phi_C(C)\phi_D(D,C)\phi_I(I)\phi_G(G,I,D)\phi_S(S,I)\phi_L(L,G)\phi_J(J,L,S)\phi_H(H,G,J) \\
&= \sum_L \sum_S \phi_J(J,L,S) \sum_G \phi_L(L,G) \sum_H \phi_H(H,G,J) \sum_I \phi_S(S,I)\phi_I(I) \sum_D \phi_(G,I,D) \sum_C \phi_C(C)\phi_D(D,C)
\end{aligned}$$

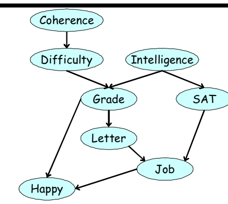
Working right to left (peeling)

$$\begin{aligned}
P(J) &= \sum_L \sum_S \phi_J(J,L,S) \sum_G \phi_L(L,G) \sum_H \phi_H(H,G,J) \sum_I \phi_S(S,I)\phi_I(I) \sum_D \phi_(G,I,D) \underbrace{\sum_C \phi_C(C)\phi_D(D,C)}_{\tau_1(D)} \\
&= \sum_L \sum_S \phi_J(J,L,S) \sum_G \phi_L(L,G) \sum_H \phi_H(H,G,J) \sum_I \phi_S(S,I)\phi_I(I) \underbrace{\sum_D \phi_(G,I,D)\tau_1(D)}_{\tau_2(G,I)} \\
&= \sum_L \sum_S \phi_J(J,L,S) \sum_G \phi_L(L,G) \sum_H \phi_H(H,G,J) \underbrace{\sum_I \phi_S(S,I)\phi_I(I)\tau_2(G,I)}_{\tau_3(G,S)} \\
&= \sum_L \sum_S \phi_J(J,L,S) \sum_G \phi_L(L,G) \underbrace{\sum_H \phi_H(H,G,J)}_{\tau_4(G,J)} \tau_3(G,S) \\
&= \sum_L \sum_S \phi_J(J,L,S) \underbrace{\sum_G \phi_L(L,G)\tau_4(G,J)\tau_3(G,S)}_{\tau_5(J,L,S)} \\
&= \sum_L \underbrace{\sum_S \phi_J(J,L,S)\tau_5(J,L,S)}_{\tau_6(J,L)} \\
&= \underbrace{\sum_L \tau_6(J,L)}_{\tau_7(J)}
\end{aligned}$$

Different ordering

$$
\begin{aligned}
P(J) &= \sum_D \sum_C \phi_D(D,C) \sum_H \sum_L \sum_S \phi_J(J,L,S) \sum_I \phi_I(I)\phi_S(S,I) \underbrace{\sum_G \phi_G(G,I,D)\phi_L(L,)\phi_H(H,G,J)}_{\tau_1(I,D,L,J,H)} \\
&= \sum_D \sum_C \phi_D(D,C) \sum_H \sum_L \sum_S \phi_J(J,L,S) \underbrace{\sum_I \phi_I(I)\phi_S(S,I)\tau_1(I,D,L,J,H)}_{\tau_2(D,L,S,J,H)} \\
&= \sum_D \sum_C \phi_D(D,C) \sum_H \sum_L \underbrace{\sum_S \phi_J(J,L,S)\tau_2(D,L,S,J,H)}_{\tau_3(D,L,J,H)} \\
&= \sum_D \sum_C \phi_D(D,C) \sum_H \underbrace{\sum_L \tau_3(D,L,J,H)}_{\tau_4(D,J,H)} \\
&= \sum_D \sum_C \phi_D(D,C) \underbrace{\sum_H \tau_4(D,J,H)}_{\tau_5(D,J)} \\
&= \sum_D \underbrace{\sum_C \phi_D(D,C)\tau_5(D,J)}_{\tau_6(D,J)} \\
&= \underbrace{\sum_D \tau_6(D,J)}_{\tau_7(J)}
\end{aligned}
$$

Dealing with evidence: method 1



- We can instantiate observed variables to their observed value:

$$
\begin{aligned}
P(J|I=1,H=0) &= \frac{P(J,I=1,H=0)}{P(I=1,H=0)} \propto P(J,I=1,H=0) \\
&= \sum_{C,D,G,L,S} P(C,D,I=1,G,S,L,J,H=0)
\end{aligned}
$$

- The denominator is $P(e) = P(I=1,H=0)$.
- For Markov networks, the denominator is $P(e) \times Z$.

Dealing with evidence: method 2

- We can associate a local evidence potential with every node, and set $\phi_i(X_i) = \delta(X_i, x_i^*)$ if X_i is observed to have value x_i^*, and $\phi_i(X_i) = 1$ otherwise:

$$
P(X_{1:n}|ev) \propto P(X_{1:n}) \prod_i P(ev_i|X_i)
$$

- e.g.,

$$
\begin{aligned}
&P(J|I=1,H=0) \propto \\
&\quad \sum_{C,D,I,G,S,L,J,H} P(C,D,I,G,S,L,J,H)\delta_I(I,1)\delta_H(H,0)
\end{aligned}
$$