L12 – Sampling, Max-Product and Discrete HMMs

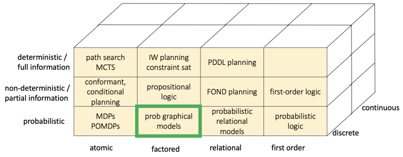
Barber 5.2.1, 23.2, AIMA 14.1–14.3

What you should know after this lecture

- How to use the max-product algorithm to find the maximum likelihood complete assignment in a factor graph
- How to use sampling in Bayes nets and factor graphs to compute approximate answers to conditional probability queries
- What a hidden Markov model is and what it is good for
- What a recursive “filter” is, in this context
- How to solve inference problems in an HMM using sum-product and max-product

Probabilistic reasoning about partially-specified world states

Factored states
Discrete-valued factors
Approximate inference
Temporal models



Finding most probable assignment in a factor graph

We can an algorithm very similar to sum product, called max product. Just as $ab + ac = a \cdot (b + c)$,
$\max(ab, ac) = a \cdot \max(b, c)$ for non-negative a.
Do forward pass with messages as for sum-product, but

$$\mu_{\phi \to V}(v) = \max_{\bar{w} \in N(\phi) \setminus V} \phi(v, \bar{w}) \prod_{W \in N(\phi) \setminus V} \mu_{W \to \phi}(w)$$

Keep track of the values of *W* that yielded the max for each *v*:

$$M_V(v) = \operatorname*{argmax}_{\bar{w} \in N(\phi) \setminus V} \phi(v, \bar{w}) \prod_{W \in N(\phi) \setminus V} \mu_{W \to \phi}(w)$$

Decoding to find most probable assignment

Work backward from root V:

$$v^* = \operatorname*{argmax}_v \mathsf{P}(v)$$

Best value for each child W_i of V:

$$w_1^*, \ldots, w_k^* = M_V(v)$$

Approximate inference methods

Especially when the model is large or loopy, exact inference can be too computationally expensive. What to do?

- Amortized inference: if you will have to solve many similar inference problems in the future, then
  - Solve a bunch of problems of the form $P(Q \mid E = e_i)$
  - Train a supervised neural network to map from e_i to $P(Q \mid E = e_i)$
  - Just use the neural net at query time
- Sampling-based methods
  - We want to compute $P(V \mid E = e)$
  - Draw samples $v_1, \ldots, v_M \sim P(V \mid E = e)$
  - Compute

$$\hat{P}_M = \frac{1}{M} \sum_{i=1}^{M} v_i$$

  - Converges to $P(V \mid E = e)$ as $M \to \infty$

Sampling in Bayes nets

Easy to do ancestral sampling to get samples of any unconditional marginal or joint $\bar{v} \sim P_\alpha(\bar{V})$ when α is a BN

1. Sort nodes in α into topological order so that all nodes $pa(V)$ come before V in the ordering.
2. For $i = 1$ to M `// number of samples
   - For $j = 1$ to N `// number of nodes in network
     $x^i_j = \text{sample}(P_\alpha(V_j \mid pa(V_j) = x^i_j[pa(V_j)])))$
3. Use $\{x^i\}_{i=1..M}$ to estimate whatever you want, e.g.

$$\hat{P}_\alpha(V_k = 1, V_j = 0) = \frac{1}{M}\mathbb{I}(x^i_k = 0 \wedge x^i_j = 0)$$

where $\mathbb{I}(a) = 1$ if a else 0

Conditional sampling

What if we want $P_\alpha(V \mid E = e)$? Two general approaches:

- Rejection sampling: do ancestral sampling, but throw away all examples in which $E \neq e$.
  - Can be very slow if $P(E = e)$ is small.
- Importance sampling: sample from an easier distribution Q, but reweight the samples to compute your result
  - Let Q be a distribution over same domain as desired distribution P and $\{x^i\}_{1,\ldots,M} \sim Q(x)$
  - Define $w_i = \frac{P(x^i)}{Q(x^i)}$ and $Z = \sum w_i$
  - Then, e.g.,

$$\hat{P}(X = 1) = \frac{1}{Z} \sum_{i=1}^{M} w_i \mathbb{I}(x^i = 1)$$

  - Necessary that $Q(x) > 0$ for any x where $P(x) > 0$.
  - Have to be able to evaluate $P(x)$ and $Q(x)$
  - If P and Q are very different, you will need large M to get a good estimate.

Bayes net importance sampling

In Bayes nets, let $Z = V \setminus E$ (the unobserved nodes)

- Fix all $E = e$ and then use ancestral sampling to get samples from

$$Q(Z) = \prod_i P(Z_i \mid \mathrm{pa}(Z_i))$$

- Importance weights

$$\frac{P(z \mid e)}{Q(z)} \propto \frac{P(z, e)}{Q(z)} = \frac{\prod_i P(z_i \mid \mathrm{pa}(Z_i)) \prod_j P(e_j \mid \mathrm{pa}(E_j))}{\prod_i P(z_i \mid \mathrm{pa}(Z_i)}$$
$$= \prod_j P(e_j \mid \mathrm{pa}(E_j))$$

The name importance sampling also used in a context of sampling from continuous distributions for a different method.

Gibbs sampling

Can be applied in Bayes nets but easiest to think of in factor graphs. Simple type of Markov chain Monte Carlo (MCMC).

- Define a Markov chain where
    - States are assignments of values to all variables
    - The stationary distribution of the chain is the desired distribution $P(\bar{V})$
- To do estimation:
    - Run the chain for a while and throw those samples away ("burn in" phase)
    - Keep (every kth) sample and use them to estimate the quantity of interest

Gibbs sampling in graphical model

1. Initialize values $v_1, \ldots, v_N$ at random
2. Loop
   - Choose i randomly from $1, \ldots, N$
   - Set v_i to a sample from

     $$P(V_i \mid V \setminus V_i = v \setminus v_i) = P(V_i \mid \text{mb}(V_i)) = \text{mb}(v_i))$$

     where $\text{mb}(V_i)$ is the Markov blanket
3. Use the v_i samples to estimate quantity of interest.

Markov blanket

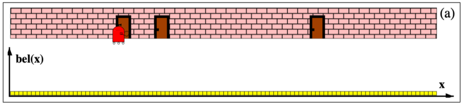
- In a factor graph, it's the neighboring nodes

$$P(V_i \mid \text{mb}(V_i) = \text{mb}(v_i)) = \prod_{\phi \in N(V_i)} \phi[\text{mb}(v_i)]$$

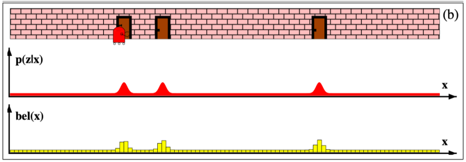
where $\phi[\text{mb}(v_i)]$ is the vector of values for variable V_i that remains after selecting the other dimensions of factor ϕ to have their associated values in $\text{mb}(v_i)$.

Robot Localization

- Robot initially has no idea where it is.



- Robot has a door detector. Intuitively (to us!) this gives the robot three possible locations it might be at.



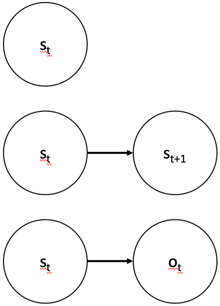
(Figure due to Thrun, Burgard and Fox, 2003, Probabilistic Robotics.)

- Concerns:
    - How do we represent those three locations? Is it really only three locations the robot can be at?
    - What happens when the robot moves?

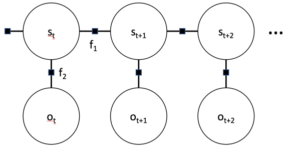
Robot Localization

- Let's say the state at each time t is s_t, where s_t is a random variable over the domain of locations (e.g., discrete locations in the hallway).
- Let's say the observations at each time step are o_t over the domain of $\{$Door, No-Door$\}$
- We need some way of linking the states at time s_t and s_{t+1} and the states to the observations. Our door detector might sometimes fail, and it'll fail in proportion to how close (or far) we are from the door. Also, when we try and move from state s_t to s_{t+1}, sometimes we'll stop short, sometimes we'll overshoot.
- Let's represent s_t and o_t as random variables, and assume that we know $P(s_{t+1}|s_t)$, $P(o_t|s_t)$ and a prior $P(s_0)$.

Resulting Graphical Model



Each of these subgraphs are the result of our assumed model of the world



When we merge common variables, we get this resulting factor graph.

Hidden Markov Models

- Idea first due to Baum (1966), used throughout communications field, summarized by Rabiner (1989)
- Defined by a model λ consisting of a tuple $\lambda = (\mathcal{S}, \mathcal{O}, A, B, \pi)$ that describe a discrete dynamical system as follows:
  - $\mathcal{S}$ is the set of hidden discrete states $\mathcal{S} = \{s(1), s(2), \ldots, s(n)\}$
  - $\mathcal{O}$ is the set of discrete observations $O = \{o(1), o(2), \ldots, o(m)\}$
  - $\mathbf{A} = \{A_{ij}\}$ is the dynamics or "transition" model, $A_{ij} = P(S_t = s(j) \mid S_{t-1} = s(i))$
  - $\mathbf{B} = \{B_{ik}\}$ is the measurement or "sensor" model, $B_{ik} = P(O_t = o(k) \mid S_t = s(i))$
  - π is the initial state distribution

Barber uses h for states, v for observations. AIMA uses $\mathbf{X}$ for states, $\mathbf{e}$ for observations, and never uses a specific symbol for the transition or observation models. We are using s for states and o for observations to be consistent with the MDP and POMDP lectures coming up.

Dependencies in Hidden Markov Models

- Note that **A**, **B** and π are potentials of the form $\phi(\cdot)$ that we saw in last lecture.
- HMM defined by:
  - $S_{0:t-1} \perp\!\!\!\perp S_{t+1:T} \mid S_t$
  - $O_t \perp\!\!\!\perp (S_{0:t-1}, S_{t+1:T}) \mid S_t$
  - We have conditional distributions $P(O_t|S_t)$ and $P(S_{t+1}|S_t)$
  - First assumption known as the "Markov" assumption.

Three Questions

Three questions we might ask:

- Given the observation sequence $\mathbf{o}_{0:T} = \{o_0, o_1, \ldots, o_T\}$, how do we efficiently compute $P(S_T|\mathbf{o}_{0:T})$, the probability of the observation sequence given the model?
- Given the observation sequence $\mathbf{o}_{0:T} = \{o_0, o_1, \ldots, o_T\}$, how do we efficiently compute $P(o_{0:T})$, the probability of the observation sequence given the model?
- Given the observation sequence $\mathbf{o}_{0:T} = \{o_0, o_1, \ldots, o_T\}$, how do we find a corresponding state sequence $\mathbf{s}*_{0:T} = \{s^*_0, s^*_1, \ldots, s^*_T\}$ which is optimal in some meaningful sense (i.e., best “explains” the observations)?

Rabiner did not include our first question, but did include an important additional question about how to estimate A and B from data.

Filtering

Compute $P(s_T \mid o_{0:T})$

- Use sum-product, with S_T as the root of the tree.
- Conditioning on evidence $o_0, \ldots, o_T$ effectively selects out a specific column of B for each time step.

$$P(S_0 \mid o_0) \propto \pi \cdot B_{o_0}$$
$$P(S_t \mid o_{0:t}) \propto \sum_{S_{t-1}} P(S_{t-1} \mid o_{0:t-1}) \cdot A \cdot B_{o_t}$$

- Nice recursive form! Often called Bayes filter.
- To connect with usual treatment, define $\alpha_t = P(S_t \mid o_{0:t})$
- Product of messages coming from previous state and from observation.

Smoothing

More generally, compute, for all t, $P(s_t \mid o_{0:T})$
Forward-Backward algorithm = sum-product

- Do forward pass, computing α from left
- Do backward pass, computing messages from the right

$$\begin{aligned}
\beta_t &\propto P(S_t \mid o_{t+1:T}) \\
&= \sum_{S_{t+1}} A \cdot B_{o_{t+1}} \cdot P(S_t \mid o_{t+2:T}) \\
&= \sum_{S_{t+1}} A \cdot B_{o_{t+1}} \cdot \beta_{t+1} \\
\beta_T &= \mathbf{1}
\end{aligned}$$

- $P(S_t \mid o_{0:T}) = \alpha_t \beta_t$

Likelihood of observation sequence

The values α_T are equivalent to $P(o_{0:T}, S_T)$, so

$$P(o_{0:T}) = \sum_s P(o_{0:T}, S_T = s) = \sum_s \alpha_T[s]$$

Derivation:

$$\begin{aligned}
P(o_{0:T}) &= \sum_{s_{0:T}} P(o_{0:T} \mid s_{0:T}) P(s_{0:T}) \\
&= \sum_{s_{0:T}} \left(\prod_{t=0}^{T} P(o_t \mid s_t) \right) P(s_{0:T}) \\
&= \sum_{s_{0:T}} \left(\left(\prod_{t=2}^{T} P(o_t \mid s_t) \right) P(s_{2:T} \mid s_1) \Big(P(s_1 \mid s_0) P(o_1 \mid s_1) \Big) P(o_0 \mid s_0) P(s_0) \right) \\
&= \sum_{s_{0:T}} B_{s_T,o_T} \cdot A_{s_{T-1},s_T} \dots B_{s_1,o_1} \cdot A_{s_0,s_1} \cdot B_{s_0,o_0} \cdot \pi(s_0) \\
&= B_{s_T,o_T} \cdot \sum_{s_T} A_{s_{T-1},s_T} \dots B_{s_1,o_1} \cdot \underbrace{\underset{\substack{\downarrow \\ \text{Marginalizing out } s_0}}{\sum_{s_0}} A_{s_0,s_1} \cdot \underbrace{B_{s_0,o_0} \cdot \pi(s_0)}_{\alpha_0}}_{\alpha_1}
\end{aligned}$$

$$\Rightarrow \alpha_{t+1} = \left[\sum_{j=0}^{|S|} \alpha_t A_{i,j} \right] \cdot B_{i,o_{t+1}} \quad \text{And} \quad P(o_{0:T}) = \sum \alpha_T$$

Maximum likelihood state sequence

Compute $s^*_{0:T} = \text{argmax}_{s_{0:T}} P(s_{0:T} \mid o_{0:T})$ using max-product algorithm!

- Forward pass to compute

$$\delta_t = \max_{s_{0:t}} P(s_{0:t} \mid o_{0:t}) = \max_{S_{t-1}} A \cdot \delta_{t-1} \cdot B_{o_t}$$

- Remember best s_{t-1} for each s

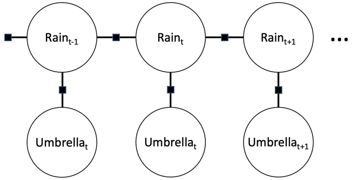
$$\psi_t(s) = \underset{S_{t-1}}{\text{argmax}}\, A \cdot \delta_{t-1} \cdot B_{o_t}$$

- Backtrace:

$$s^*_T = \underset{S_T}{\text{argmax}}\, \delta_T$$
$$s^*_t = \psi_{t+1}(s^*_{t+1})$$

Also called the Viterbi algorithm

Viterbi Decoding Example (from AIMA)

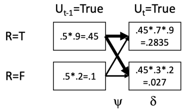


R_{t-1}	$P(R_{t-1})$
T	.5
F	.5

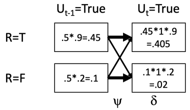
R_{t-1}	$P(R_t{=}T)$
T	.7
F	.3

R_t	$P(U_t{=}T)$
T	.9
F	.2

One step of Viterbi decoding would be:



If we set the $P(R_t = T|R_{t-1} = T) = 1$ and $P(R_t = F|R_{t-1} = F) = 1$, we get a slightly different graph:



- On the right, it might be the case that the $R_t = F$ really is the most likely state. How might this happen?

Next time

- Graphical models with continuous distributions
- Kalman filtering