Problem Session Week 8

Trevor Maxfield
maxfit@stanford.edu

Minae Kwon
minae@cs.stanford.edu

May 26th, 2023

CS 221 - Spring 2023, Stanford University

Reviewing Lecture Material

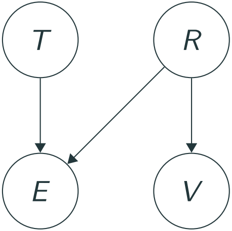
Reviewing Lecture Material

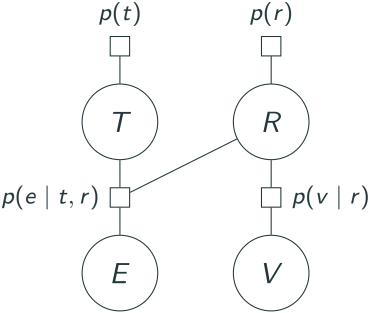
Bayesian Networks and HMMs

Definition
Let $X = (X_1, \ldots, X_n)$ be random variables. A Bayesian network is a directed acyclic graph (DAG) that specifies a joint distribution over X as a product of local conditional distributions, one for each node:

$$\mathbb{P}(X_1 = x_1, \ldots, X_n = x_n) = \prod_{i=1}^{n} p(x_i | x_{\text{Parents}(i)})$$

Note that lowercase p is used to denote *local* conditional distribution (only conditioning on parents), which is specified as part of the network.

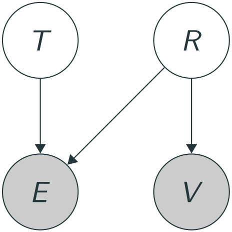


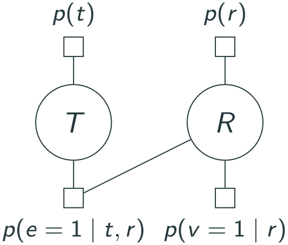


$$
\begin{aligned}
\mathbb{P}(T = t, R = r, E = e, V = v) &= p(t)p(r)p(e \mid t, r)p(v \mid r) \\
&= \frac{1}{Z} \prod \text{factors}
\end{aligned}
$$

Joint probability (Bayesian Network definition) is just Markov Network with normalization constant $Z = 1$.

Probabilistic Inference - Conditioning



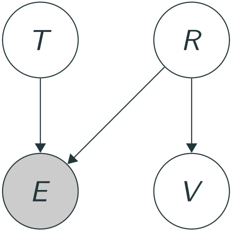


$$\mathbb{P}(T = t, R = r | E = 1, V = 1) = \frac{p(t)p(r)p(e = 1 \mid t, r)p(v = 1 \mid r)}{\mathbb{P}(E = 1, V = 1)}$$

Conditional probability (Bayesian Network with evidence) is just Markov Network with normalization $Z = \mathbb{P}(\text{Evidence})$.

Also $\mathbb{P}(A|B)\mathbb{P}(B) = \mathbb{P}(A)\mathbb{P}(B|A)$

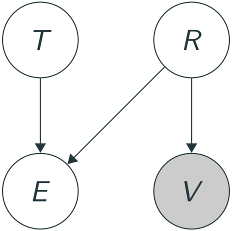
Probabilistic Inference - Unobserved Leaves



$$
\begin{aligned}
\mathbb{P}(T = t, R = r \mid E = 1) &= \sum_{v} \mathbb{P}(T = t, R = r, V = v \mid E = 1) \\
&= \sum_{v} \frac{p(t)p(r)p(e = 1 \mid t, r)p(v \mid r)}{\mathbb{P}(E = 1)} \\
&= \frac{p(t)p(r)p(e = 1 \mid t, r)}{\mathbb{P}(E = 1)}
\end{aligned}
$$

Throw away (marginalize out) unobserved leaves before inference.

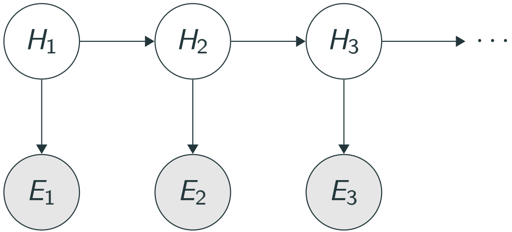
Probabilistic Inference - Independence



$$\begin{aligned}\mathbb{P}(T = t \mid V = 1) &= \sum_{r,e} \mathbb{P}(T = t, R = r, E = e \mid V = 1) \\ &= \sum_{r,e} \frac{p(t)p(r)p(e \mid t, r)p(v = 1 \mid r)}{\mathbb{P}(V = 1)} \\ &= p(t) \sum_{r} \frac{p(r)p(v = 1 \mid r)}{\mathbb{P}(V = 1)} = p(t)\end{aligned}$$

Ignore disconnected components.

Hidden Markov Models

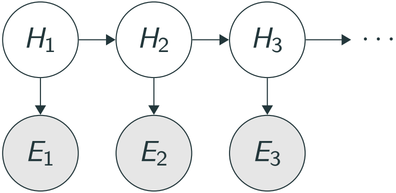


$$\mathbb{P}(H = h, E = e) = p(h_1) \prod_{i=2}^{n} p(h_i \mid h_{i-1}) \prod_{i=1}^{n} p(e_i \mid h_i)$$

Where $H = (H_1, \ldots, H_n)$ and $E = (E_1, \ldots, E_n)$.

True state moves in H, potentially inaccurate observations of H_i through E_i.

Hidden Markov Models



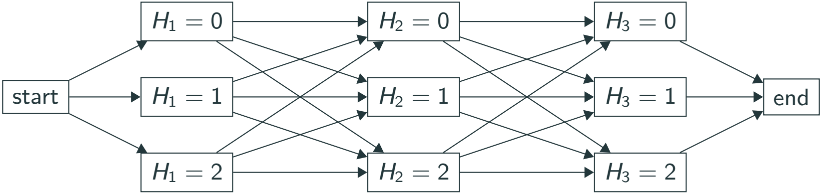
Two common questions:

1. Filtering: $\mathbb{P}(H_i \mid E_1, \ldots, E_i)$. Distribution of H_i given evidence to that point.
2. Smoothing: $\mathbb{P}(H_i \mid E_1, \ldots, E_n)$. Distribution of H_i given all evidence, including future.

Reviewing Lecture Material

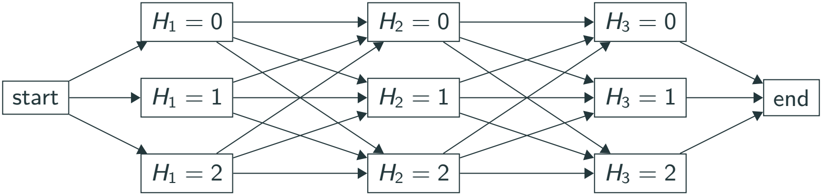
Lattices and Forward Backward

Lattice Representation



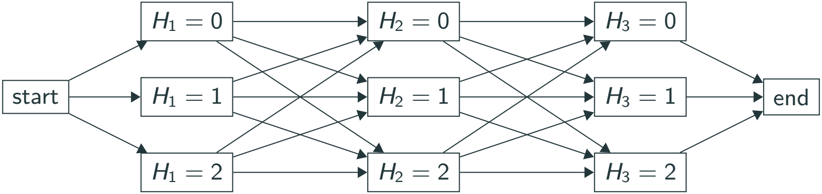
- Each edge is a probability/weight: the probability of transitioning with that edge in H space multiplied with the probability of what was observed (e_i, given true h_i).
- What is the weight on start$\rightarrow H_1 = 2$?

Lattice Representation



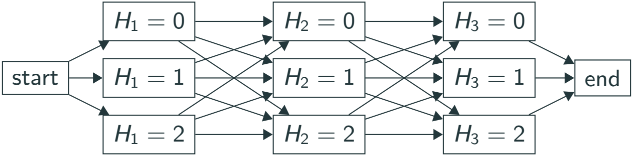
- Each edge is a probability/weight: the probability of transitioning with that edge in H space multiplied with the probability of what was observed (e_i, given true h_i).
- What is the weight on start$\rightarrow$ $H_1 = 2$?
  $p(h_1 = 2)p(e_1 \mid h_1 = 2)$
- What is the weight from $h_2 = x$ to $h_3 = y$ with $e_3 = 1$?

Lattice Representation



- Each edge is a probability/weight: the probability of transitioning with that edge in H space multiplied with the probability of what was observed (e_i, given true h_i).
- What is the weight on start$\rightarrow$ $H_1 = 2$?
  $p(h_1 = 2)p(e_1 \mid h_1 = 2)$
- What is the weight from $h_2 = x$ to $h_3 = y$ with $e_3 = 1$?
  $p(h_3 = y \mid h_2 = x)p(e_3 = 1 \mid h_3 = y)$

Lattice Representation



$$\mathbb{P}[\text{edge}] = p(h_i \mid h_{i-1})p(e_i \mid h_i)$$

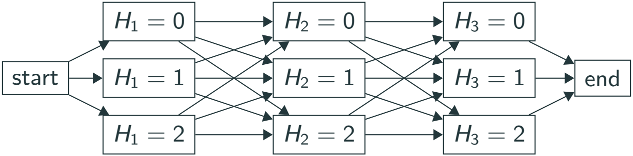
Start to end paths are just $P(H = h, E = e)$ by definition of Bayesian Networks.

What if we want conditional probability (smoothing)?

$$\mathbb{P}(H_j = h_j \mid E = e) = \sum_{h \in H_{-j}} \frac{p(h_1)\prod_{i=1}^{n} p(e_i \mid h_i)\prod_{i=2}^{n} p(h_i \mid h_{i-1})}{\mathbb{P}(E = e)}$$

Numerator is sum of cost of all paths through $H_j = h_j$, normalized by probability of observed evidence.

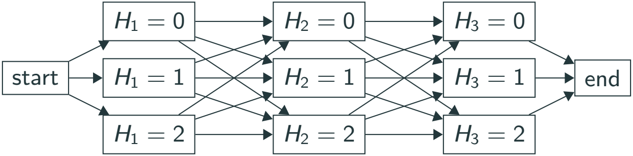
Forward Backward



Need cost of all paths through a given node.

- **Forward**:
  $F_i(h_i) = \sum_{h_{i-1}} F_{i-1}(h_{i-1})\text{Weight}(H_{i-1} = h_{i-1}, H_i = h_i)$
  - Sum of weights of paths from start to $H_i = h_i$.
- **Backward**:
  $B_i(h_i) = \sum_{h_{i+1}} B_{i+1}(h_{i+1})\text{Weight}(H_i = h_i, H_{i+1} = h_{i+1})$
  - Sum of weights of paths from $H_i = h_i$ to end.
- **Total**: $S_i(h_i) = F_i(h_i)B_i(h_i)$
  - Sum of weights of paths from start to end through $H_i = h_i$.

Forward Backward



Recall we wanted to compute:

$$\mathbb{P}(H_j = h_j \mid E = e) = \sum_{h \in H_{-j}} \frac{p(h_1) \prod_{i=1}^{n} p(e_i \mid h_i) \prod_{i=2}^{n} p(h_i \mid h_{i-1})}{\mathbb{P}(E = e)}$$

Numerator was cost of all paths through $H_j = h_j$ given evidence $E = e$. This is $S_j(h_j)$!

Denominator? Sum of weights of *all* paths given the evidence.

$$\mathbb{P}(E = e) = \sum_{h_k} S_j(h_k)$$

Reviewing Lecture Material

Particle Filtering

Particle Filtering

- Forward Backward (smoothing) is $O(n|\text{Domain}|^2)$. Too slow!
- Use particle filtering for approximate probabilistic inference.
- Can ignore improbable locations (low probability) given the evidence.
- Sacrifice accuracy for speed!

Beam Search for HMMs

- Initialize $C \leftarrow [\{\}]$
- For each $i = 1, \ldots, n$:
  - **Extend:** $C' \leftarrow \{h \cup \{H_i : v\} \ : \ h \in C, v \in \text{Domain}_i\}$
  - Create new C' by joining existing entries $h \in C$ with all possible $H_i = v$.
  - **Prune:** $C \leftarrow K$ particles of C' with highest weights (beam).
- Normalize weights to get approximate $\hat{\mathbb{P}}(H_1, \ldots, H_n \mid E = e)$
- Sum probabilities to get any approximate $\hat{\mathbb{P}}(H_i \mid E = e)$

Extending is slow (considers all possible next values) and prune is greedy (not always the best).

Particle Filtering

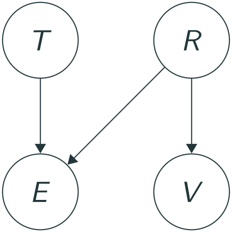
Rather than **extend** (exhaustive) and **prune** (greedy), we run the following steps to generate each next entry in H:

1. Propose: for each particle $(h_1, \ldots, h_i)$ sample $H_{i+1} \sim p(h_{i+1} \mid h_i)$.
2. Weight: For each existing particle $(h_1, \ldots, h_{i+1})$, weight it by probability of observed e_{i+1}, $p(e_{i+1} \mid h_{i+1})$.
3. Resample: What if particles have really small weight from previous step? Normalize the weights, resample K particles $(h_1, \ldots, h_{i+1})$ using those weights.

Reviewing Lecture Material

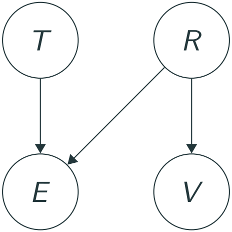
Supervised Learning

Supervised Learning



Where do the parameters come from? Need local conditional distributions, but how?

Supervised Learning



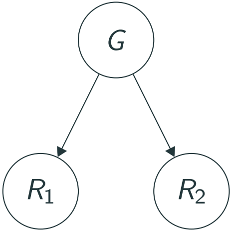
Where do the parameters come from? Need local conditional distributions, but how?

Counting!

- Data: Example assignments of all variables (X).
- Use this to determine local condition probabilities (θ).

Parameter Sharing

- **Parameter Sharing**: Local conditional distributions of different variables can share the same parameters. $p(R_1 = r \mid g) = p(R_2 = r \mid g)$.
- In HMMs, rather than $p(h_i \mid h_{i+1})$ and $p(e_i \mid h_i)$ for all i, could just have $p_{\text{start}}, p_{\text{transition}}, p_{\text{emit}}$. Less expressive but easier to learn!



$$\mathbb{P}(X = x) = \prod_{i=1}^{n} p_{d_i}(x_i \mid x_{\text{Parents(i)}})$$

Counting!

Input: Full assignments $x \in \mathcal{D}_{\text{train}}$

Output: Parameters $\theta = \{p_d \ : \ d \in D\}$ (D is collection of distributions)

- Count: For each $x_i \in x \in \mathcal{D}_{\text{train}}$
  - Increment $\text{count}_{d_i}(x_{\text{Parents}(i)}, x_i)$
- Normalize: For each d and local assignment $x_{\text{Parents}(i)}$:
  - Set $p_d(x_i \mid x_{\text{Parents}(i)}) \propto \text{count}_d(x_{\text{parents}(i)}, x_i)$

This is just the closed form solution of the maximum likelihood objective:

$$\max_{\theta} \prod_{x \in \mathcal{D}_{\text{train}}} \mathbb{P}(X = x; \theta)$$

Reviewing Lecture Material

EM Algorithm

EM Algorithm

What happens if we don't observe some variables (e.g. hidden ones)? Can't count!

Assume that H is hidden but we observe $E = e$. Maximize the probability of observing e using our parameter θ:

$$\max_{\theta} \prod_{e \in \mathcal{D}_{\text{train}}} \mathbb{P}(E = e; \theta) = \max_{\theta} \prod_{e \in \mathcal{D}_{\text{train}}} \sum_{h} \mathbb{P}(H = h, E = e; \theta)$$

Marginalize out what we can't observe - Maximum Marginal Likelihood

EM Algorithm

Generalization of K-means, centroids become parameters θ and the cluster assignments are the hidden variables H.

- Initialize θ randomly (parameters of our distributions)
- Repeat until convergence:
  - E-Step: Compute $q(h) = \mathbb{P}(H = h \mid E = e; \theta)$ for each h (Bayesian inference)
    - Create fully-observed weighted examples (h, e) with weight $q(h)$
  - M-Step: Maximum likelihood (count and normalize) on weighted examples to get new θ (weight each appearance by $q(h)$)

Reviewing Lecture Material

Summary

HMM Algorithms

- Forward Backward
  - Dynamic programming for inference, exact.
- Particle Filtering
  - Use particles to represent approximate H distributions.
  - Scales to large H space (unlike forward-backward).
  - Maintains better particle diversity (compared to beam search).
- Learning local conditional distributions
  - Maximum Likelihood (counting and normalizing)
  - EM Algorithm for hidden variables.

Problems

Reviewing Lecture Material

- Bayesian Networks and HMMs
- Lattices and Forward Backward
- Particle Filtering
- Supervised Learning
- EM Algorithm
- Summary

Problems