6.891 Machine learning and neural networks

Tommi Jaakkola
MIT AI LAB
tommi@ai.mit.edu

LECTURE 20 (November 21, 2000)

Future lectures:

No lecture (Thanksgiving)	–	Thu Nov 23
No lecture	–	Tue Nov 28
Lecture 21 (Prof. Leslie Kaelbling)	–	Thu Nov 30

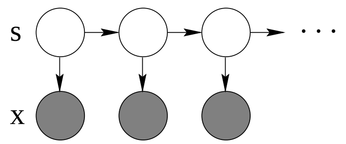
Topics

- Inference in graphical models
  - chains, trees
  - general markov random fields
  - general bayesian networks
- Estimation in graphical models
  - maximum likelihood, maximum a posteriori (MAP)
- (Learning the graph structure)

Nature of probabilistic inference

- Example: a hidden Markov model

$$P(s_0, x_0, \ldots, s_n, x_n) = P_0(s_0)\, P_o(x_0|s_0)\, P_1(s_1|s_0) \cdots \qquad (1)$$



- Given the observation sequence $x_0^*, \ldots, x_n^*$, all the information about the associated hidden states is already contained in the joint probability distribution

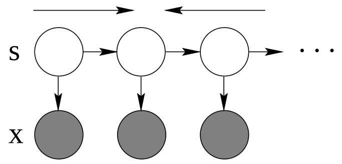
$$P(s_0, x_0^*, \ldots, s_n, x_n^*) \qquad (2)$$

- What's left to do?

Nature of probabilistic inference

- In probabilistic inference we wish to *explicate* the relevant information

  This involves propagation of information across the graph model
- Forward-backward algorithm:



  Forward step: information from the past about the current state

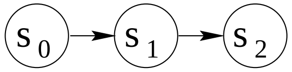
  Backward step: information from future observations about the current state
- We want analogous computations for more general graph models

Objective

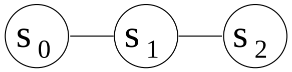
- Our objective: to exploit the graph structure for efficient inference calculations
- For this we need
  1. to define an appropriate data structure (junction tree) where these calculations can be made
  2. to specify how information is propagated in such structures
- Note: the resulting propagation algorithms do not *change* information, they merely transform it so that the relevant information is available *locally* and *explicitly*

Simple Markov chain example

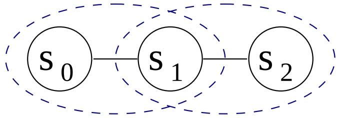
- Take a simple Markov chain



- We can replace the directed edges with undirected edges without affecting the graph semantics (or the underlying probability model)

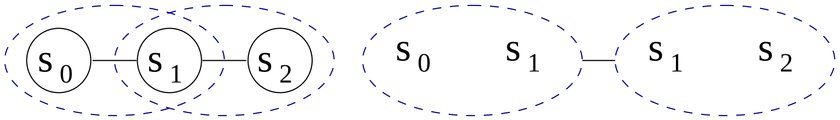


- We can identify the cliques in the undirected graph

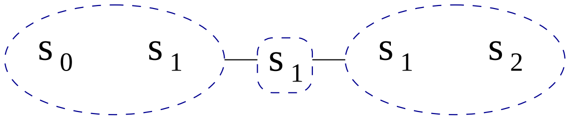


Markov chain example cont'd

- The cliques can be connected into a hyper-graph (two cliques are connected if they have variables in common)

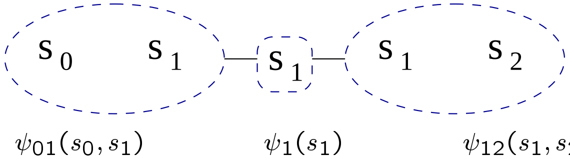


- Finally, we can explicate the overlap between the cliques by defining *separators*, sets of variables that the cliques have in common



- This is known as a *junction tree*

Probabilities and junction tree

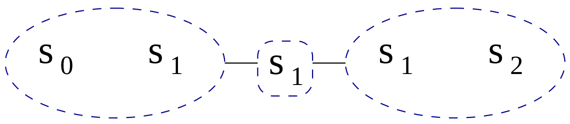


- The joint distribution over the markov chain can be defined by associating potential functions with the cliques (and the separator)

$$P(s_0, s_1, s_2) = \frac{P(s_0, s_1)\, P(s_1, s_2)}{P(s_1)} = \frac{\psi_{01}(s_0, s_1)\, \psi_{12}(s_1, s_2)}{\psi_1(s_1)} \quad (3)$$

- Note that the information (marginal probabilities) about the variables in the cliques resides *locally* and is *explicit*

Evidence in a junction tree



$\psi_{01}(s_0, s_1)$ $\psi_1(s_1)$ $\psi_{12}(s_1, s_2)$

- When we acquire evidence about the values of the variables, the relevant information need not be local nor explicit any more

$$P(s_0^*, s_1, s_2) = \frac{P(s_0^*, s_1)\, P(s_1, s_2)}{P(s_1)} = \frac{\psi_{01}(s_0^*, s_1)\, \psi_{12}(s_1, s_2)}{\psi_1(s_1)} \quad (4)$$

  Here we assume that we have observed s_0^*

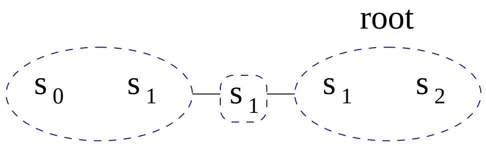
- To incorporate this evidence, we multiply the corresponding clique potential with an indicator function

$$\psi_{01}(s_0, s_1) \leftarrow \psi_{01}(s_0, s_1)\, \delta(s_0, s_0^*) \quad (5)$$

  where $\delta(s_0, s_0^*) = 1$ if $s_0 = s_0^*$ and zero otherwise.
  (potential is zero if we contradict the evidence)

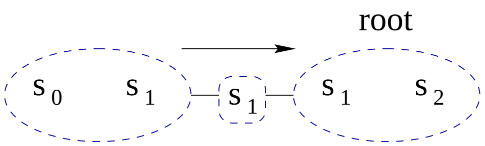
Junction tree algorithm

- To achieve locality, we must propagate the relevant information from other parts of the model to the appropriate cliques



- The junction tree (clustering) algorithm
  1. Pick a root node
  2. Collect information towards the root
  3. Distribute information away from the root
- Why do we need the two passes?

Collect operation



$\psi_{01}(s_0, s_1)$ $\psi_1(s_1)$ $\psi_{12}(s_1, s_2)$

- Compute a target for the separator

$$\psi_1'(s_1) \leftarrow \sum_{s_0} \psi_{01}(s_0, s_1) \quad (= P(s_0^*, s_1)\,) \quad (6)$$

- Update the collecting clique (root in our case) based on this *new* information

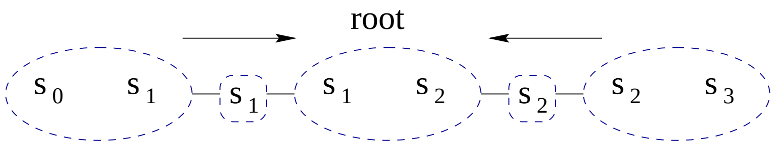
$$\psi_{12}(s_1, s_2) \leftarrow \frac{\psi_1'(s_1)}{\psi_1(s_1)}\, \psi_{12}(s_1, s_2) \quad (7)$$

  (nothing would change if there were no evidence)

- Update the separator

$$\psi_1(s_1) \leftarrow \psi_1'(s_1) \quad (8)$$

General collect operation

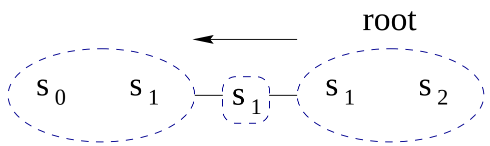


- When collecting from multiple neighbors, we must update the cliques based on all the incoming information

$$\psi_{12}(s_1, s_2) \leftarrow \frac{\psi_1'(s_1)}{\psi_1(s_1)} \frac{\psi_2'(s_2)}{\psi_2(s_2)}\, \psi_{12}(s_1, s_2) \quad (9)$$

  (again nothing would change if there were no evidence)

Distribute operation



$\psi_{01}(s_0, s_1)$ $\psi_1(s_1)$ $\psi_{12}(s_1, s_2)$

- Compute a target value for the separator (now in the other direction)

$$\psi_1'(s_1) \leftarrow \sum_{s_2} \psi_{12}(s_1, s_2) \tag{10}$$

- Update the receiving clique based on this *new* information

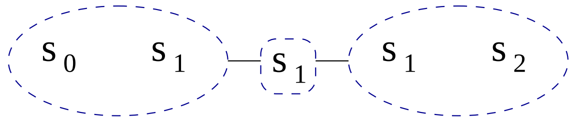
$$\psi_{01}(s_0, s_1) \leftarrow \frac{\psi_1'(s_1)}{\psi_1(s_1)} \psi_{01}(s_0, s_1) \tag{11}$$

(note that in our case there's is no new information; update doesn't change anything)

- Finally, we update the separator

$$\psi_1(s_1) \leftarrow \psi_1'(s_1) \tag{12}$$

Junction tree



$\psi_{01}(s_0, s_1)$ $\psi_1(s_1)$ $\psi_{12}(s_1, s_2)$

- After both propagation operations, the relevant information is again stored locally

$$\psi_{01}(s_0, s_1) \propto P(s_0, s_1|\text{evidence}) \tag{13}$$
$$\psi_1(s_1) \propto P(s_1|\text{evidence}) \tag{14}$$
$$\psi_{12}(s_1, s_2) \propto P(s_1, s_2|\text{evidence}) \tag{15}$$

Initialization

- Do we have to always start with the perfect solution?

$$\psi_{01}(s_0, s_1) = P(s_0, s_1) \tag{16}$$
$$\psi_1(s_1) = P(s_1) \tag{17}$$
$$\psi_{12}(s_1, s_2) = P(s_1, s_2) \tag{18}$$

- We could, for example, initially set

$$\psi_{01}(s_0, s_1) = P(s_0)P(s_1|s_0) \tag{19}$$
$$\psi_1(s_1) = 1 \tag{20}$$
$$\psi_{12}(s_1, s_2) = P(s_2|s_1) \tag{21}$$

and then apply the collect and distribute operations to achieve the "perfect solution" given above

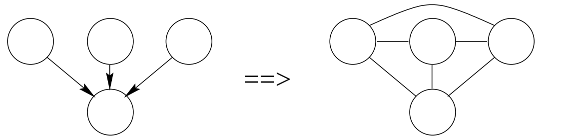
(the separators are initialized to unity; each conditional probability is assigned to (any single) clique that fully contains the associated variables)

Junction tree construction

- The junction tree algorithm involves the following preprocessing steps
  1. Transform a Bayesian network into an undirected graph model
  2. Identify the cliques in the undirected graph
  3. Construct the junction tree from the hypergraph (graph where cliques serve as nodes)
- These are graph operations and need to be done only once for any fixed graph structure
- In general, we need a couple of additional steps ...
  - moralization
  - triangulation

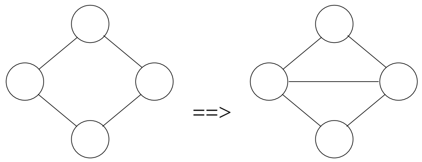
More general preprocessing steps

- *Moralization*



Marry the parents (connect all the parents and drop the arrows)

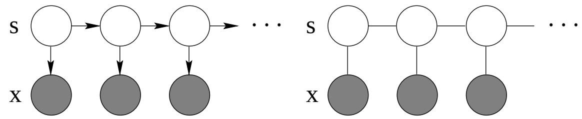
- *Triangulation*



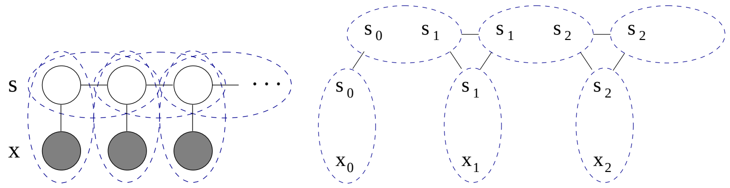
Triangulation permits us to organize the resulting cliques into a tree structure (junction tree)

Example: hidden Markov model

- First we transform the HMM into an undirected graph model

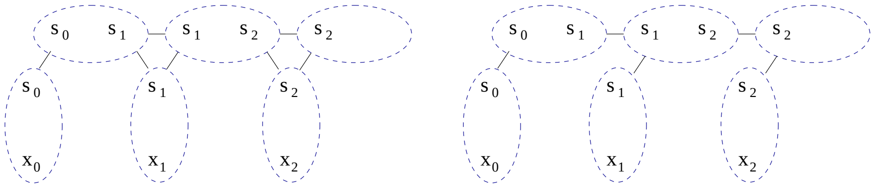


- The resulting undirected graph is already triangulated
- Then we identify the cliques and construct the hyper-graph by connecting cliques that have variables in common

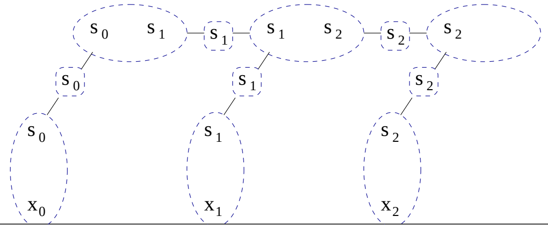


Example cont'd

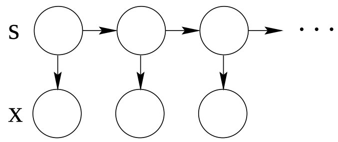
- The hyper-graph can be transformed into a tree structure by dropping edges so long as any non-adjacent cliques that have variables in common still have these variables in the path between them



- Finally, we can construct the junction tree for HMMs



Parameter estimation in Bayesian networks

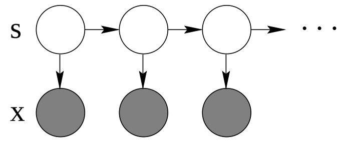


- We distinguish two cases
  1. Complete observations where we have a value assignment for each variable in the graph model
  2. Incomplete observations (much harder)
- When we have complete observations, the estimation problem reduces to n independent problems of estimating each conditional probability separately (e.g., via maximum likelihood)

$$\max \sum_{t=1}^{T} \log P_o(x_0^t|s_0^t) \tag{22}$$

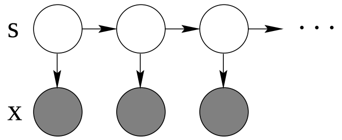
Parameter estimation with incomplete data

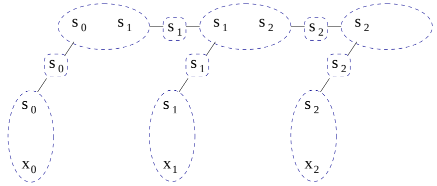
- In graph models such as HMMs we often have to estimate the parameters based on *incomplete observations*, observations that do not provide a value assignment for all the variables in the model



- In these cases, we need to infer (soft) value assignments based on the incomplete information
- The EM algorithm is applicable for general graphical models

Parameter estimation





- The EM-algorithm

  E-step: compute the posterior probabilities over the relevant variables such as $\hat{P}(s_0, x_0|t^{th}$ obs.)

  We obtain these probabilities directly from the junction tree algorithm

  M-step: estimate the conditional probabilities such as $P_o(x_0|s_0)$ by maximizing the likelihood of the inferred value assignments

$$\max \sum_{t=1}^{T} \sum_{s_0, x_0} \hat{P}(s_0, x_0|t^{th} \text{ obs.}) \log P_o(x_0|s_0) \qquad (23)$$

Structure estimation

...