Brief Introduction to Factor Graphs

Part 1

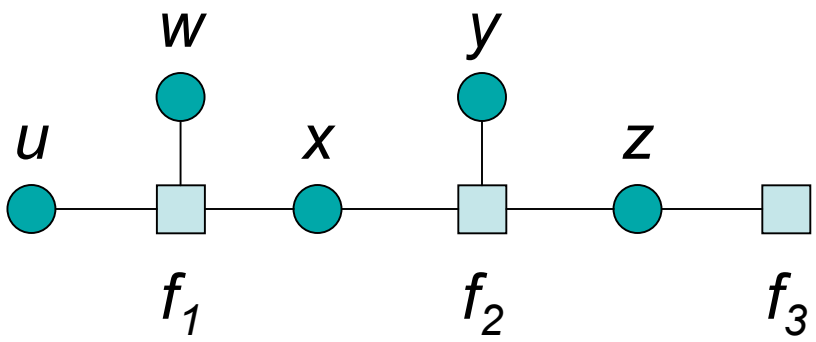
9/29/2008

Background

- Tame combinatorics in many calculations
  - Decoding codes (origin of factor graphs)
  - Bayesian networks and Markov random fields
  - HMMs (and signal processing more generally)
  - *Production match?*
- Form of divide-and-conquer with *nearly decomposable* components
- Many standard state-of-the-art algorithms can be derived from factor graph *sum-product algorithm*
  - Belief propagation in Bayesian networks
  - Forward-backward algorithm in HMMs
  - Kalman filter, Viterbi algorithm, FFT, turbo decoding
  - *Rete algorithm (or equivalent)?*

Factor Graphs

- Decompose functions into product of nearly independent *factors* (or *local functions*)
  - E.g., $f(u,w,x,y,z) = f_1(u,w,x)f_2(x,y,z)f_3(z)$
- Draw as bipartite graph
  - Nodes for factors and variables
  - Links between factors and their variables



Bayesian Network (BN) Example

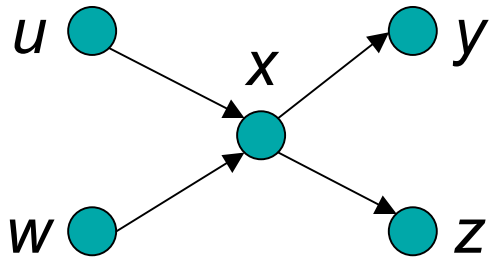
- Probabilistic reasoning involves computations of various quantities from *joint probability distributions* over random variables
  - E.g., compute the *marginal probability* $p(u)$ from the joint probability distribution $p(u,w,x,y,z)$ by summing out/over all of the other variables
    - $p(u) = \text{SUM}_{w,x,y,z}\, p(u,w,x,y,z)$
  - Key for tractability is to do so without having to explicitly examine every combination of values of all of the other variables

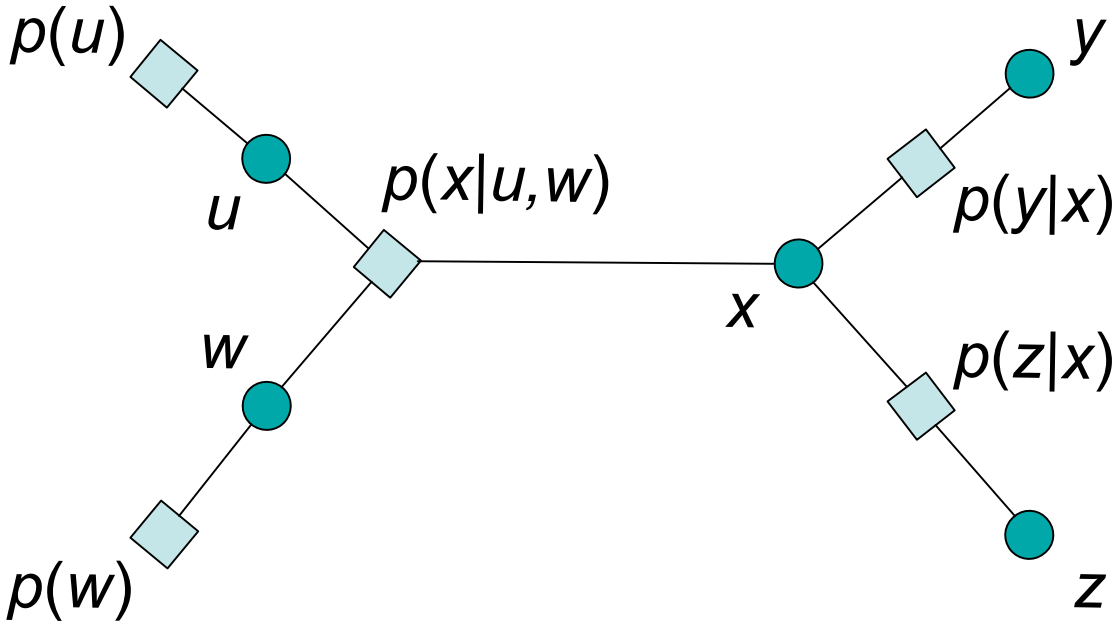
BN Example (cont.)

- A Bayesian network represents a joint probability distribution as the product of the conditional probabilities of each random variable
  - E.g., $p(u,w,x,y,z) = p(u)p(w)p(x|u,w)p(y|x)p(z|x)$
- Each conditional probability only involves a subset of the total set of variables (its *parents*)
  - Each variable is *conditionally independent* of all of the other variables, given its parents
    - I.e., once you know the values of the parent variables, the probability of a value of the variable can be determined independently of the values of all of the other variables
- Can radically reduce scope of combinatorics

Example BN and Factor Graph

$p(u,w,x,y,z) = p(u)p(w)p(x|u,w)p(y|x)p(z|x)$





Production Match Example

C_1: (<a> ^type <b>)

C_2: (<a> ^name X)

C_3: (<b> ^type <c>)

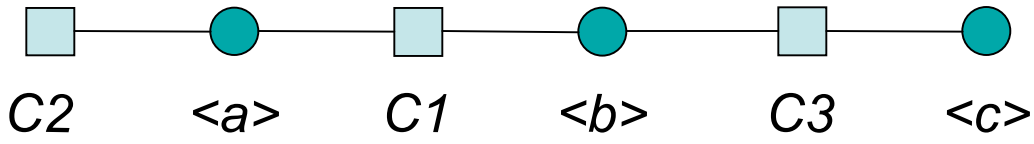
→

(<b> ^name Y)

For a factor graph

- Variables are variables
- Conditions are factors

$PM(a,b,c)=C_1(a,b)C_2(a)C_3(b,c)$



The Sum-Product Algorithm

- Can be viewed as a message-passing algorithm where each node is a processor and the links are communication channels between them
- "The message sent from a node *v* on an edge *e* is the product of the local function at *v* (or the unit function if *v* is a variable node) with all messages received at *v* on edges other than *e*, summarized for the variable associated with *e*."
  - *Summarization* is summing out/over all other variables
  - Although, some variations use *max* rather than *sum*
    - *Summary product algorithm* generalizes over such variations as long as the two operations (product and *summary*) together define a *semi-ring*
- *More details on this algorithm in our next meeting*

Why Should We Care?

- Uniform representation&algorithm for signals&symbols
  - Can it yield a deeper integration and/or smoother interface?
- Uniform representation&algorithm for probability&rules
  - Can it yield a deeper integration and/or smoother interface?
- Potentially provide underlying commonality between procedural and declarative memory
  - Can it provide a more principled/unified approach to both?
- Can it go beyond HMMs in sequential processing to model the MDPs used in Theory of Mind and/or whatever is needed for episodic memory?
- *Can it provide a uniform technology layer (a high-level language) for constructing and exploring the modules and processes needed in a cognitive architecture?*