Making Predictions with Dynamic Programming

SPFLODD

October 4, 2011

Inference

- Eventually, you need to run your structured predictor on test data!
- Till now we have focused on learning mostly.
- For sequence labeling and segmentation models with local interactions, decoding was pretty much always the Viterbi algorithm.
- Variations:
  - forward-backward
  - cost-augmented decoding

Notation for Linear Models

- Training data: $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$
- Testing data: $\{(x_{N+1}, y_{N+1}), \ldots (x_{N+N'}, y_{N+N'})\}$
- Feature function: $\mathbf{g}$
- Weights: $\mathbf{w}$
- Decoding:

$$\text{decode}(\mathbf{w}, \boldsymbol{x}) \quad = \quad \arg\max_{\boldsymbol{y}} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y})$$

- Learning:

$$\text{learn}\left(\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N\right) \quad = \quad \arg\max_{\mathbf{w}} \Phi\left(\mathbf{w}, \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N\right)$$

- Evaluation:

$$\frac{1}{N'} \sum_{i=1}^{N'} \text{cost}\left(\text{decode}\left(\text{learn}\left(\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N\right), \boldsymbol{x}_{N+i}\right), \boldsymbol{y}_{N+i}\right)$$

Probabilistic Inference Problems

Given values for some random variables ($\mathbf{X} \subset \mathbf{V}$) …

- *Most Probable Explanation: what are the *most probable* values of the *rest* of the r.v.s $\mathbf{V} \setminus \mathbf{X}$?

(More generally …)

- *Maximum *A Posteriori* (MAP): what are the most probable values of *some* other r.v.s, $\mathbf{Y} \subset (\mathbf{V} \setminus \mathbf{X})$?

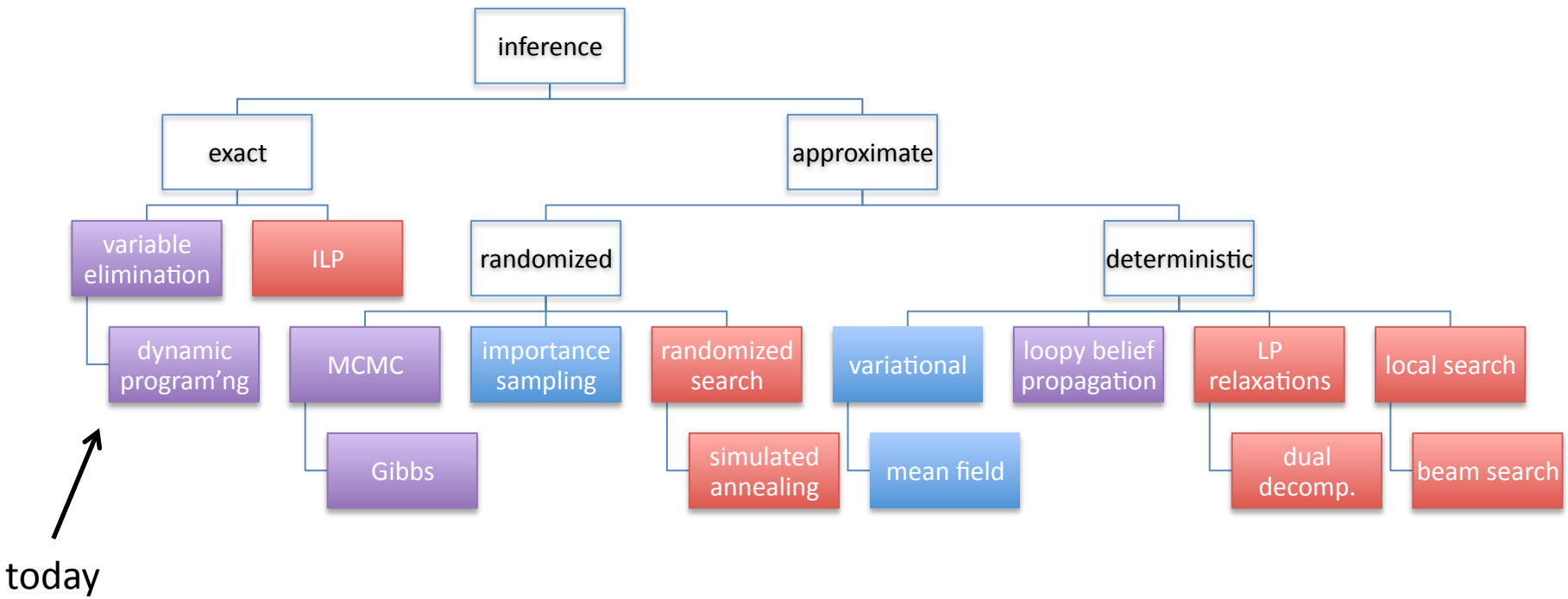
These do not need to be probabilistic! Change "most probable" to "maximum scoring."

- Random sampling from the posterior over values of **Y**
- Full posterior over values of **Y**
- Marginal probabilities from the posterior over **Y**

- *Minimum Bayes risk: What is the **Y** with the lowest expected cost?
- *Cost-augmented decoding: What is the most *dangerous* value of **Y**, compared to true **y***?

*Different kinds of **decoding**.

Approaches to Inference



Dynamic Programming

- Examples you've seen:
  - Viterbi, forward, backward algorithms for HMMs
  - Minimum-cost edit distance
  - Dijkstra's shortest path
- Can we define it more broadly?

General Views of Decoding

- For HMMs, the decoding algorithm we usually think of first is the Viterbi algorithm.
  - This is just one example.

- We will view decoding in a few different ways.
  - Sequence models as a running example.
  - These views are not just for HMMs.
  - Sometimes they will lead us back to Viterbi!

Different Views of Decoding

1. Probabilistic Graphical Models

- View the linguistic structure as a collection of random variables that are interdependent.
- Represent interdependencies as a directed or undirected graphical model.
- Conditional probability tables (BNs) or factors (MNs) encode the probability distribution.

Inference in Graphical Models

- General algorithm for exact MAP inference: variable elimination.
  - Iteratively solve for the best values of each variable conditioned on values of "preceding" neighbors.
  - Then trace back.

The Viterbi algorithm is an instance of max-product variable elimination!

Hidden Markov Model

- **X** and **Y** are both sequences of symbols
  - **X** is a sequence from the vocabulary Σ
  - **Y** is a sequence from the state space Λ

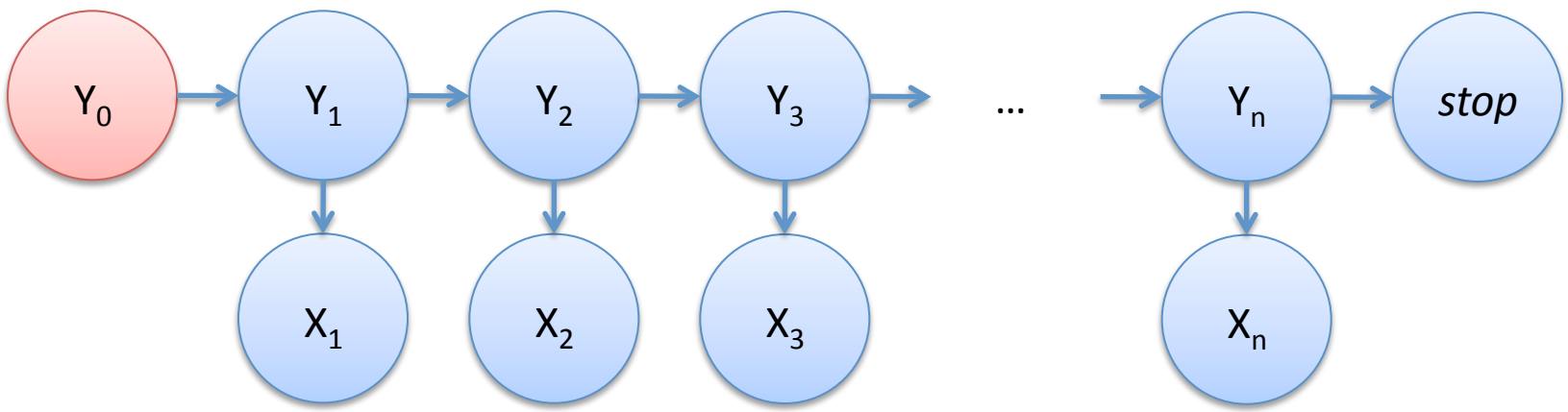
$$p(\boldsymbol{X} = \boldsymbol{x}, \boldsymbol{Y} = \boldsymbol{y}) \quad = \quad \left(\prod_{i=1}^{n} p(x_i \mid y_i) p(y_i \mid y_{i-1}) \right) p(stop \mid y_n)$$

- Parameters:
  - Transitions p(y' | y)
    - including p(*stop* | y), p(y | *start*)
  - Emissions p(x | y)

Hidden Markov Model

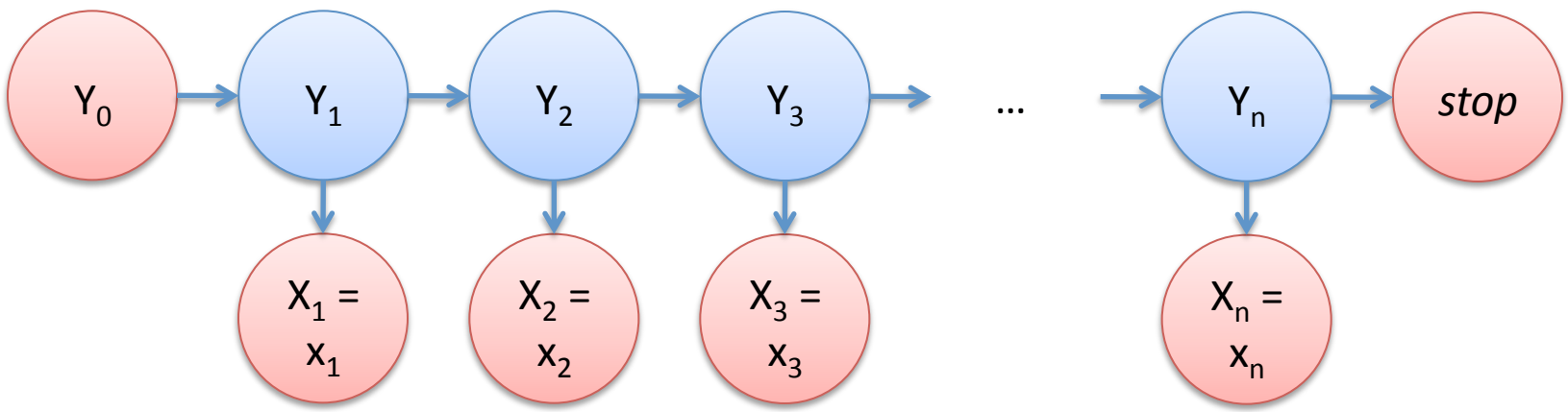
- The joint model's independence assumptions are easy to capture with a Bayesian network.

$$p(\boldsymbol{X} = \boldsymbol{x}, \boldsymbol{Y} = \boldsymbol{y}) \quad = \quad \left(\prod_{i=1}^{n} p(x_i \mid y_i) p(y_i \mid y_{i-1}) \right) p(stop \mid y_n)$$



Hidden Markov Model

- The usual inference problem (decoding) is to find the most probable value of **Y** given **X** = **x**.
  - Instantiates "most probable explanation" inference and "maximum a posteriori" inference.



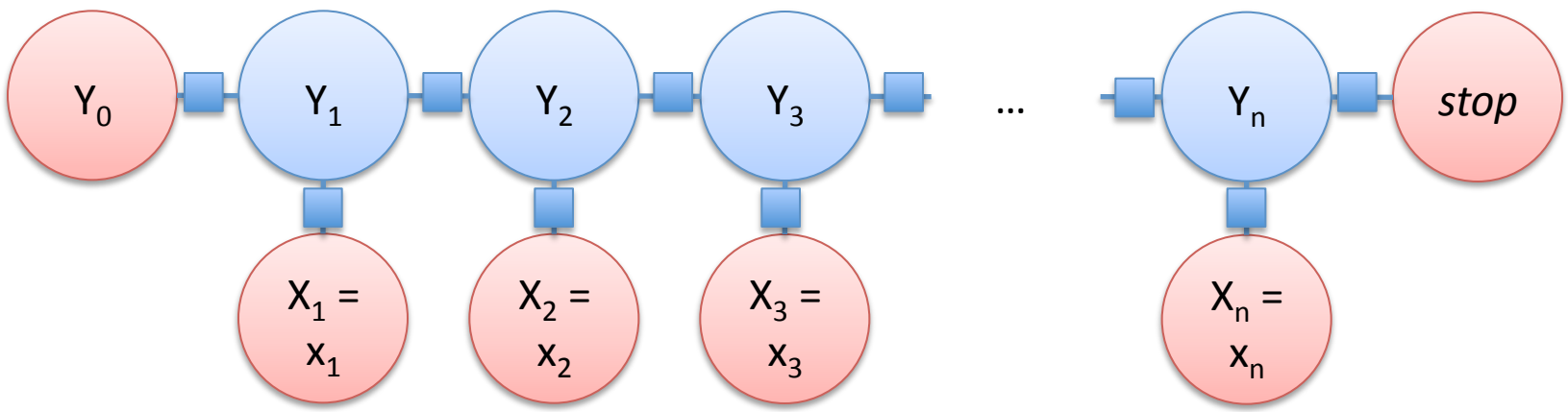
A Graphical Models View of Viterbi

1. Convert to a factor graph.
2. Reduce factors to respect observation at each position.
3. Do variable elimination from left to right.
   - Calculates the best prefix-path scores into each state at each position.
4. Trace back to recover the labels.

Hidden Markov Model

- The usual inference problem is to find the most probable value of **Y** given **X** = **x**.

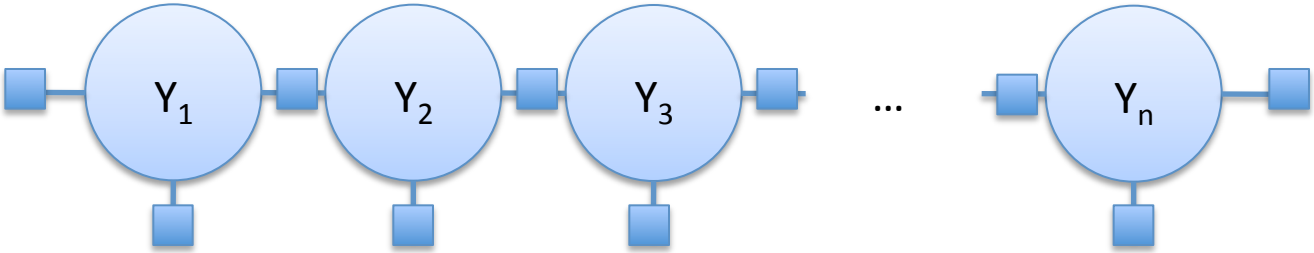
- Factor graph:



Hidden Markov Model

- The usual inference problem is to find the most probable value of **Y** given **X** = **x**.

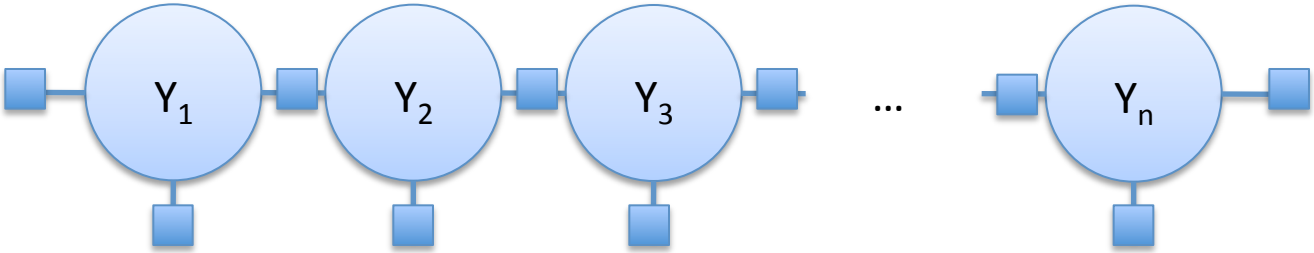
- Factor graph after reducing factors to respect evidence:



Hidden Markov Model

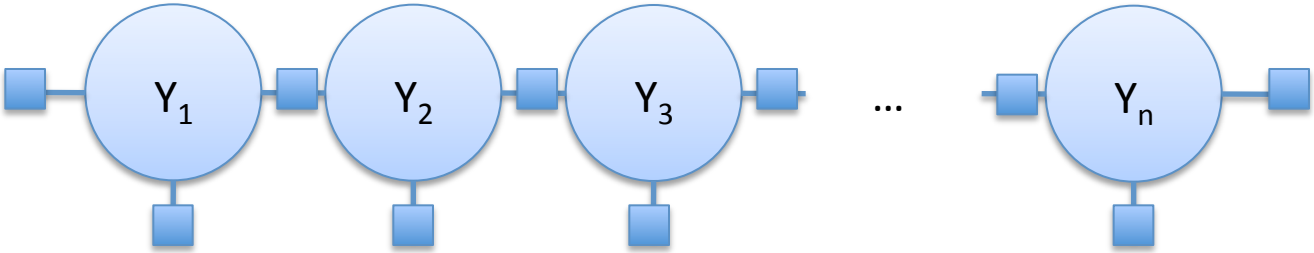
- The usual inference problem is to find the most probable value of **Y** given **X** = **x**.

- Clever ordering should be apparent!



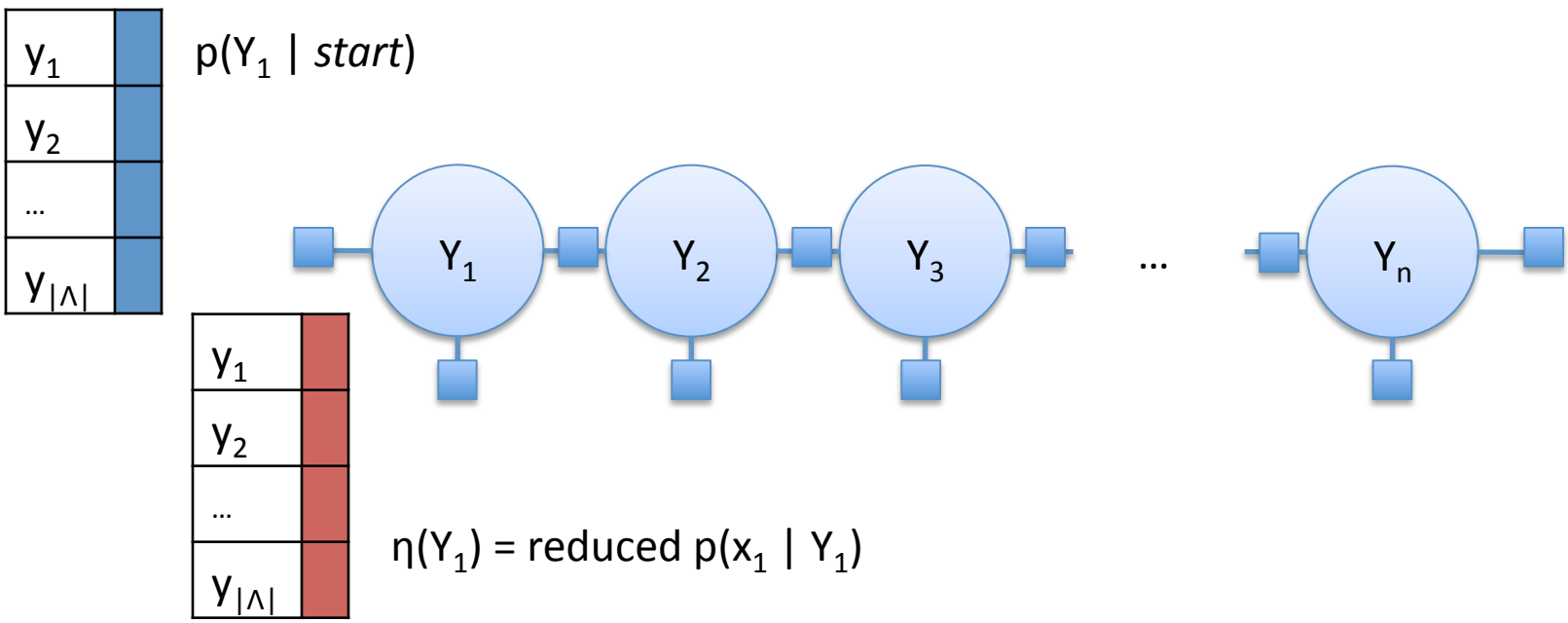
Hidden Markov Model

- When we eliminate Y_1, we take a product of three relevant factors.
  - $p(Y_1 \mid \textit{start})$
  - $\eta(Y_1)$ = reduced $p(x_1 \mid Y_1)$
  - $p(Y_2 \mid Y_1)$



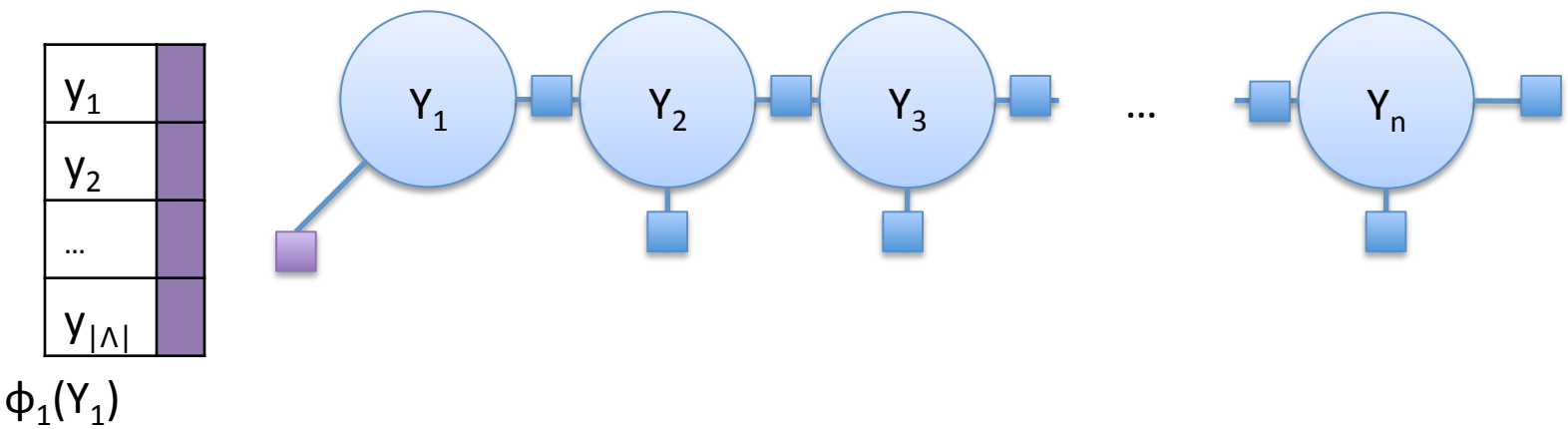
Hidden Markov Model

- When we eliminate Y_1, we first take a product of two factors that only involve Y_1.



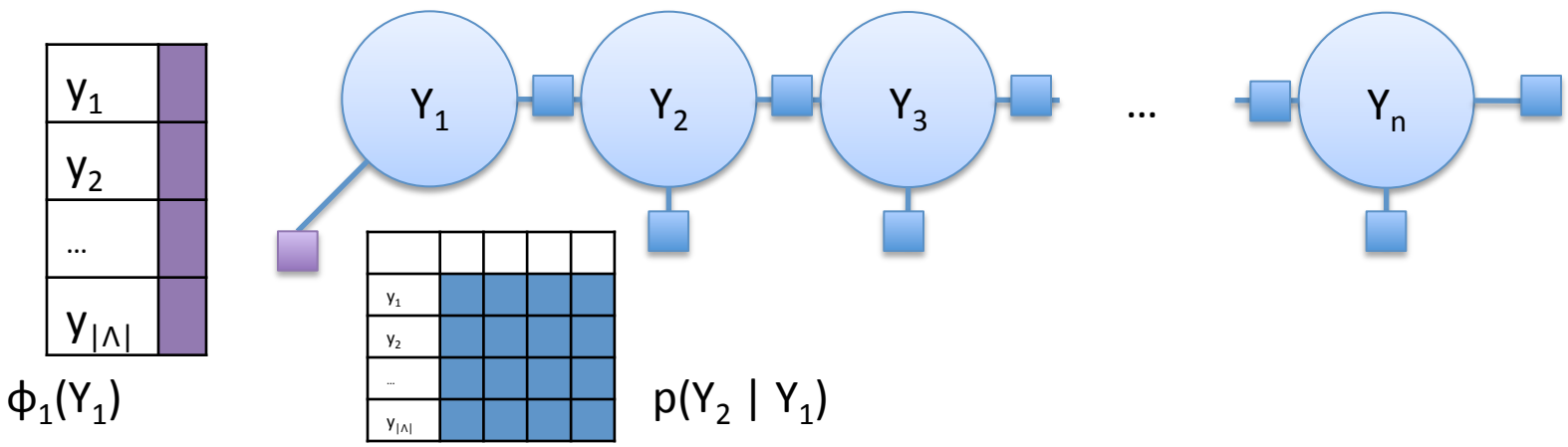
Hidden Markov Model

- When we eliminate Y_1, we first take a product of two factors that only involve Y_1.
- This is the Viterbi probability vector for Y_1.



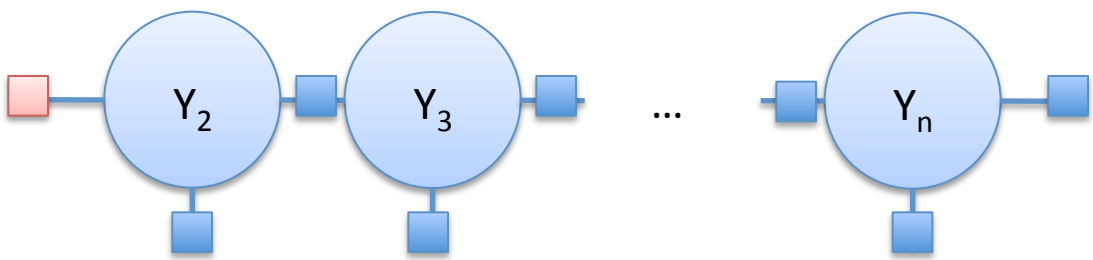
Hidden Markov Model

- When we eliminate Y_1, we first take a product of two factors that only involve Y_1.
- This is the Viterbi probability vector for Y_1.
- Eliminating Y_1 equates to solving the Viterbi probabilities for Y_2.



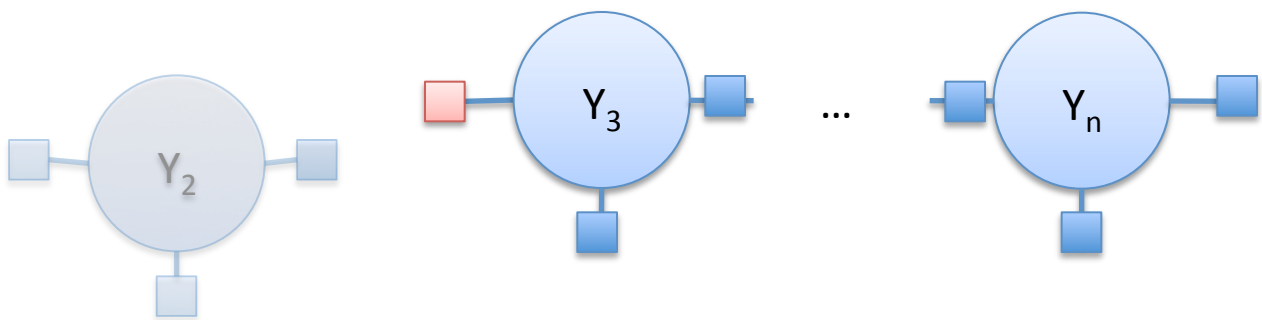
Hidden Markov Model

- Product of all factors involving Y_1, then reduce.
  - $\phi_2(Y_2) = \max_{y \in Val(Y_1)} (\phi_1(y) \times p(Y_2 \mid y))$
  - This factor holds Viterbi probabiliies for Y_2.



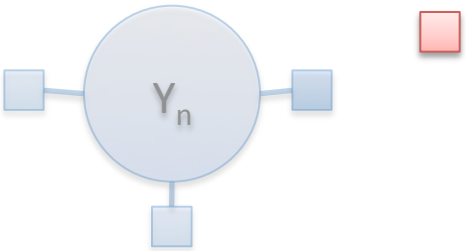
Hidden Markov Model

- When we eliminate Y_2, we take a product of the analogous two relevant factors.
- Then reduce.
  - $\phi_3(Y_3) = \max_{y \in Val(Y_2)} (\phi_2(y) \times p(Y_3 \mid y))$



Hidden Markov Model

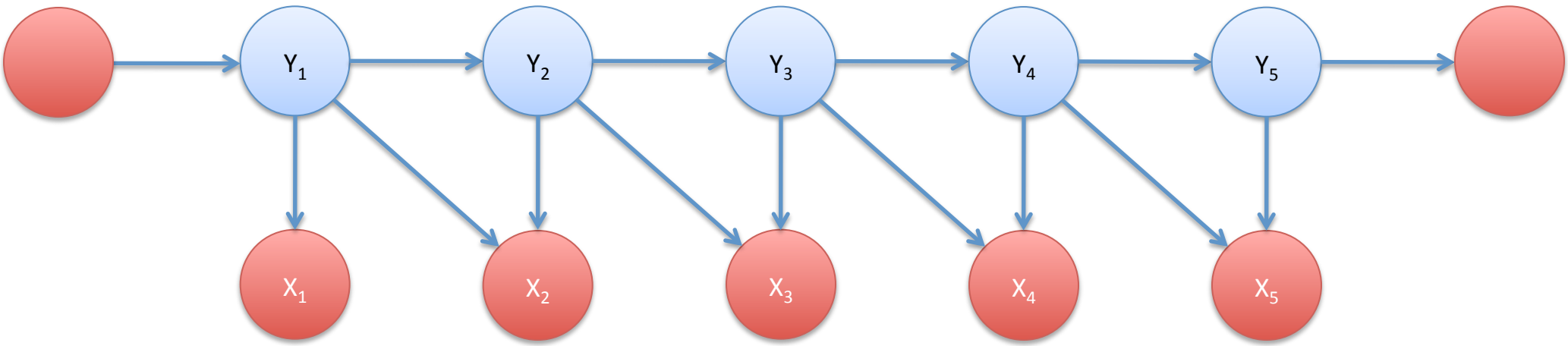
- At the end, we have one final factor with one row, ϕ_{n+1}.
- This is the score of the best sequence.
- Use backtrace to recover values.



Why Think This Way?

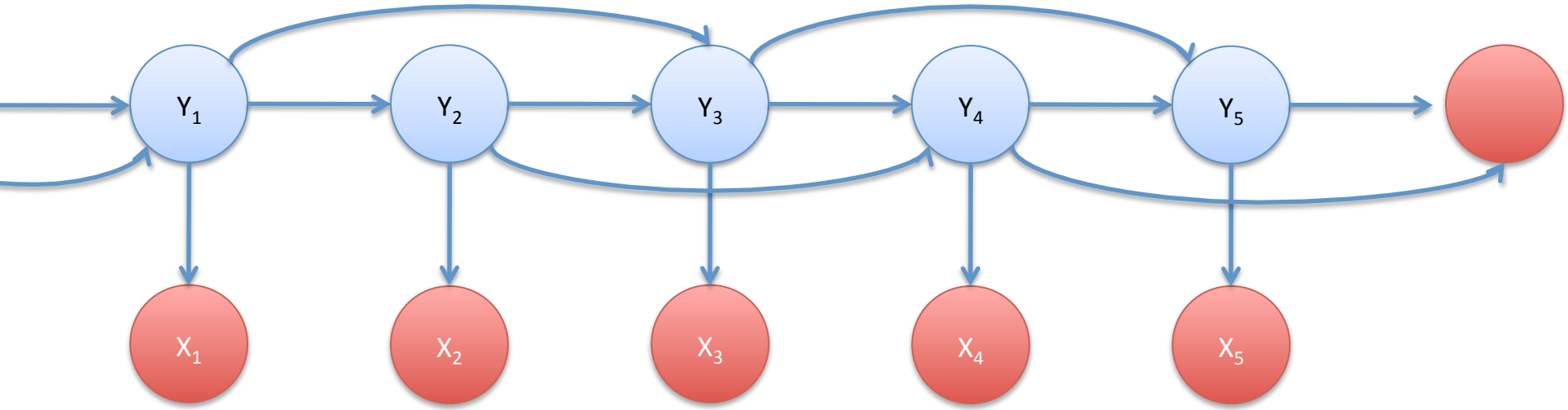
- Easy to see how to generalize HMMs.
  - More evidence
  - More factors
  - More hidden structure
  - More dependencies
- Probabilistic interpretation of factors is *not* central to finding the "best" **Y** …
  - Many factors are not conditional probability tables.

Generalization Example 1



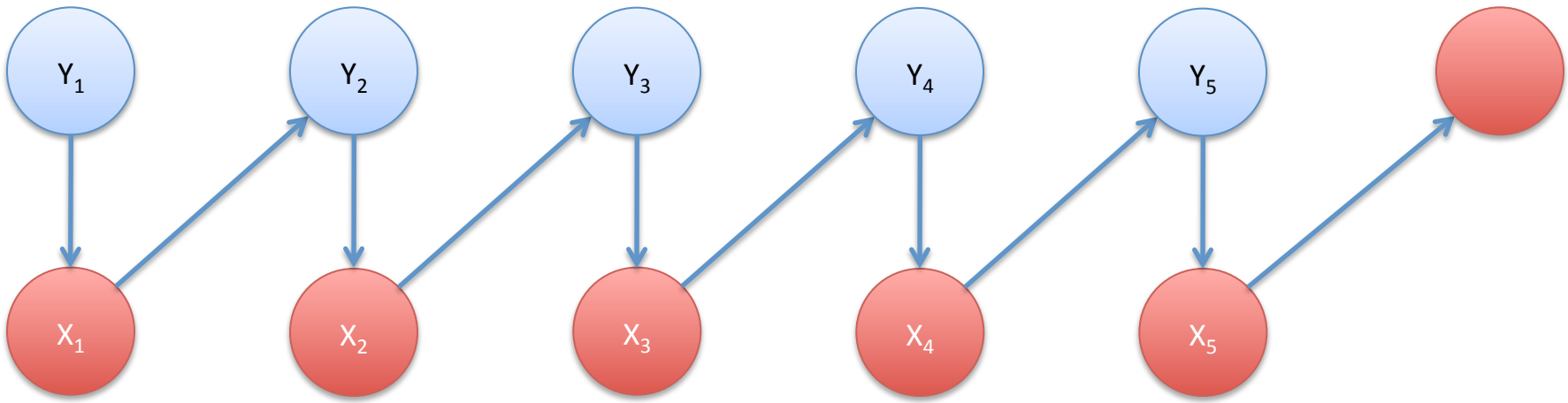
- Each word also depends on previous state.

Generalization Example 2



- "Trigram" HMM

Generalization Example 3



- Aggregate bigram model (Saul and Pereira, 1997)

General Decoding Problem

- Two structured random variables, **X** and **Y**.
  - Sometimes described as *collections* of random variables.
- "Decode" observed value **X** = **x** into some value of **Y**.

- Usually, we seek to maximize some score.
  - Might be probability, might not.

MAP = Optimizing a Linear Score

- Bayesian network:

$$\sum_i \log p(x_i \mid \text{parents}(X_i)) + \sum_j \log p(y_j \mid \text{parents}(Y_j))$$

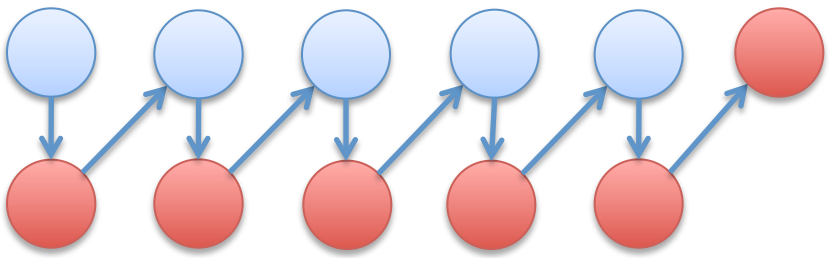
- Markov network:

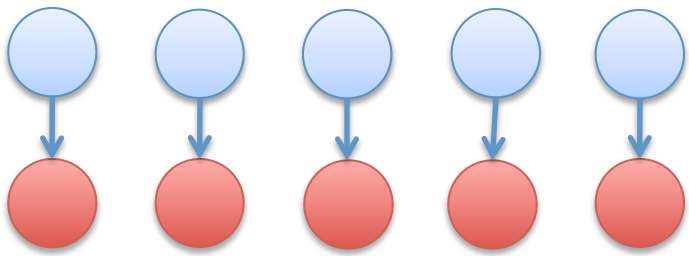
$$\sum_C \log \phi_C \left(\{x_i\}_{i \in C}, \{y_j\}_{j \in C}\right)$$

- This only works if every variable is in **X** or **Y**.

Inference in Graphical Models

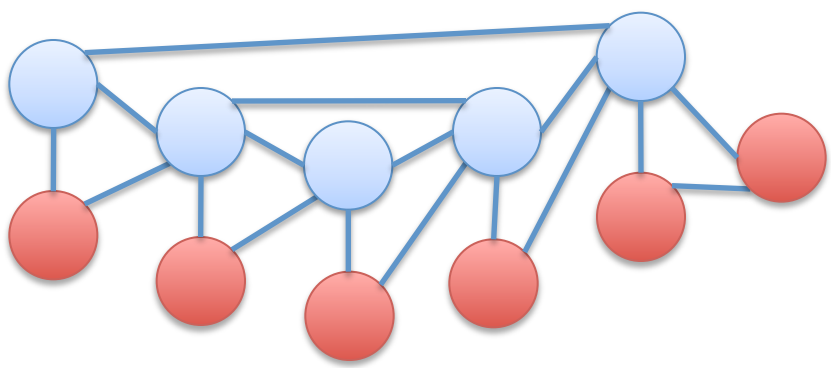
- Remember: more edges make inference more expensive.
  - Fewer edges means stronger independence.
- Really pleasant:

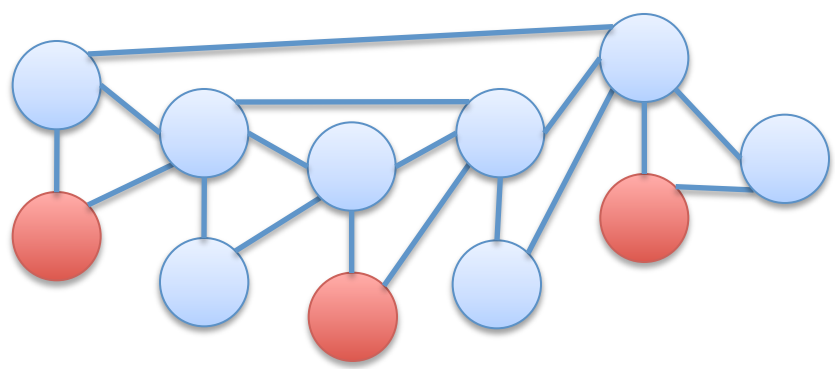




Inference in Graphical Models

- Remember: more edges make inference more expensive.
  - Fewer edges means stronger independence.
- Really unpleasant:





2. Weighted Parsing

Grammars

- Grammars are often associated with natural language parsing, but they are extremely powerful for imposing constraints.
- We can add weights to them.
  - HMMs are a kind of weighted regular grammar (closely connected to WFSAs)
  - PCFGs are a kind of weighted CFG
  - Many, many more.
- Weighted parsing: find the maximum-weighted derivation for a string **x**.

Decoding as Weighted Parsing

- Every valid **y** is a grammatical derivation (parse) for **x**.
  - HMM: sequence of "grammatical" states is one allowed by the transition table.
- Augment parsing algorithms with weights and find the best parse.

The Viterbi algorithm is an instance of recognition by a weighted grammar!

BIO Tagging as a CFG

$$
\begin{array}{lllllllllll}
N_{\Rightarrow} & \rightarrow & B\ R_B & R_B & \rightarrow & B\ R_B & R_I & \rightarrow & B\ R_B & R_O & \rightarrow & B\ R_B \\
N_{\Rightarrow} & \rightarrow & O\ R_O & R_B & \rightarrow & O\ R_O & R_I & \rightarrow & O\ R_O & R_O & \rightarrow & O\ R_O \\
 & & & R_B & \rightarrow & I\ R_I & R_I & \rightarrow & I\ R_I & & & \\
 & & & R_B & \rightarrow & \epsilon & R_I & \rightarrow & \epsilon & R_O & \rightarrow & \epsilon \\
 & & & & & & & & & & & \\
 & \forall x \in \Sigma, & & B & \rightarrow & x & I & \rightarrow & x & O & \rightarrow & x
\end{array}
$$

- Weighted (or probabilistic) CKY is a dynamic programming algorithm very similar in structure to classical CKY.

3. Paths and Hyperpaths

Best Path

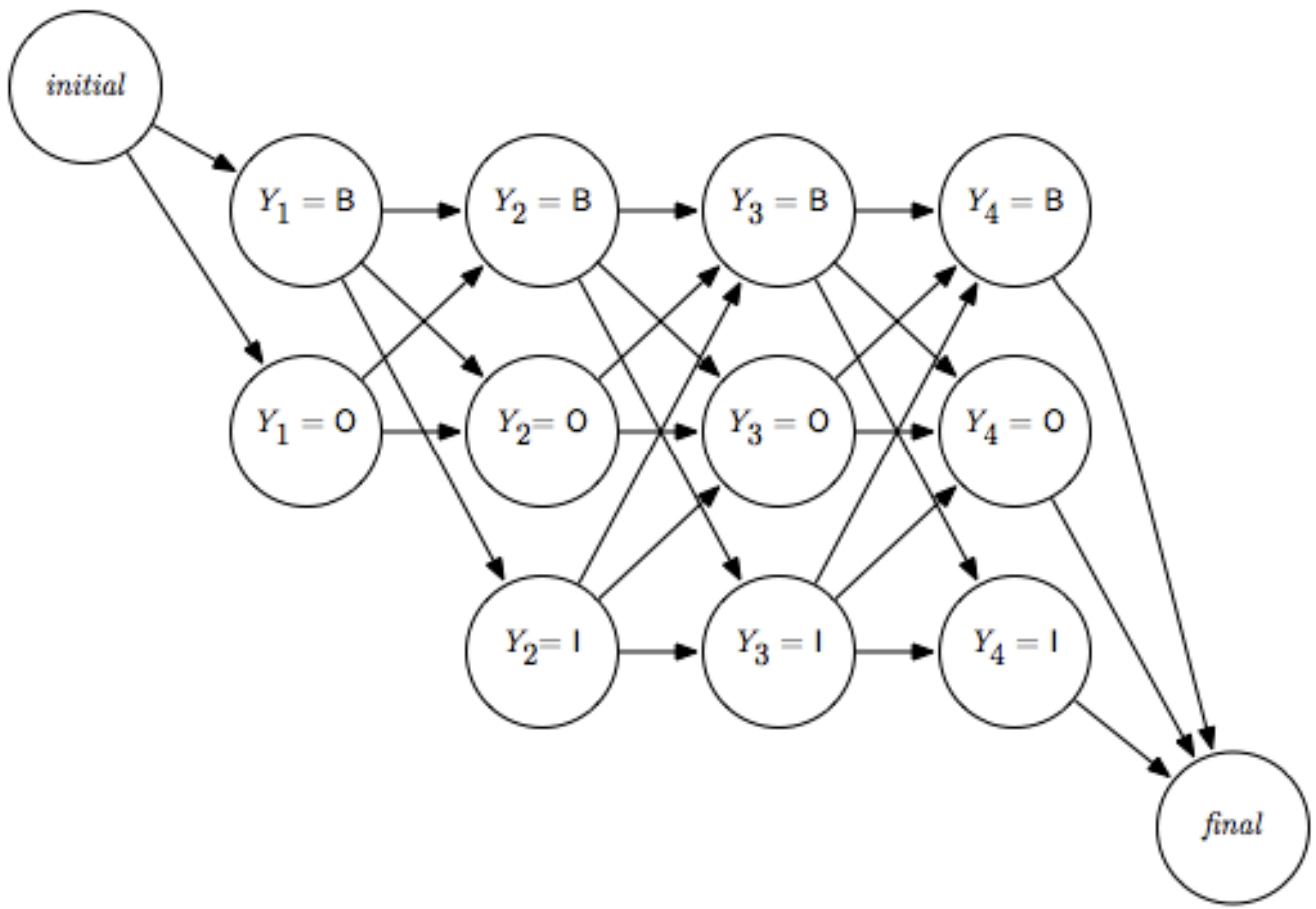
- General idea: take ***x*** and build a graph.
- Score of a path factors into the edges.

$$\arg\max_{\boldsymbol{y}} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) \quad = \quad \arg\max_{\boldsymbol{y}} \mathbf{w}^\top \sum_{e \in \text{Edges}} \mathbf{f}(e) \mathbf{1}\{e \text{ is crossed by } \boldsymbol{y}\text{'s path}\}$$

- Decoding is finding the *best* path.

The Viterbi algorithm is an instance of finding a best path!

“Lattice” View of Viterbi

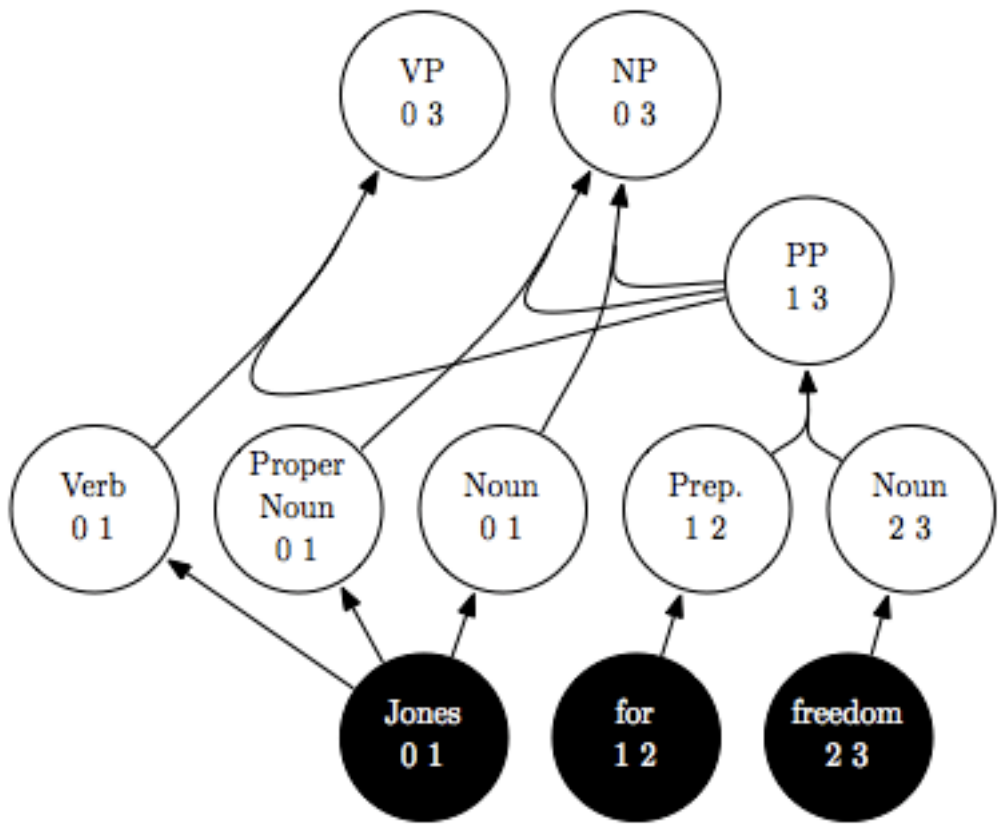


Minimum Cost Hyperpath

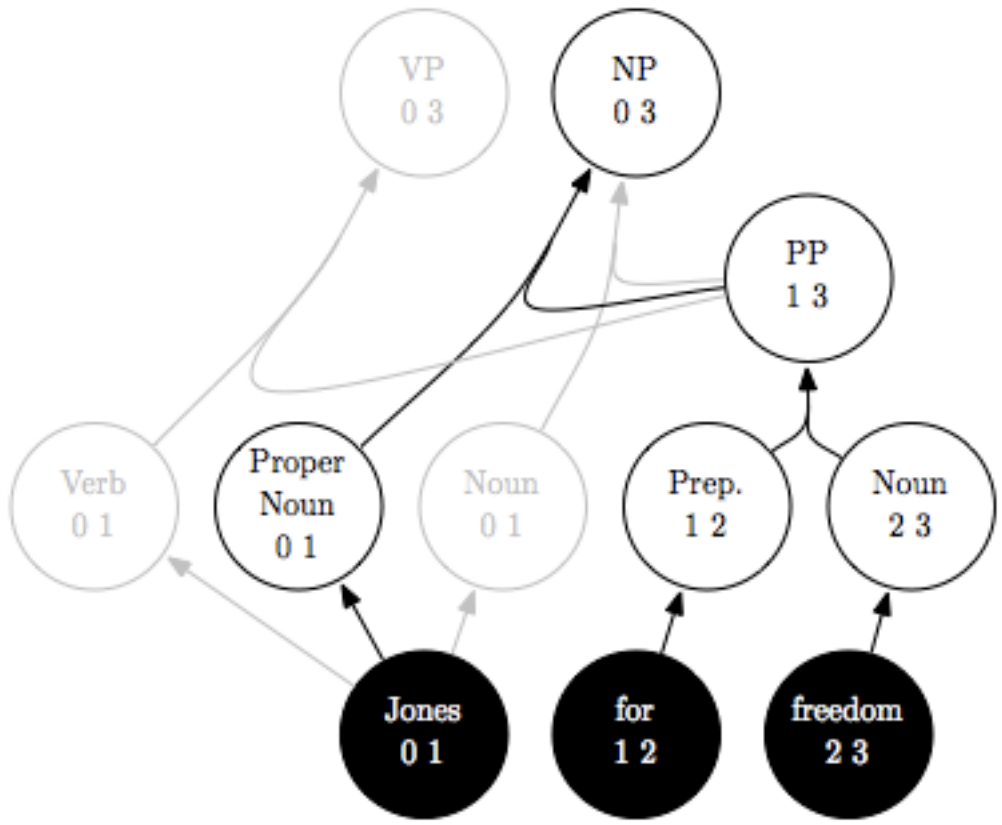
- General idea: take ***x*** and build a hypergraph.
- Score of a hyperpath factors into the hyperedges.
- Decoding is finding the best *hyperpath*.

- This connection was elucidated by Klein and Manning (2002).

Parsing as a Hypergraph

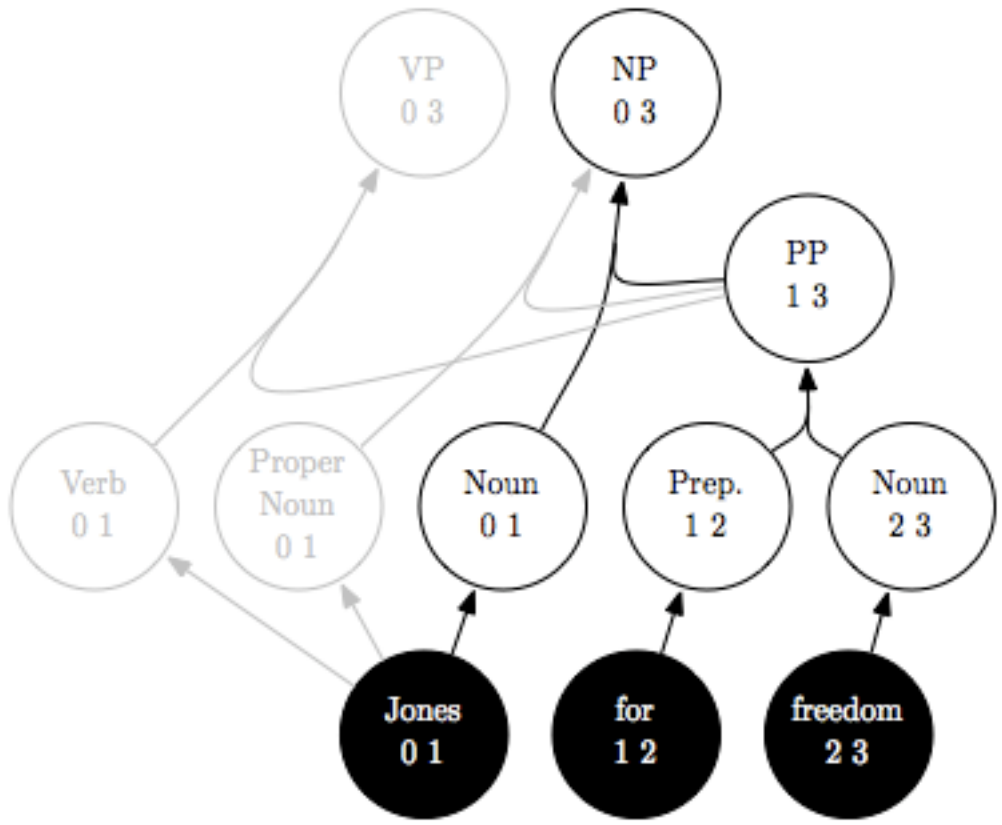


Parsing as a Hypergraph



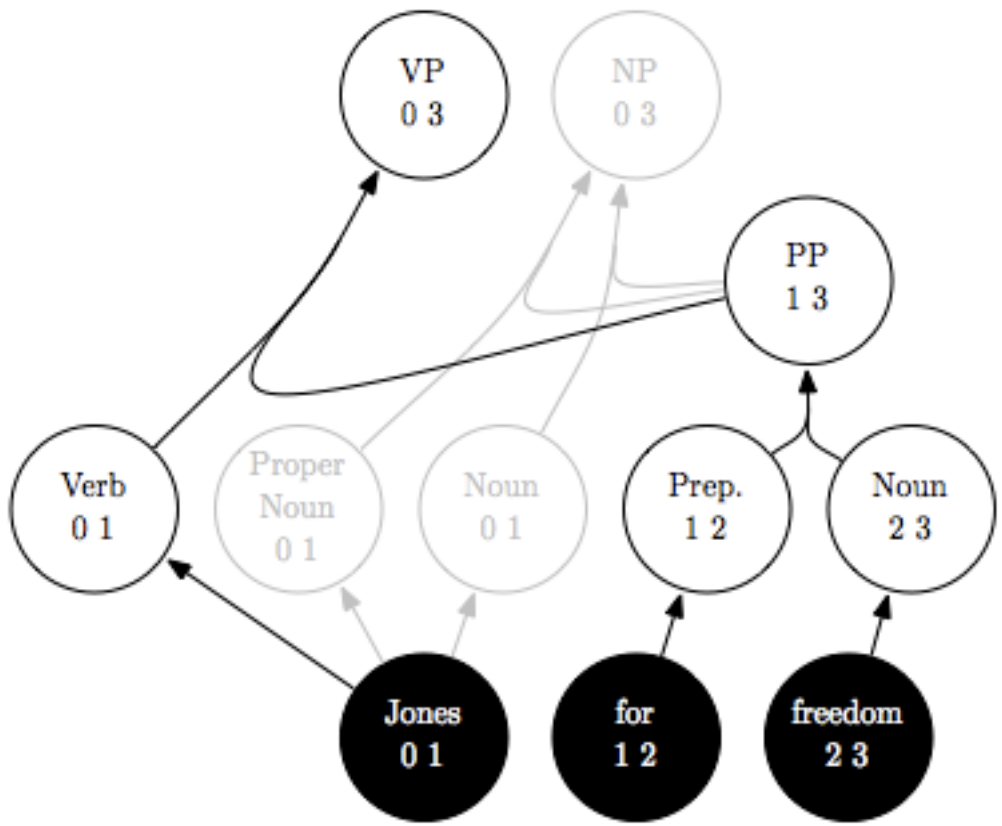
cf. “Dean for democracy”

Parsing as a Hypergraph



Forced to work on his thesis, sunshine streaming in the window, Mike experienced a ...

Parsing as a Hypergraph



Forced to work on his thesis, sunshine streaming in the window, Mike began to ...

Why Hypergraphs?

- Useful, compact encoding of the hypothesis space.
  - Build hypothesis space using local features, maybe do some filtering.
  - Pass it off to another module for more fine-grained scoring with richer or more expensive features.

4. Weighted Logic Programming

Logic Programming

- Start with a set of axioms and a set of inference rules.

$$\forall A, C, \quad \text{ancestor}(A, C) \Leftarrow \text{parent}(A, C)$$
$$\forall A, C, \quad \text{ancestor}(A, C) \Leftarrow \bigvee_B \text{ancestor}(A, B) \wedge \text{parent}(B, C)$$

- The goal is to prove a specific theorem, goal.
- Many approaches, but we assume a *deductive* approach.
  - Start with axioms, iteratively produce more theorems.

$$
\begin{aligned}
& \text{label-bigram}(\text{“B”}, \text{“B”}) \\
& \text{label-bigram}(\text{“B”}, \text{“I”}) \\
& \text{label-bigram}(\text{“B”}, \text{“O”}) \\
& \text{label-bigram}(\text{“I”}, \text{“B”}) \\
& \text{label-bigram}(\text{“I”}, \text{“I”}) \\
& \text{label-bigram}(\text{“I”}, \text{“O”}) \\
& \text{label-bigram}(\text{“O”}, \text{“B”}) \\
& \text{label-bigram}(\text{“O”}, \text{“O”}) \\
\forall x \in \Sigma, \quad & \text{labeled-word}(x, \text{“B”}) \\
\forall x \in \Sigma, \quad & \text{labeled-word}(x, \text{“I”}) \\
\forall x \in \Sigma, \quad & \text{labeled-word}(x, \text{“O”})
\end{aligned}
$$

$$
\begin{aligned}
\forall \ell \in \Lambda, \quad \mathsf{v}(\ell, 1) &= \text{labeled-word}(x_1, \ell) \\
\forall \ell \in \Lambda, \quad \mathsf{v}(\ell, i) &= \bigvee_{\ell' \in \Lambda} \mathsf{v}(\ell', i-1) \wedge \text{label-bigram}(\ell', \ell) \wedge \text{labeled-word}(x_i, \ell) \\
\text{goal} &= \bigvee_{\ell \in \Lambda} \mathsf{v}(\ell, n)
\end{aligned}
$$

Weighted Logic Programming

- Twist: axioms have weights.
- Want the proof of goal with the best score:

$$\arg\max_{\boldsymbol{y}} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) \quad = \quad \arg\max_{\boldsymbol{y}} \mathbf{w}^\top \sum_{a \in \text{Axioms}} \mathbf{f}(a) \mathit{freq}(a; \boldsymbol{y})$$

- Note that axioms can be used more than once in a proof (**y**).

Whence WLP?

- Shieber, Schabes, and Pereira (1995): many parsing algorithms can be understood in the same deductive logic framework.
- Goodman (1999): add weights, get many useful NLP algorithms.
- Eisner, Goldlust, and Smith (2004, 2005): semiring-generic algorithms, Dyna.

Dynamic Programming

- Break a problem into slightly smaller problems with **optimal substructure**.
  - Best path to v depends on best paths to all u such that (u,v) $\in$ E.
- **Overlapping subproblems**: each subproblem gets used repeatedly, and there aren't too many of them.
- Three main strategies for DP:
  - Viterbi, Levenshtein edit distance, CKY: predefined, "clever" ordering.
  - Memoization
  - Agenda (Dijkstra's algorithm, A*)
- Things to remember in general:
  - The hypergraph may too big to represent explicitly; exhaustive calculation may be too expensive.
  - The hypergraph may or may not have properties that make "clever" orderings possible.
  - DP does *not* imply polynomial time and space! Most common approximations when the desired state space is too big: beam search, cube pruning, agendas with early stopping, ...

Summary

- Decoding is the general problem of choosing a complex structure.
  - Linguistic analysis, machine translation, speech recognition, …
  - Statistical models are usually involved (not necessarily probabilistic).
  - Often a subroutine when learning; almost always when evaluating.
- No perfect general view, but much can be gained through a combination of views.