Lecture 3:
Structures and Decoding

Outline

1. Structures in NLP
2. HMMs as BNs
   - Viterbi algorithm as variable elimination
3. Linear models
4. Five views of decoding

Two Meanings of "Structure"

- Yesterday: structure of a graph for modeling a collection of random variables together.
- Today: linguistic structure.
  - Sequence labelings (POS, IOB chunkings, …)
  - Parse trees (phrase-structure, dependency, …)
  - Alignments (word, phrase, tree, …)
  - Predicate-argument structures
  - Text-to-text (translation, paraphrase, answers, …)

A Useful Abstraction?

- We think so.
- Brings out commonalities:
  - Modeling formalisms (e.g., linear models with features)
  - Learning algorithms (lectures 4-6)
  - Generic inference algorithms
- Permits sharing across a wider space of problems.
- Disadvantage: hides engineering details.

Familiar Example:
Hidden Markov Models

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

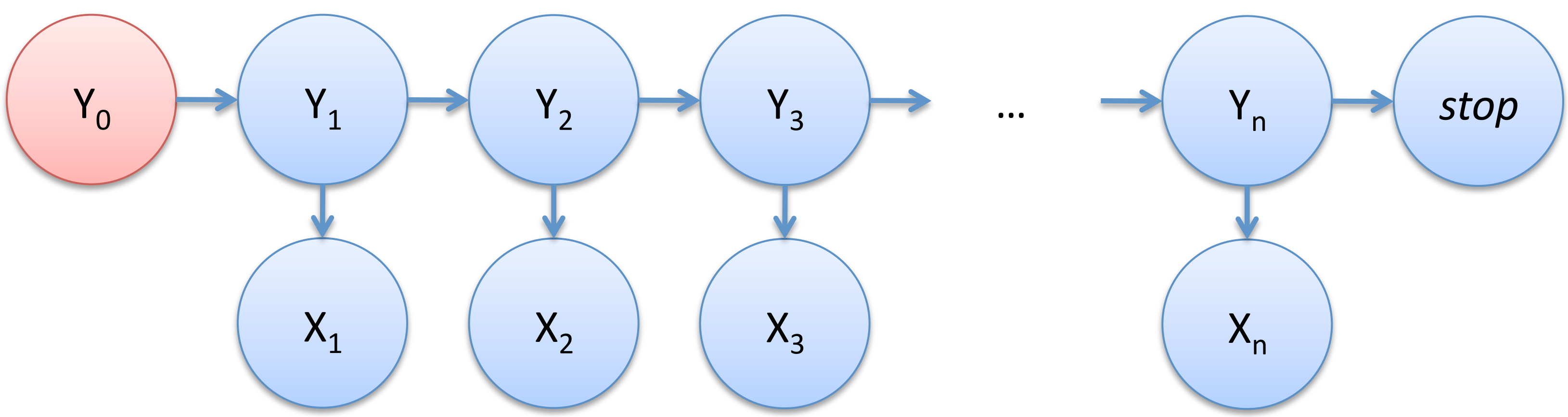
$$p(\boldsymbol{X} = \boldsymbol{x}, \boldsymbol{Y} = \boldsymbol{y}) \quad = \quad \left(\prod_{i=1}^{n} p(x_i \mid y_i) p(y_i \mid y_{i-1}) \right) p(stop \mid y_n)$$

- 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.

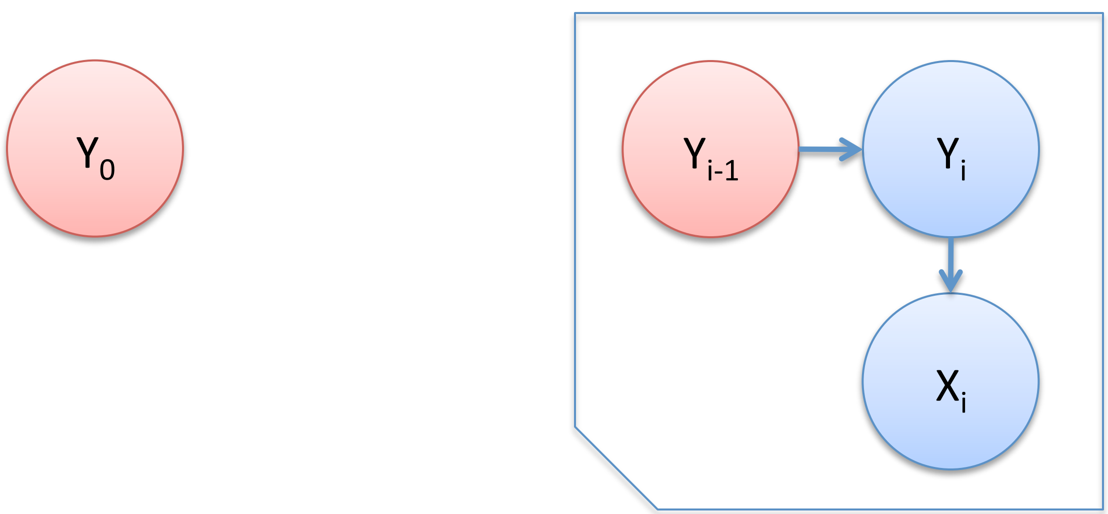
$$p(\boldsymbol{X} = \boldsymbol{x}, \boldsymbol{Y} = \boldsymbol{y}) = \left(\prod_{i=1}^{n} p(x_i \mid y_i) p(y_i \mid y_{i-1}) \right) p(stop \mid y_n)$$



Hidden Markov Model

- The joint model instantiates **dynamic Bayesian networks**.

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

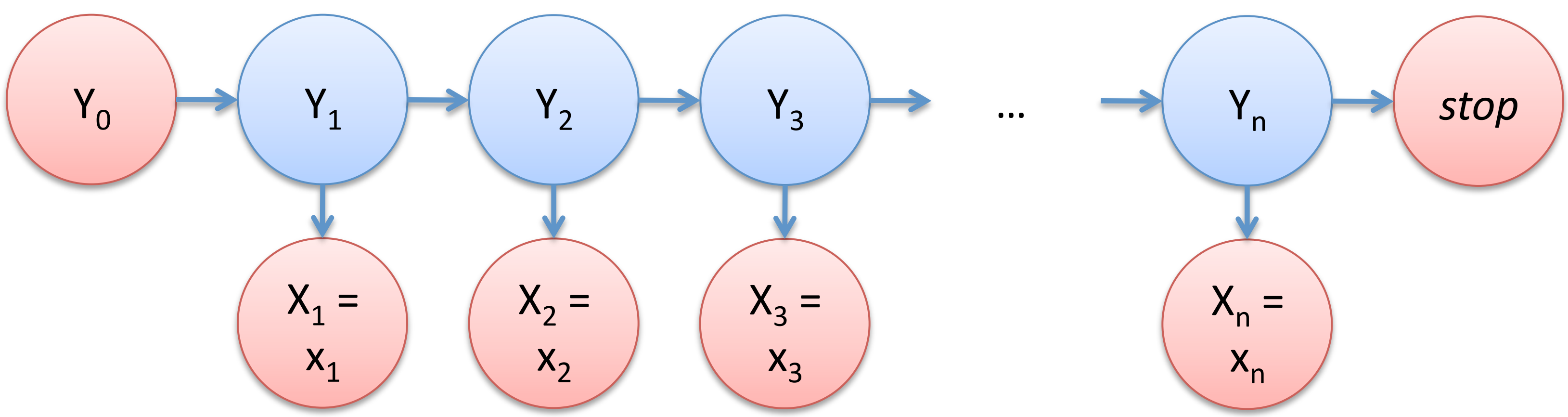


template that gets copied as many times as needed

Hidden Markov Model

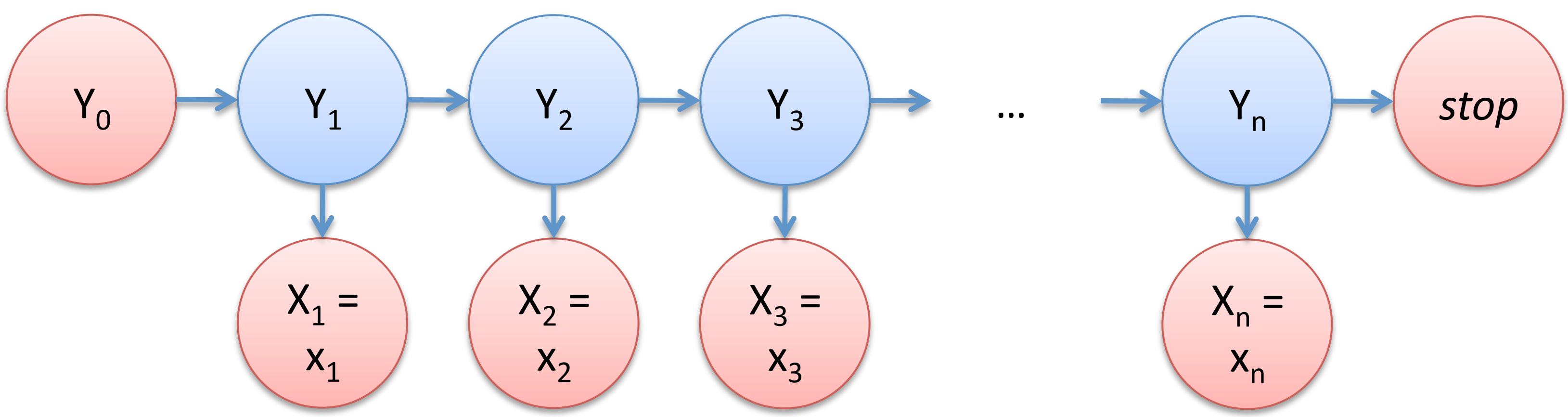
- Given **X**'s value as evidence, the dynamic part becomes unnecessary, since we know n.

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



Hidden Markov Model

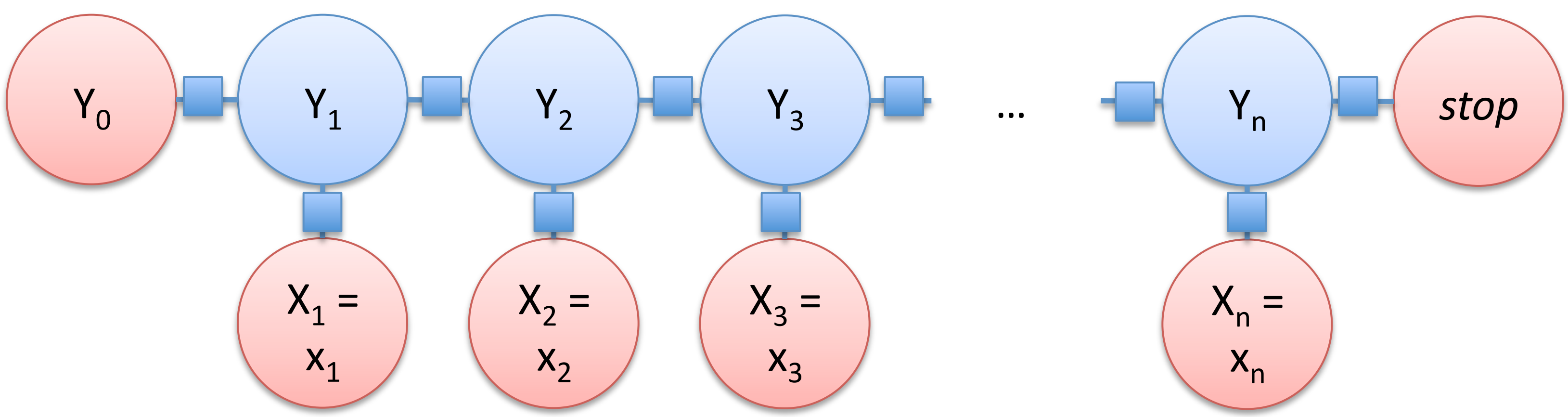
- The usual inference problem is to find the most probable value of **Y** given **X** = **x**.



Hidden Markov Model

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

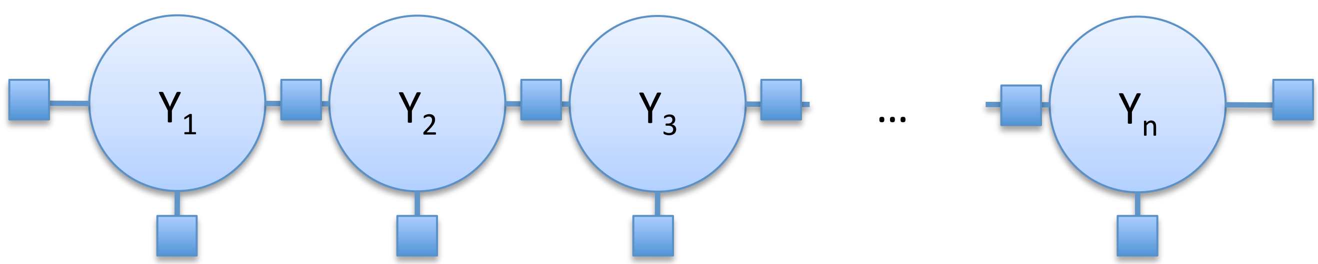
- Factor graph:



Hidden Markov Model

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

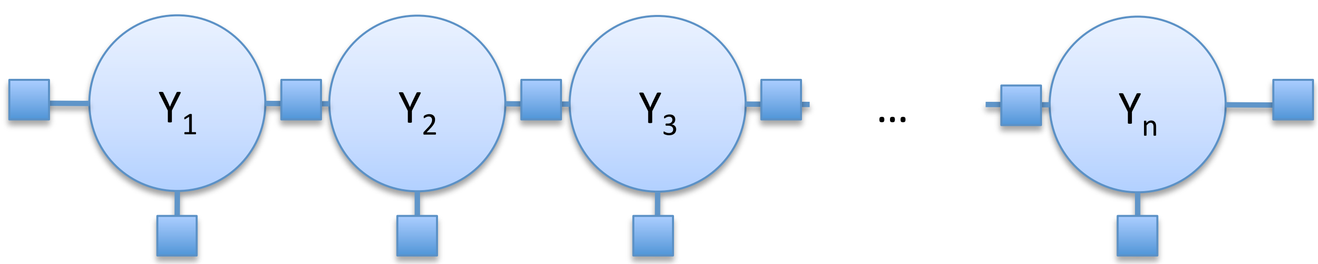
- Factor graph after reducing factors to respect evidence:



Hidden Markov Model

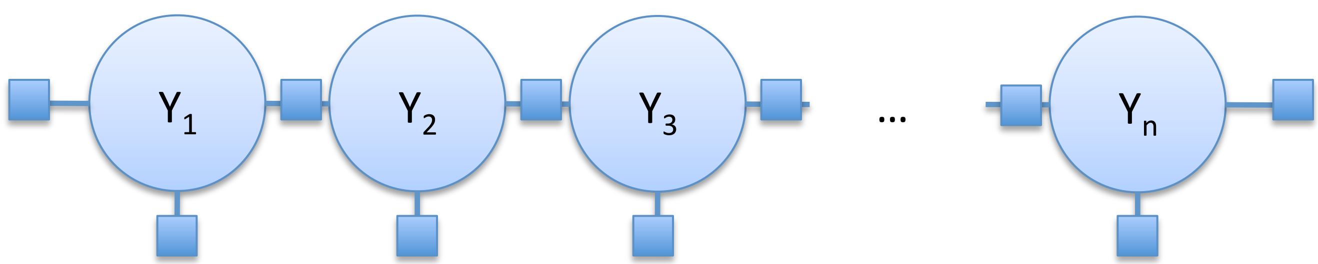
- The usual inference problem is to find the most probable value of **Y** given **X** = **x**.

- Clever ordering should be apparent!



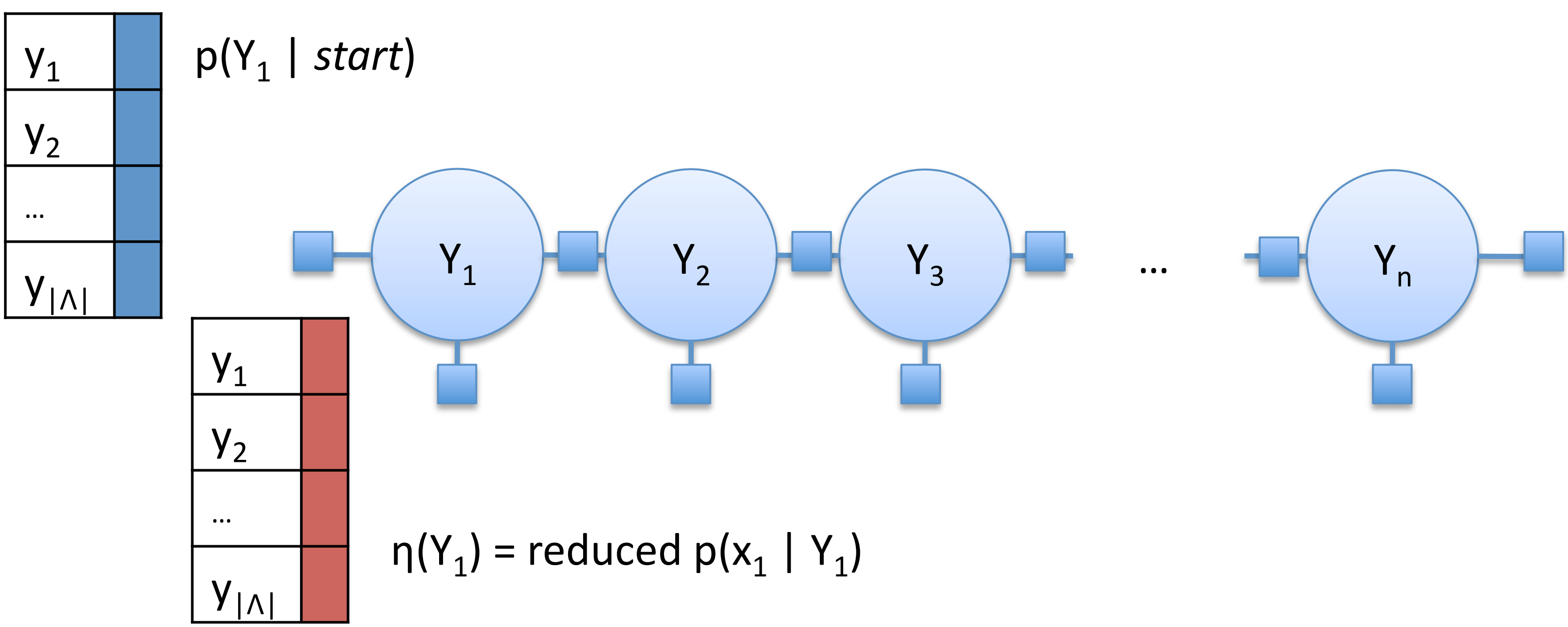
Hidden Markov Model

- 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.

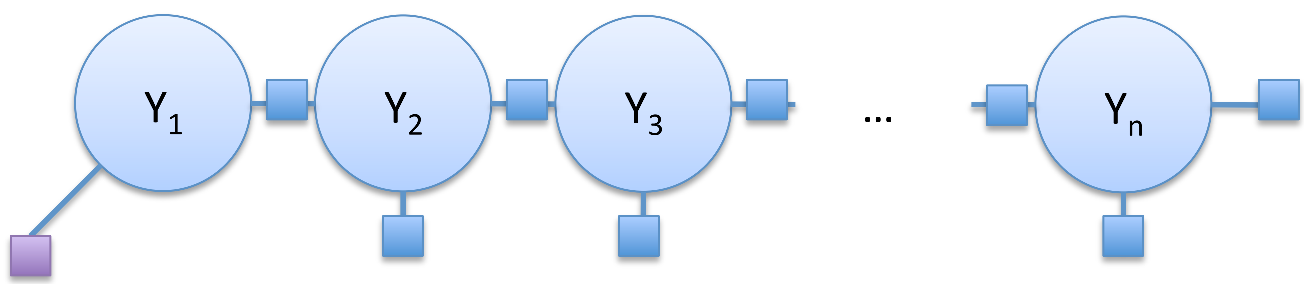


Hidden Markov Model

- 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.

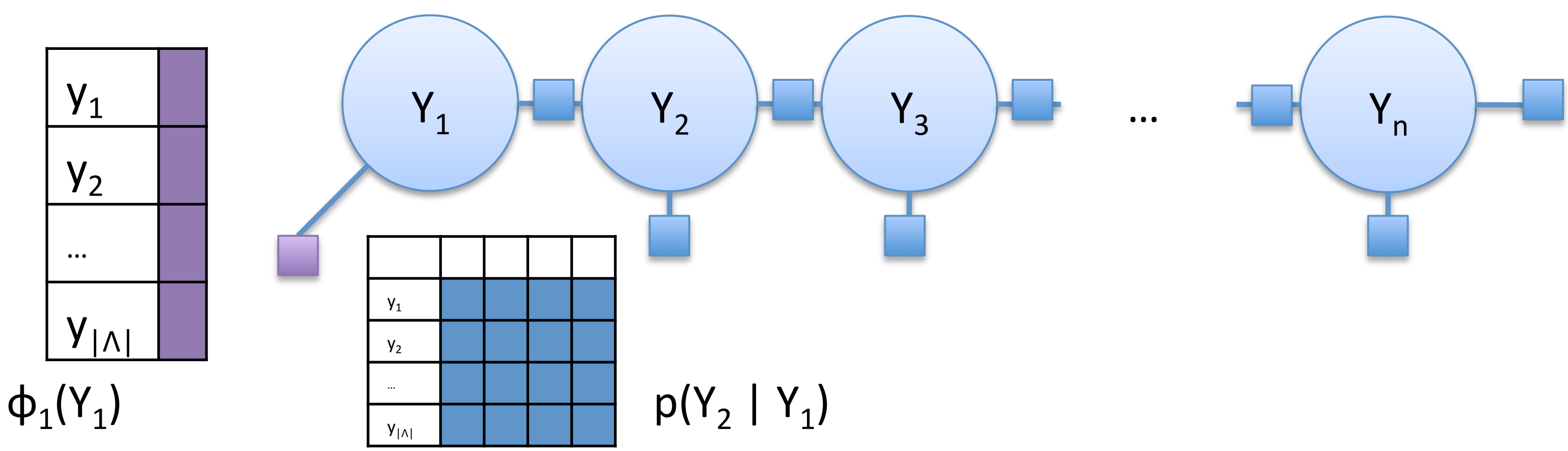
y_1	
y_2	
…	
$y_{\|\Lambda\|}$	

$\phi_1(Y_1)$



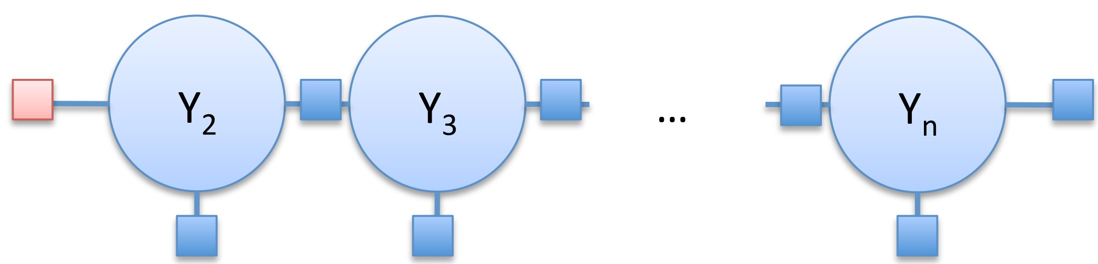
Hidden Markov Model

- 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.



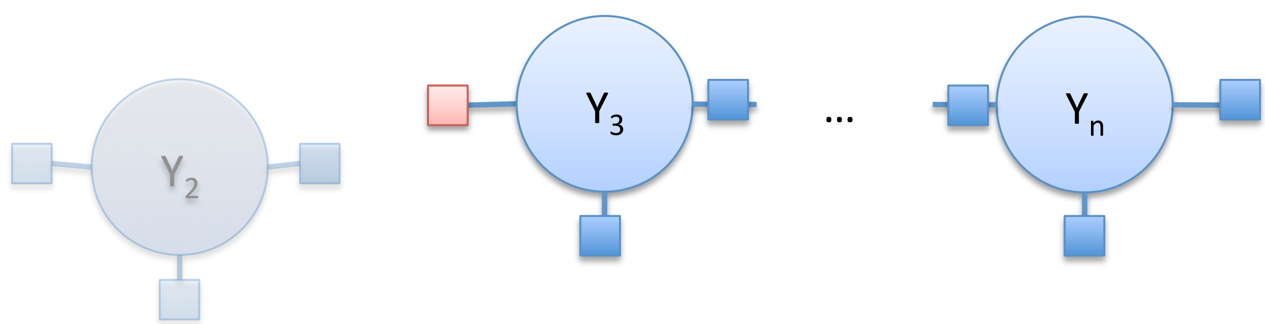
Hidden Markov Model

- 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 probabiliiesy for Y_2.



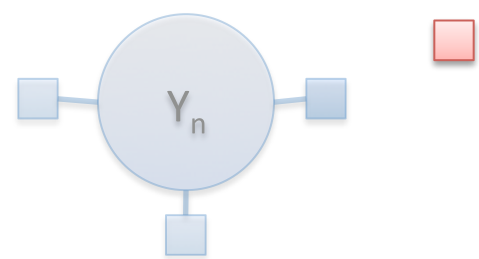
Hidden Markov Model

- 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

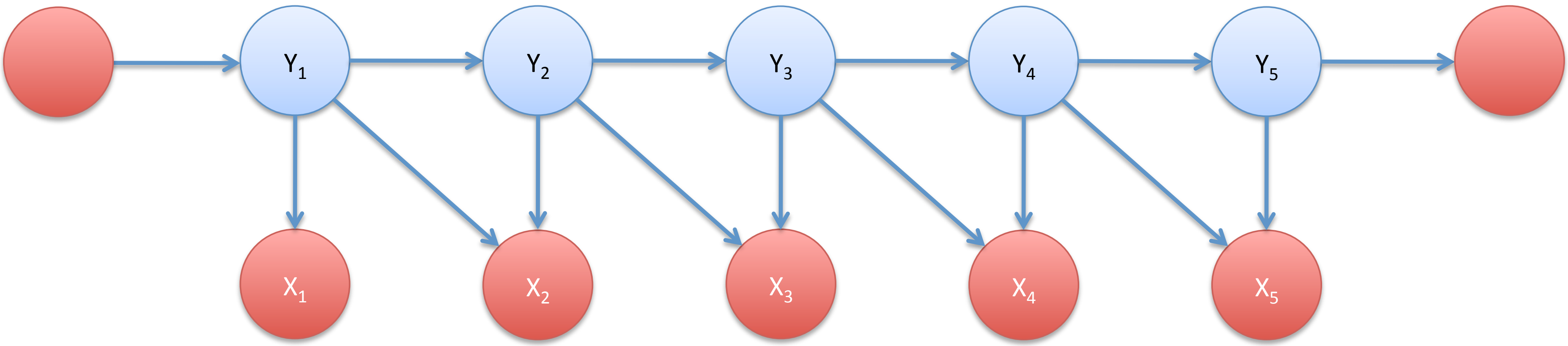
- At the end, we have one final factor with one row, ϕ_{n+1}.
- This is the score of the best sequence.
- Use backtrace to recover values.



Why Think This Way?

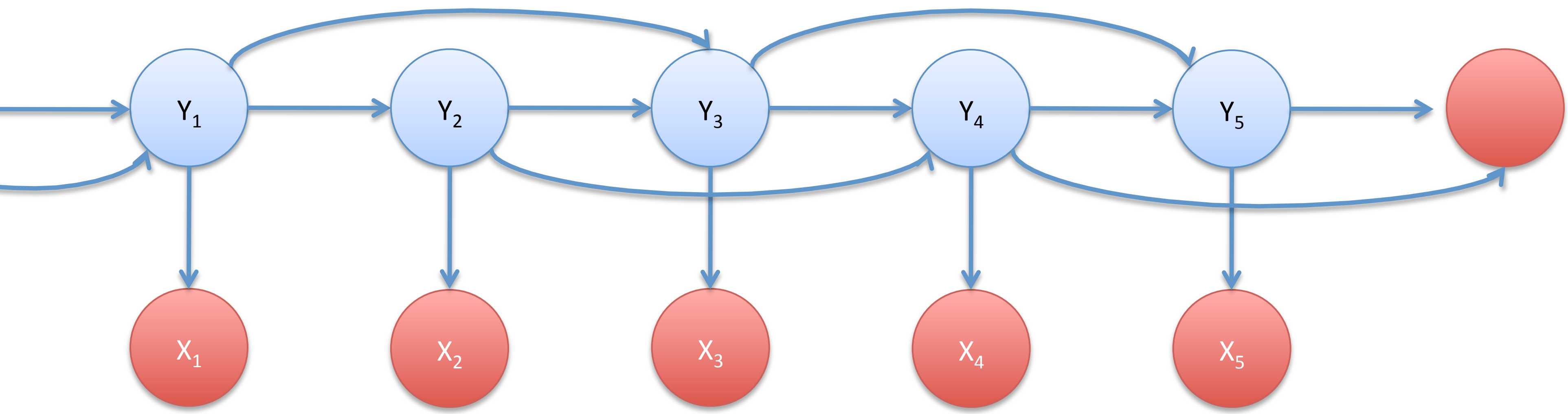
- Easy to see how to generalize HMMs.
  - More evidence
  - More factors
  - More hidden structure
  - More dependencies
- Probabilistic interpretation of factors is *not* central to finding the "best" **Y** …
  - Many factors are not conditional probability tables.

Generalization Example 1



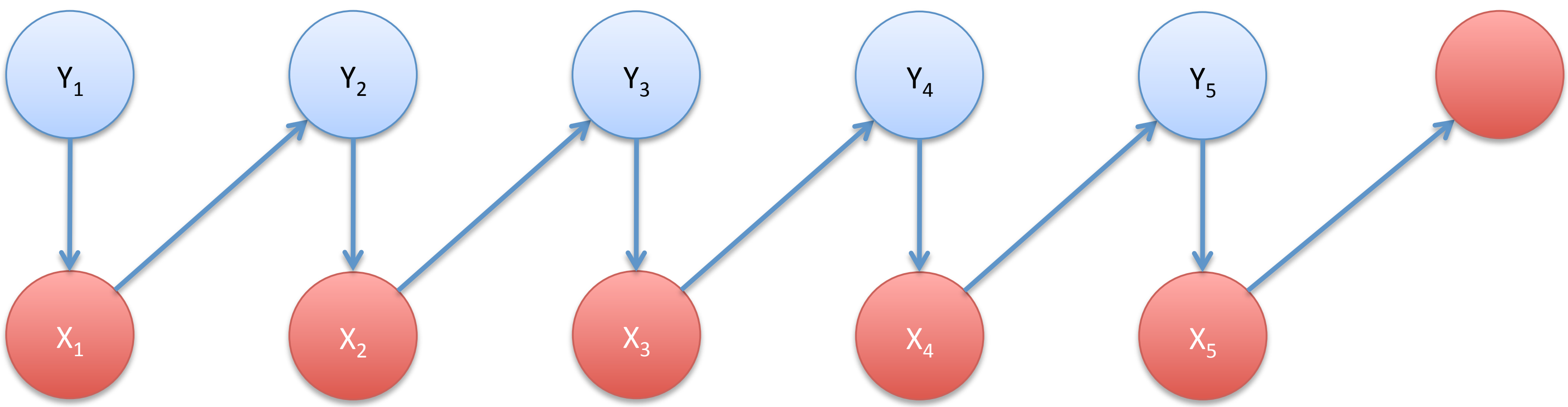
- Each word also depends on previous state.

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.
  - E.g., MAP inference from yesterday.

Linear Models

- Define a feature vector function **g** that maps (**x**, **y**) pairs into d-dimensional real space.
- Score is linear in **g**(**x**, **y**).

$$
\begin{aligned}
score(\boldsymbol{x}, \boldsymbol{y}) &= \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) \\
\boldsymbol{y}^* &= \arg\max_{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}}} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y})
\end{aligned}
$$

- Results:
  – decoding seeks **y** to maximize the score.
  – learning seeks **w** to … do something we'll talk about later.
- Extremely general!

Generic Noisy Channel as Linear Model

$$\begin{aligned}
\hat{\boldsymbol{y}} &= \arg\max_{\boldsymbol{y}} \log\left(p(\boldsymbol{y}) \cdot p(\boldsymbol{x} \mid \boldsymbol{y})\right) \\
&= \arg\max_{\boldsymbol{y}} \log p(\boldsymbol{y}) + \log p(\boldsymbol{x} \mid \boldsymbol{y}) \\
&= \arg\max_{\boldsymbol{y}} w_{\boldsymbol{y}} + w_{\boldsymbol{x}|\boldsymbol{y}} \\
&= \arg\max_{\boldsymbol{y}} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y})
\end{aligned}$$

- Of course, the two probability terms are typically composed of "smaller" factors; each can be understood as an exponentiated weight.

Max Ent Models as Linear Models

$$
\begin{aligned}
\hat{\boldsymbol{y}} &= \arg\max_{\boldsymbol{y}} \log p(\boldsymbol{y} \mid \boldsymbol{x}) \\
&= \arg\max_{\boldsymbol{y}} \log \frac{\exp \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y})}{z(\boldsymbol{x})} \\
&= \arg\max_{\boldsymbol{y}} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) - \log z(\boldsymbol{x}) \\
&= \arg\max_{\boldsymbol{y}} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y})
\end{aligned}
$$

HMMs as Linear Models

$$
\begin{aligned}
\hat{\boldsymbol{y}} &= \arg\max_{\boldsymbol{y}} \log p(\boldsymbol{x}, \boldsymbol{y}) \\
&= \arg\max_{\boldsymbol{y}} \left(\sum_{i=1}^{n} \log p(x_i \mid y_i) + \log p(y_i \mid y_{i-1}) \right) + \log p(stop \mid y_n) \\
&= \arg\max_{\boldsymbol{y}} \left(\sum_{i=1}^{n} w_{y_i \downarrow x_i} + w_{y_{i-1} \rightarrow y_i} \right) + w_{y_n \rightarrow stop} \\
&= \arg\max_{\boldsymbol{y}} \sum_{y,x} w_{y \downarrow x} freq(y \downarrow x; \boldsymbol{y}, \boldsymbol{x}) + \sum_{y,y'} w_{y \rightarrow y'} freq(y \rightarrow y'; \boldsymbol{y}) \\
&= \arg\max_{\boldsymbol{y}} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y})
\end{aligned}
$$

Running Example

		1	2	3	4	5	6	7	8	9	10
$\boldsymbol{x}$	=	Britain	sent	warships	across	the	English	Channel	Monday	to	rescue
$\boldsymbol{y}$	=	B	O	O	O	O	B	I	B	O	O
$\boldsymbol{y}'$	=	O	O	O	O	O	B	I	B	O	O

11	12	13	14	15	16	17	18	19	20
Britons	stranded	by	Eyjafjallajökull	's	volcanic	ash	cloud	.	
B	O	O	B	O	O	O	O	O	O
B	O	O	B	O	O	O	O	O	O

- IOB sequence labeling, here applied to NER
- Often solved with HMMs, CRFs, M^3Ns …

feature function $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$		$g(\boldsymbol{x}, \boldsymbol{y})$	$g(\boldsymbol{x}, \boldsymbol{y}')$
bias:	count of i s.t. $y_i = \mathsf{B}$	5	4
	count of i s.t. $y_i = \mathsf{I}$	1	1
	count of i s.t. $y_i = \mathsf{O}$	14	15
lexical:	count of i s.t. $x_i = \textit{Britain}$ and $y_i = \mathsf{B}$	1	0
	count of i s.t. $x_i = \textit{Britain}$ and $y_i = \mathsf{I}$	0	0
	count of i s.t. $x_i = \textit{Britain}$ and $y_i = \mathsf{O}$	0	1
downcased:	count of i s.t. $lc(x_i) = \textit{britain}$ and $y_i = \mathsf{B}$	1	0
	count of i s.t. $lc(x_i) = \textit{britain}$ and $y_i = \mathsf{I}$	0	0
	count of i s.t. $lc(x_i) = \textit{britain}$ and $y_i = \mathsf{O}$	0	1
	count of i s.t. $lc(x_i) = \textit{sent}$ and $y_i = \mathsf{O}$	1	1
	count of i s.t. $lc(x_i) = \textit{warships}$ and $y_i = \mathsf{O}$	1	1
shape:	count of i s.t. $shape(x_i) = \textit{Aaaaaaa}$ and $y_i = \mathsf{B}$	3	2
	count of i s.t. $shape(x_i) = \textit{Aaaaaaa}$ and $y_i = \mathsf{I}$	1	1
	count of i s.t. $shape(x_i) = \textit{Aaaaaaa}$ and $y_i = \mathsf{O}$	0	1
prefix:	count of i s.t. $pre_1(x_i) = B$ and $y_i = \mathsf{B}$	2	1
	count of i s.t. $pre_1(x_i) = B$ and $y_i = \mathsf{I}$	0	0
	count of i s.t. $pre_1(x_i) = B$ and $y_i = \mathsf{O}$	0	1
	count of i s.t. $pre_1(x_i) = s$ and $y_i = \mathsf{O}$	2	2
	count of i s.t. $shape(pre_1(x_i)) = A$ and $y_i = \mathsf{B}$	5	4
	count of i s.t. $shape(pre_1(x_i)) = A$ and $y_i = \mathsf{I}$	1	1
	count of i s.t. $shape(pre_1(x_i)) = A$ and $y_i = \mathsf{O}$	0	1
	$[\![shape(pre_1(x_1)) = A \wedge y_1 = \mathsf{B}]\!]$	1	0
	$[\![shape(pre_1(x_1)) = A \wedge y_1 = \mathsf{O}]\!]$	0	1
gazetteer:	count of i s.t. x_i is in the gazetteer and $y_i = \mathsf{B}$	2	1
	count of i s.t. x_i is in the gazetteer and $y_i = \mathsf{I}$	0	0
	count of i s.t. x_i is in the gazetteer and $y_i = \mathsf{O}$	0	1
	count of i s.t. $x_i = \textit{sent}$ and $y_i = \mathsf{O}$	1	1

(What is *Not* A Linear Model?)

- Models with hidden variables

$$\arg\max_{\boldsymbol{y}} p(\boldsymbol{y} \mid \boldsymbol{x}) = \arg\max_{\boldsymbol{y}} \sum_{\boldsymbol{z}} p(\boldsymbol{y}, \boldsymbol{z} \mid \boldsymbol{x})$$

- Models based on non-linear kernels

$$\arg\max_{\boldsymbol{y}} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) \quad = \quad \arg\max_{\boldsymbol{y}} \sum_{i=1}^{N} \alpha_i K\left(\langle \boldsymbol{x}_i, \boldsymbol{y}_i \rangle, \langle \boldsymbol{x}, \boldsymbol{y} \rangle\right)$$

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

Five 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!

MAP is Linear Decoding

- Bayesian network:

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

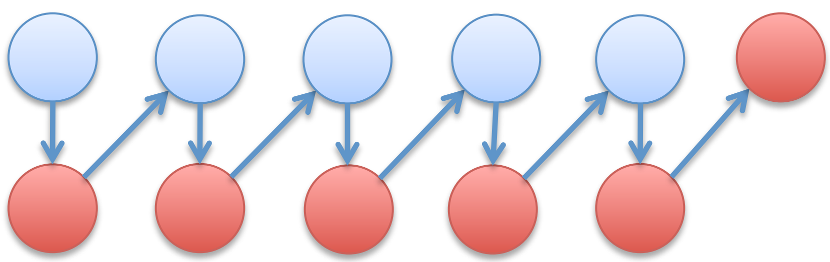
- Markov network:

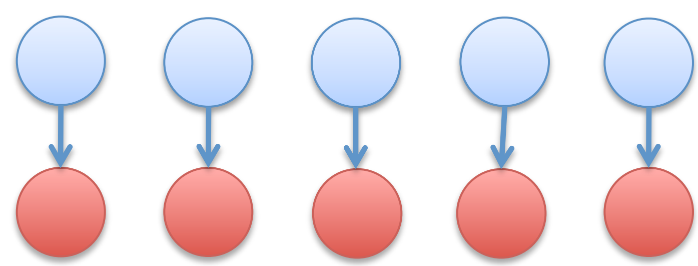
$$\sum_C \log \phi_C\left(\{x_i\}_{i \in C}, \{y_j\}_{j \in C}\right)$$

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

Inference in Graphical Models

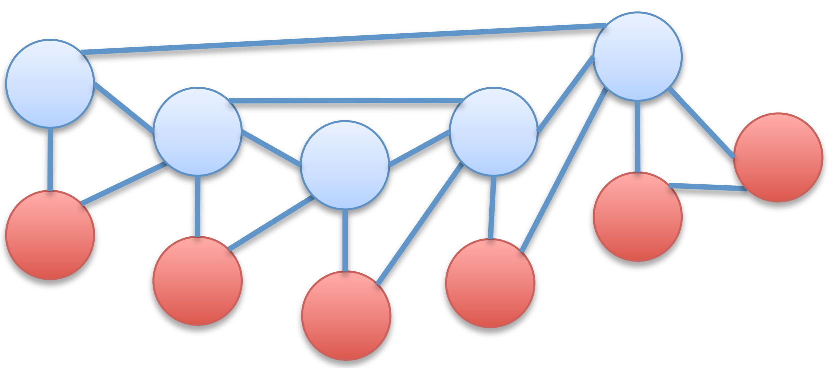
- Remember: more edges make inference more expensive.
  - Fewer edges means stronger independence.
- Really pleasant:

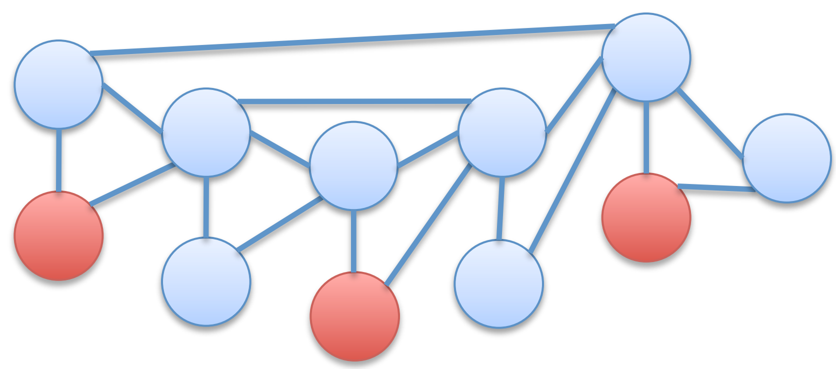




Inference in Graphical Models

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





2. Polytopes

“Parts”

- Assume that feature function **g** breaks down into local parts.

$$\mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) \quad = \quad \sum_{i=1}^{\#parts(\boldsymbol{x})} \mathbf{f}(\Pi_i(\boldsymbol{x}, \boldsymbol{y}))$$

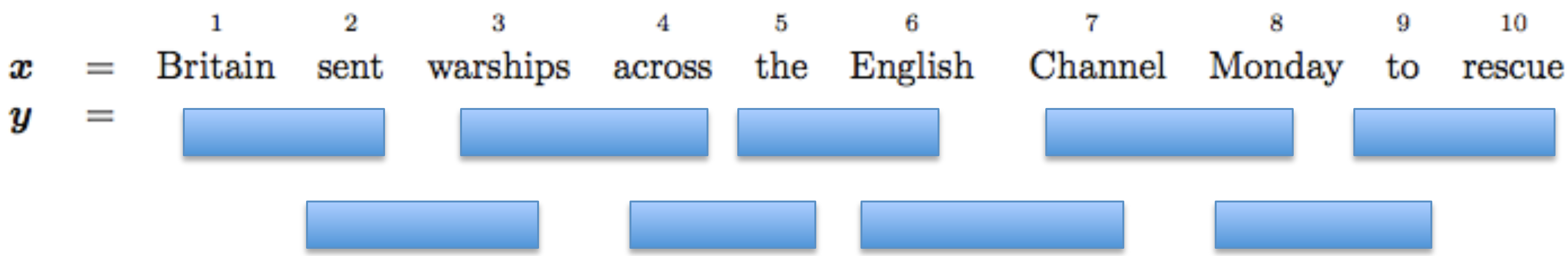
- Each part has an alphabet of possible values.
  - Decoding is choosing values for all parts, with consistency constraints.
  - (In the graphical models view, a part is a clique.)

Example

		1	2	3	4	5	6	7	8	9	10
$\boldsymbol{x}$	=	Britain	sent	warships	across	the	English	Channel	Monday	to	rescue
$\boldsymbol{y}$	=										

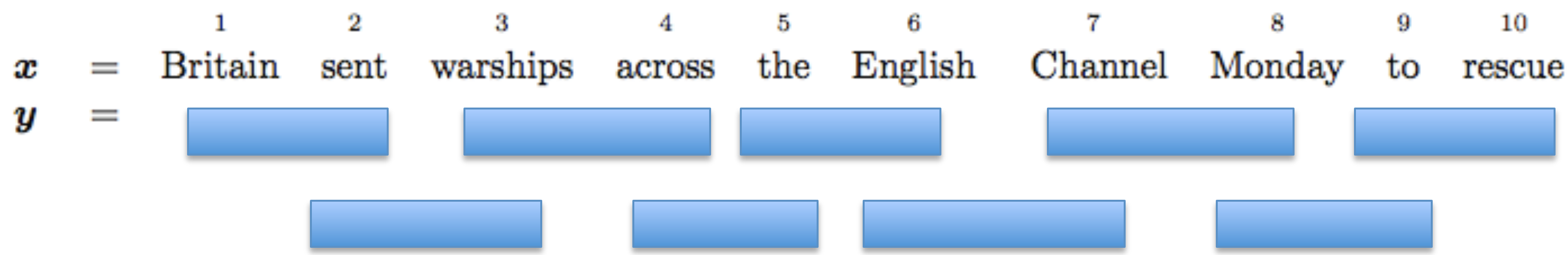
- One part per word, each is in {B, I, O}
- No features look at multiple parts
  - Fast inference
  - Not very expressive

Example



- One part per bigram, each is in {BB, BI, BO, IB, II, IO, OB, OO}
- Features and constraints can look at pairs
  - Slower inference
  - A bit more expressive

Geometric View



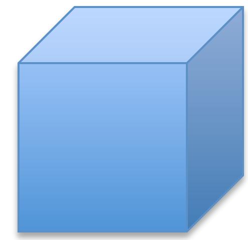
- Let $z_{i,\boldsymbol{\pi}}$ be 1 if part i takes value $\boldsymbol{\pi}$ and 0 otherwise.
- **z** is a vector in $\{0, 1\}^N$
  – N = total number of localized part values
  – Each **z** is a vertex of the unit cube

Score is Linear in **z**

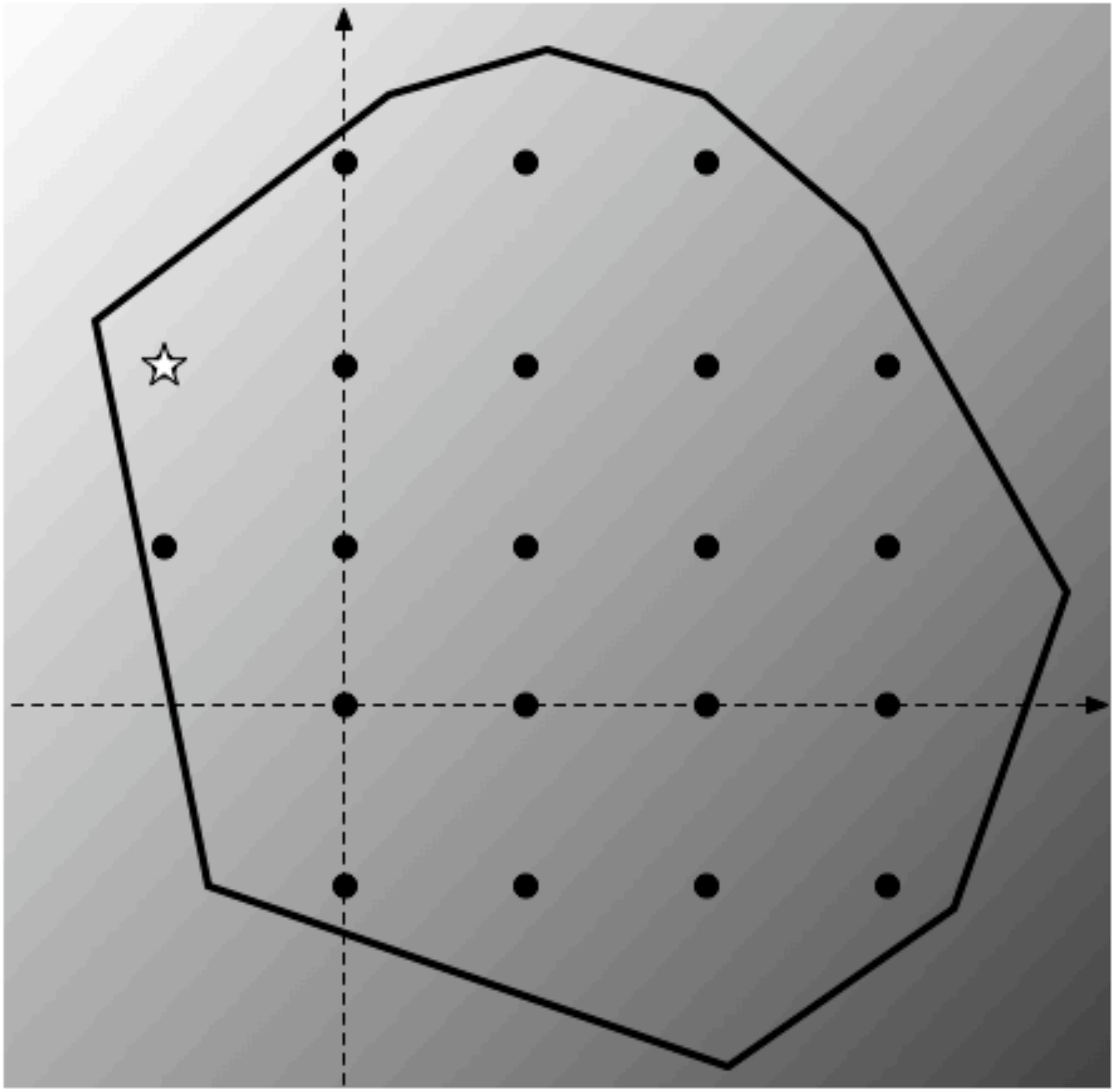
$$
\begin{aligned}
\arg\max_{\boldsymbol{y}} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) &= \arg\max_{\boldsymbol{y}} \mathbf{w}^\top \sum_{i=1}^{\#parts(\boldsymbol{x})} \mathbf{f}(\Pi_i(\boldsymbol{x}, \boldsymbol{y})) \\
&= \arg\max_{\boldsymbol{y}} \mathbf{w}^\top \sum_{i=1}^{\#parts(\boldsymbol{x})} \sum_{\boldsymbol{\pi} \in \mathrm{Values}(\Pi_i)} \mathbf{f}(\boldsymbol{\pi}) \mathbf{1}\{\Pi_i(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\pi}\} \\
&= \arg\max_{\mathbf{z} \in \mathcal{Z}_{\boldsymbol{x}}} \mathbf{w}^\top \sum_{i=1}^{\#parts(\boldsymbol{x})} \sum_{\boldsymbol{\pi} \in \mathrm{Values}(\Pi_i)} \mathbf{f}(\boldsymbol{\pi}) z_{i,\boldsymbol{\pi}} \\
&= \arg\max_{\mathbf{z} \in \mathcal{Z}_{\boldsymbol{x}}} \mathbf{w}^\top \mathbf{F}_{\boldsymbol{x}} \mathbf{z} \\
&= \arg\max_{\mathbf{z} \in \mathcal{Z}_{\boldsymbol{x}}} \left(\mathbf{w}^\top \mathbf{F}_{\boldsymbol{x}}\right) \mathbf{z}
\end{aligned}
$$

not really equal; need to transform back to get **y**

Polyhedra

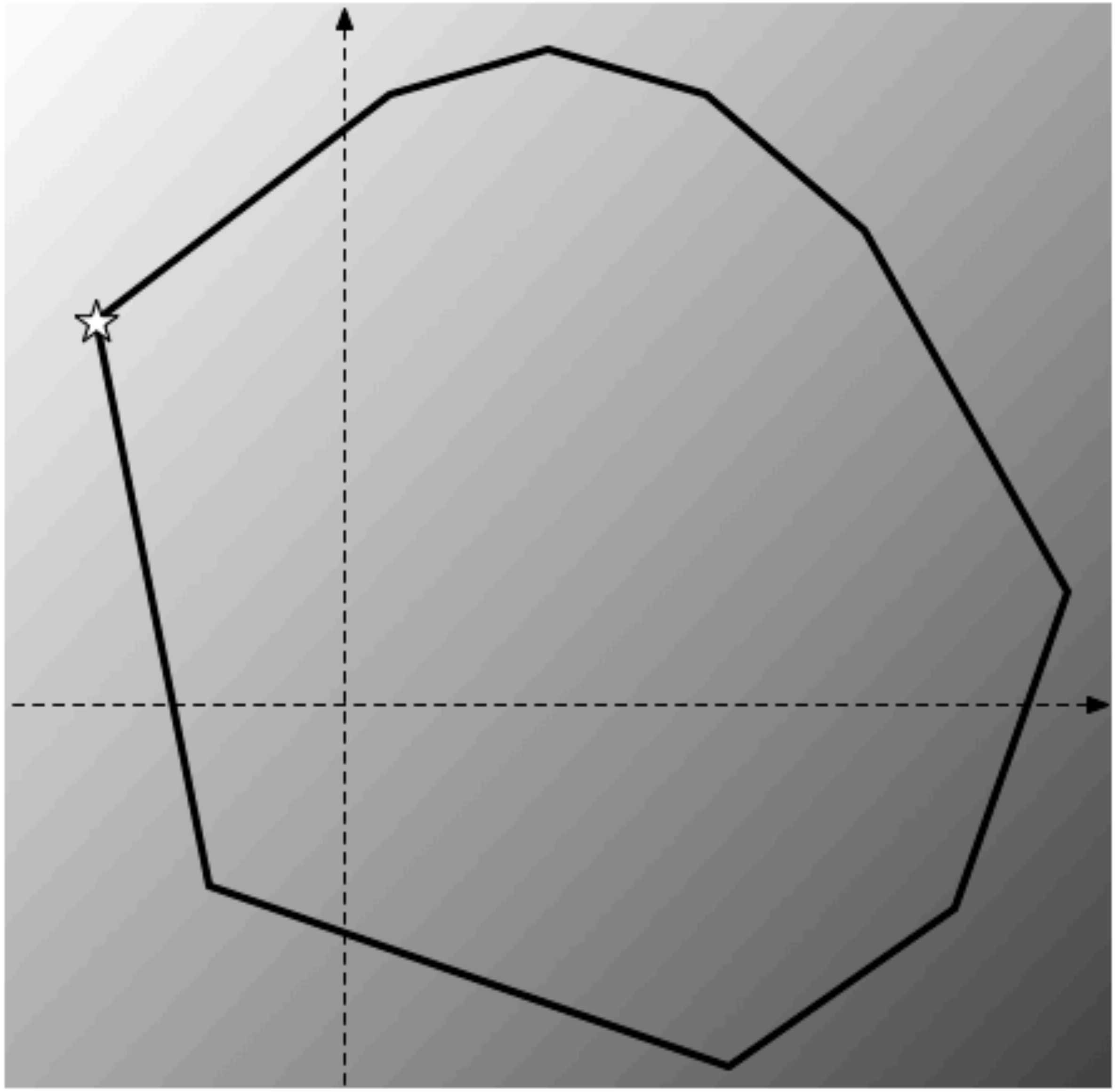


- Not all vertices of the *N*-dimensional unit cube satisfy the constraints.
  - E.g., can't have $z_{1,\mathrm{BI}} = 1$ and $z_{2,\mathrm{BI}} = 1$
- Sometimes we can write down a small (polynomial number) of linear constraints on **z**.
- Result: linear objective, linear constraints, integer constraints …



Integer Linear Programming

- Very easy to add new constraints and non-local features.
- Many decoding problems have been mapped to ILP (sequence labeling, parsing, …), but it's *not* always trivial.
- NP-hard in general.
  – But there are packages that often work well in practice (e.g., CPLEX)
  – Specialized algorithms in some cases
  – LP relaxation for approximate solutions



Remark

- Graphical models assumed a probabilistic interpretation
  - Though they are not always learned using a probabilistic interpretation!

- The polytope view is agnostic about how you interpret the weights.
  - It only says that the decoding problem is an ILP.

3. 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

$$
\begin{array}{lll@{\qquad}lll@{\qquad}lll@{\qquad}lll}
N & \rightarrow & B\ R_B & R_B & \rightarrow & B\ R_B & R_I & \rightarrow & B\ R_B & R_O & \rightarrow & B\ R_B \\
N & \rightarrow & O\ R_O & R_B & \rightarrow & O\ R_O & R_I & \rightarrow & O\ R_O & R_O & \rightarrow & O\ R_O \\
 & & & R_B & \rightarrow & I\ R_I & R_I & \rightarrow & I\ R_I & & & \\
 & & & R_B & \rightarrow & \epsilon & R_I & \rightarrow & \epsilon & R_O & \rightarrow & \epsilon \\
\\
\multicolumn{3}{r}{\forall x \in \Sigma,} & B & \rightarrow & x & I & \rightarrow & x & O & \rightarrow & x
\end{array}
$$

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

4. Paths and Hyperpaths

Best Path

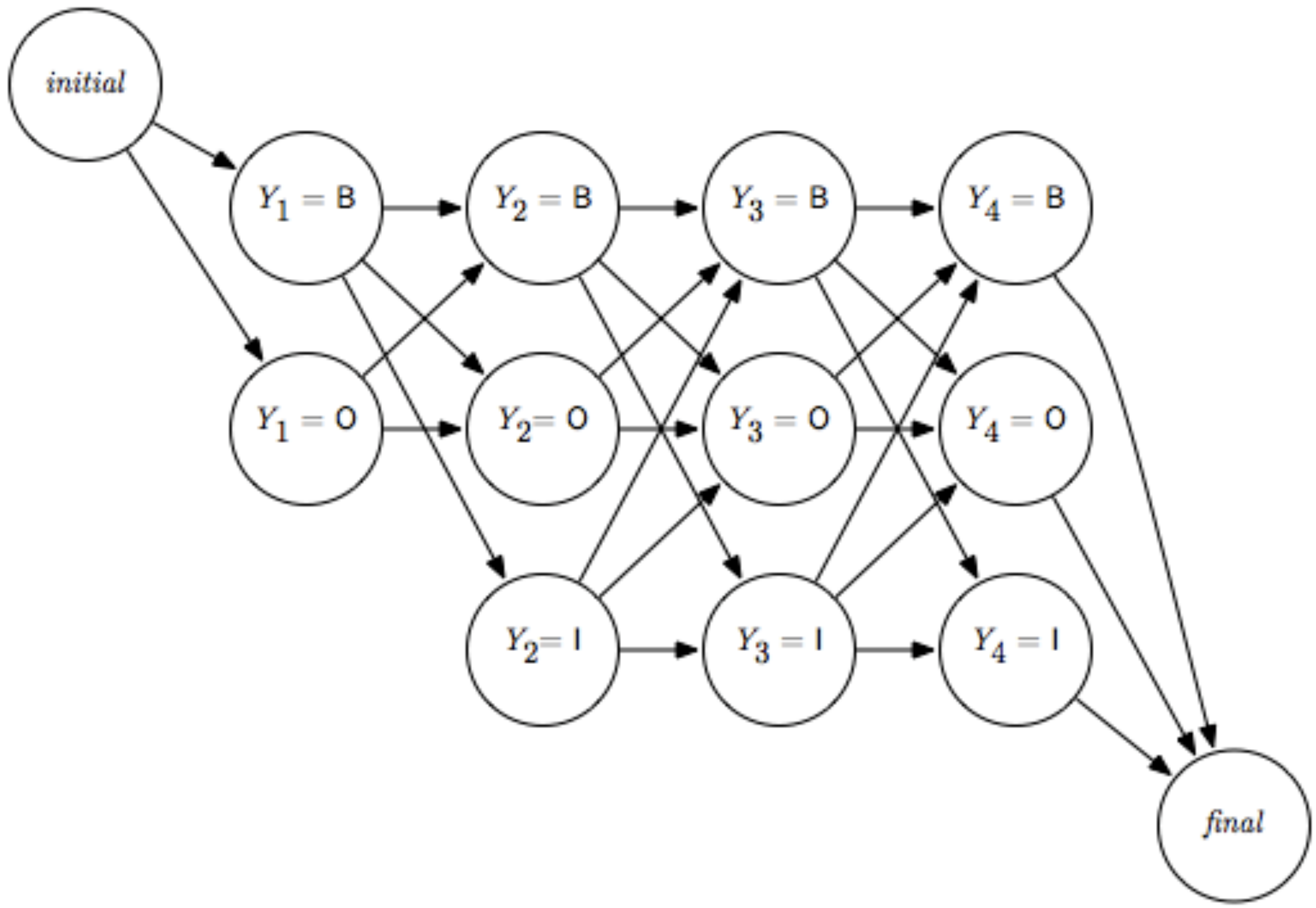
- General idea: take ***x*** and build a graph.
- Score of a path factors into the edges.

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

- Decoding is finding the *best* path.

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

“Lattice” View of Viterbi

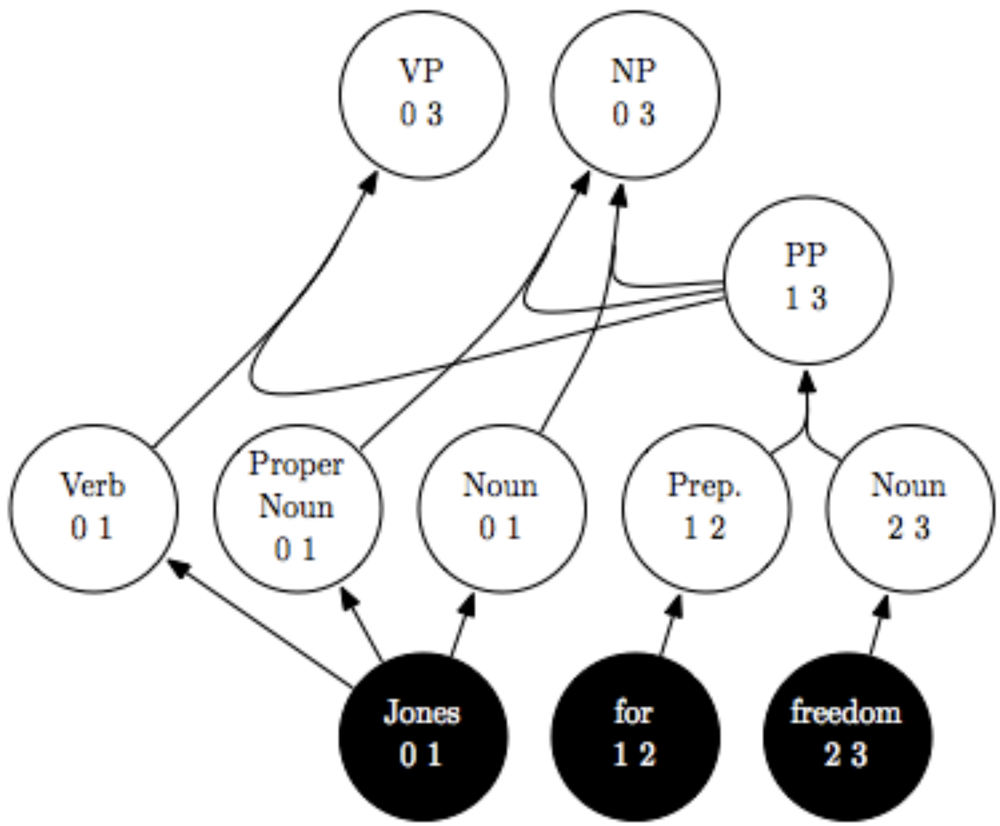


Minimum Cost Hyperpath

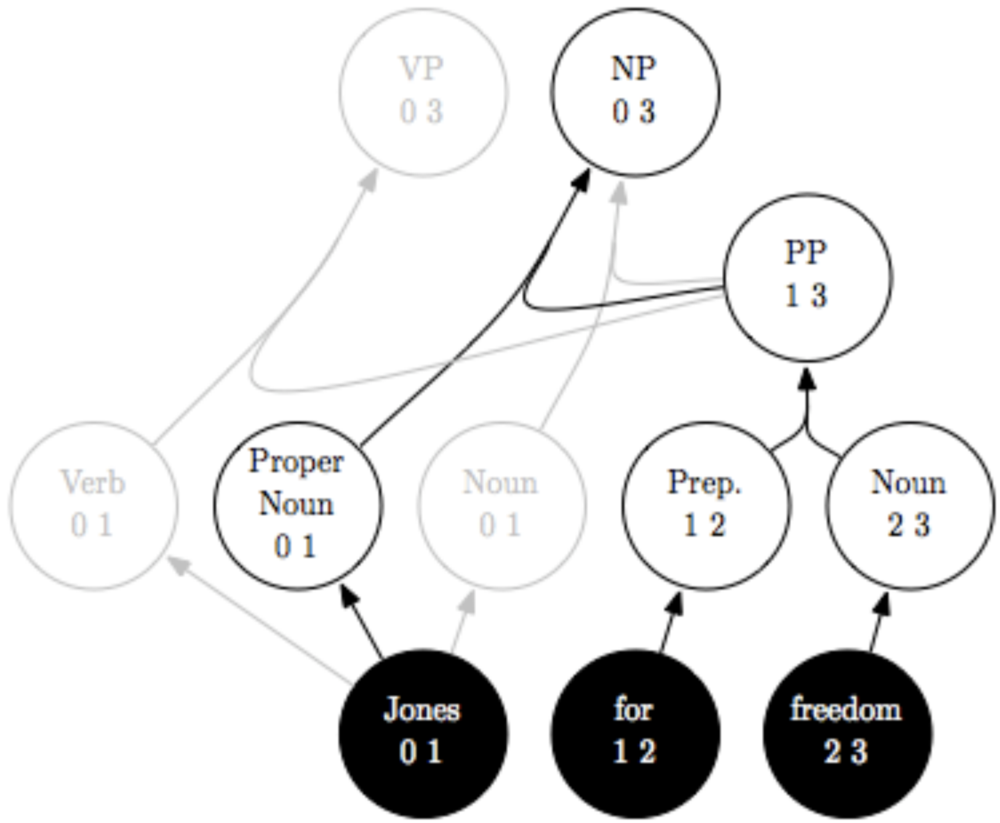
- General idea: take ***x*** and build a hypergraph.
- Score of a hyperpath factors into the hyperedges.
- Decoding is finding the best *hyperpath*.

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

Parsing as a Hypergraph

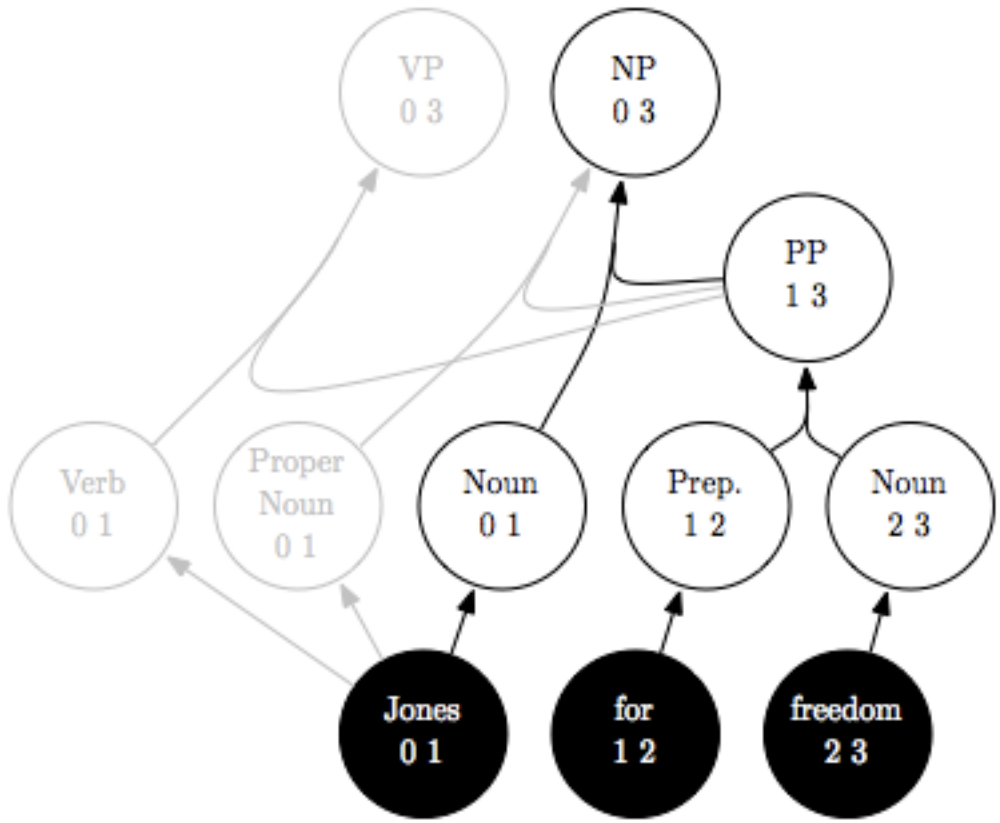


Parsing as a Hypergraph



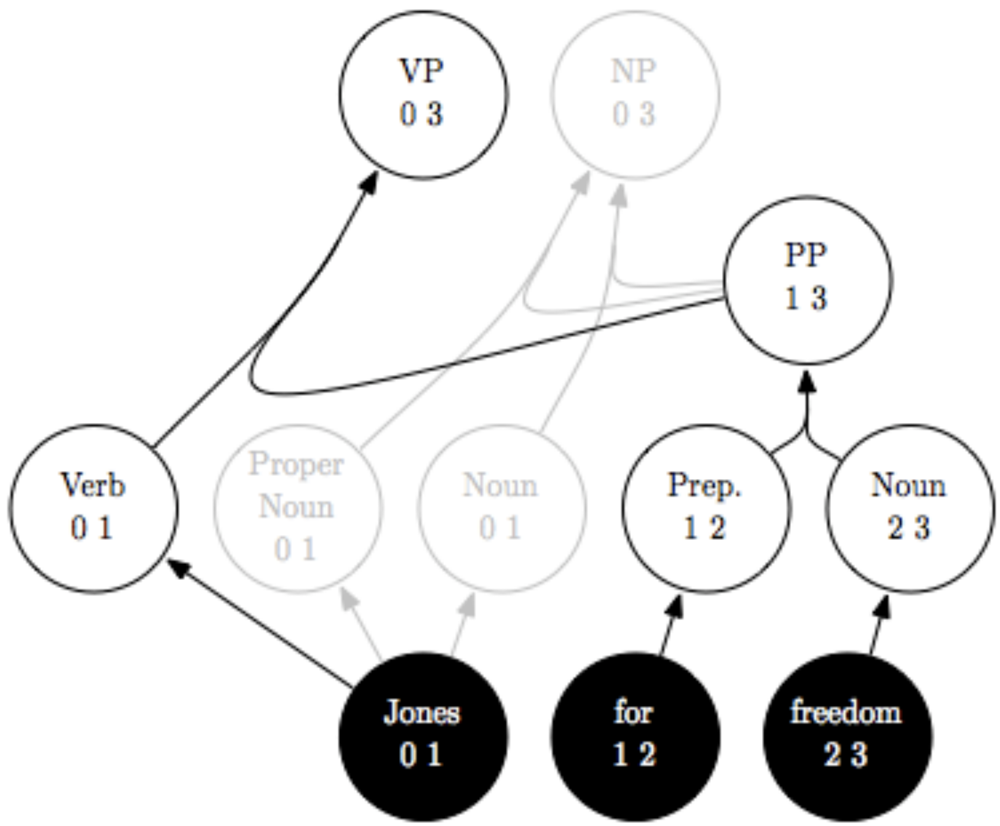
cf. “Dean for democracy”

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.

5. Weighted Logic Programming

Logic Programming

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

$$
\begin{aligned}
\forall A, C, \quad & \mathsf{ancestor}(A, C) && \Leftarrow \quad \mathsf{parent}(A, C) \\
\forall A, C, \quad & \mathsf{ancestor}(A, C) && \Leftarrow \quad \bigvee_{B} \mathsf{ancestor}(A, B) \wedge \mathsf{parent}(B, C)
\end{aligned}
$$

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

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

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

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

- Most views (exception is polytopes) can be understood as DP algorithms.
  - The low-level *procedures* we use are often DP.
  - Even DP is too high-level to know the best way to implement.
- 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, ...
  - Other views suggest others.

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).
- No perfect general view, but much can be gained through a combination of views.