

Probability and Structure in Natural Language Processing

Noah Smith

Heidelberg University, November 2014

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?

- I think so.
- Brings out commonalities:
 - Modeling formalisms (e.g., linear models with features)
 - Learning algorithms (lectures 3-4)
 - 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

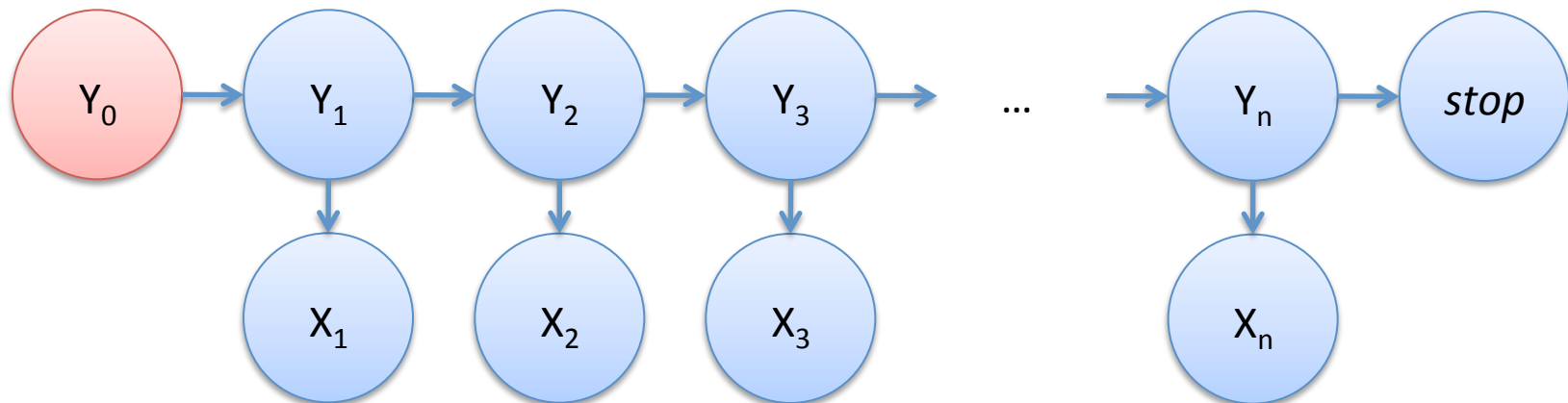
$$p(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}) = \left(\prod_{i=1}^n p(x_i | y_i) p(y_i | y_{i-1}) \right) p(\text{stop} | y_n)$$

- Parameters:
 - Transitions $p(y' | y)$
 - including $p(\text{stop} | y)$, $p(y | \text{start})$
 - Emissions $p(x | y)$

Hidden Markov Model

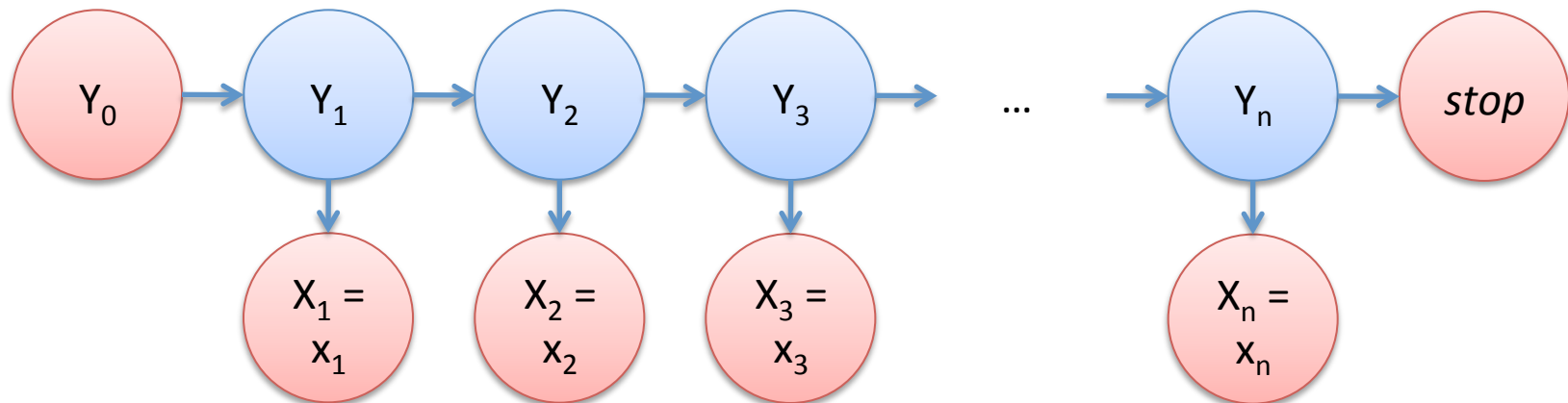
- The joint model's independence assumptions are easy to capture with a Bayesian network.

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



Hidden Markov Model

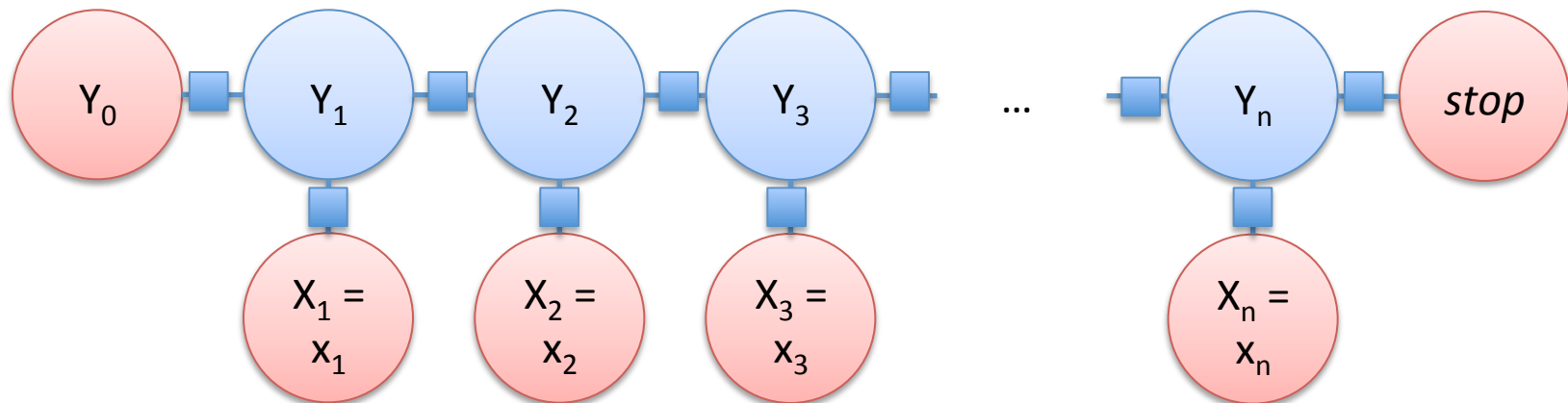
- The usual inference problem is to find the most probable value of \mathbf{Y} given $\mathbf{X} = \mathbf{x}$.



Hidden Markov Model

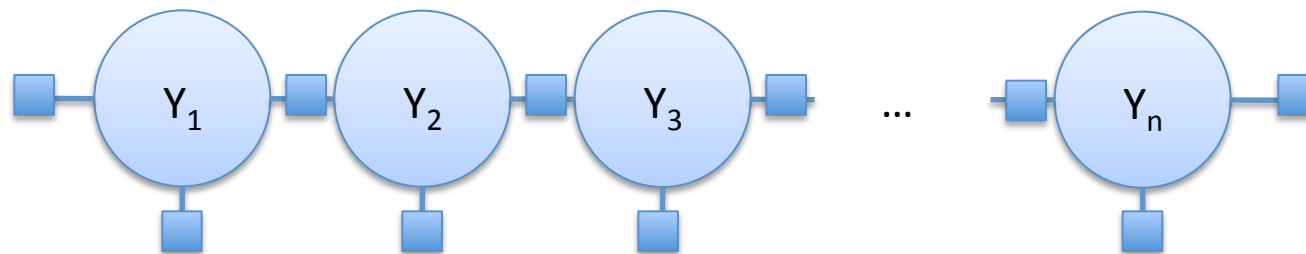
- The usual inference problem is to find the most probable value of \mathbf{Y} given $\mathbf{X} = \mathbf{x}$.

- Factor graph:



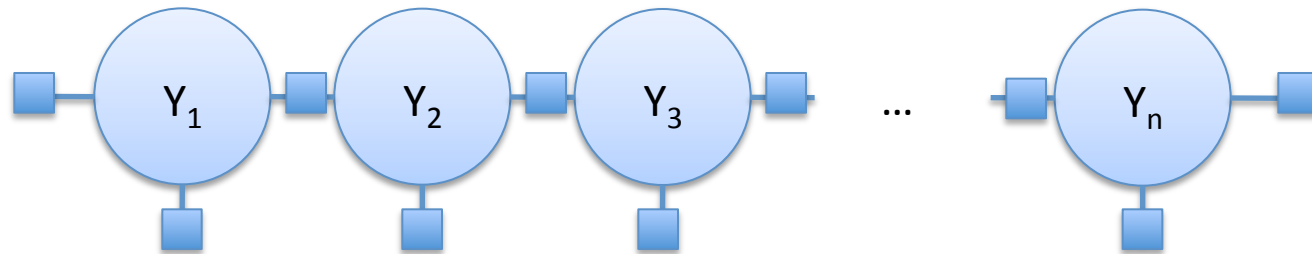
Hidden Markov Model

- The usual inference problem is to find the most probable value of \mathbf{Y} given $\mathbf{X} = \mathbf{x}$.
- Factor graph after reducing factors to respect evidence:



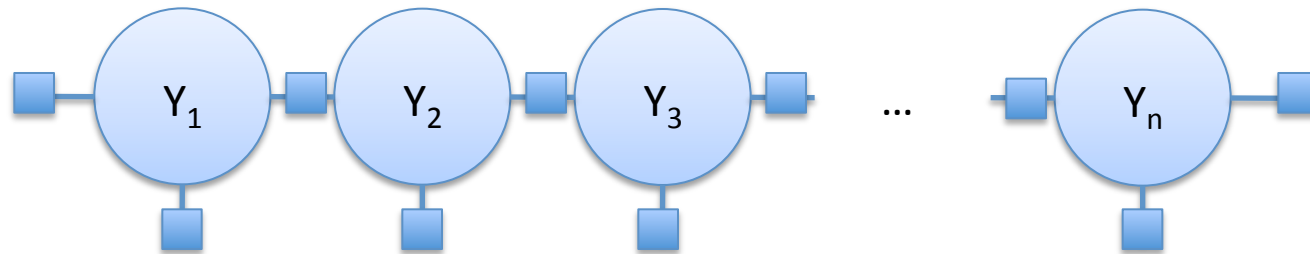
Hidden Markov Model

- The usual inference problem is to find the most probable value of \mathbf{Y} given $\mathbf{X} = \mathbf{x}$.
- Clever ordering should be apparent!



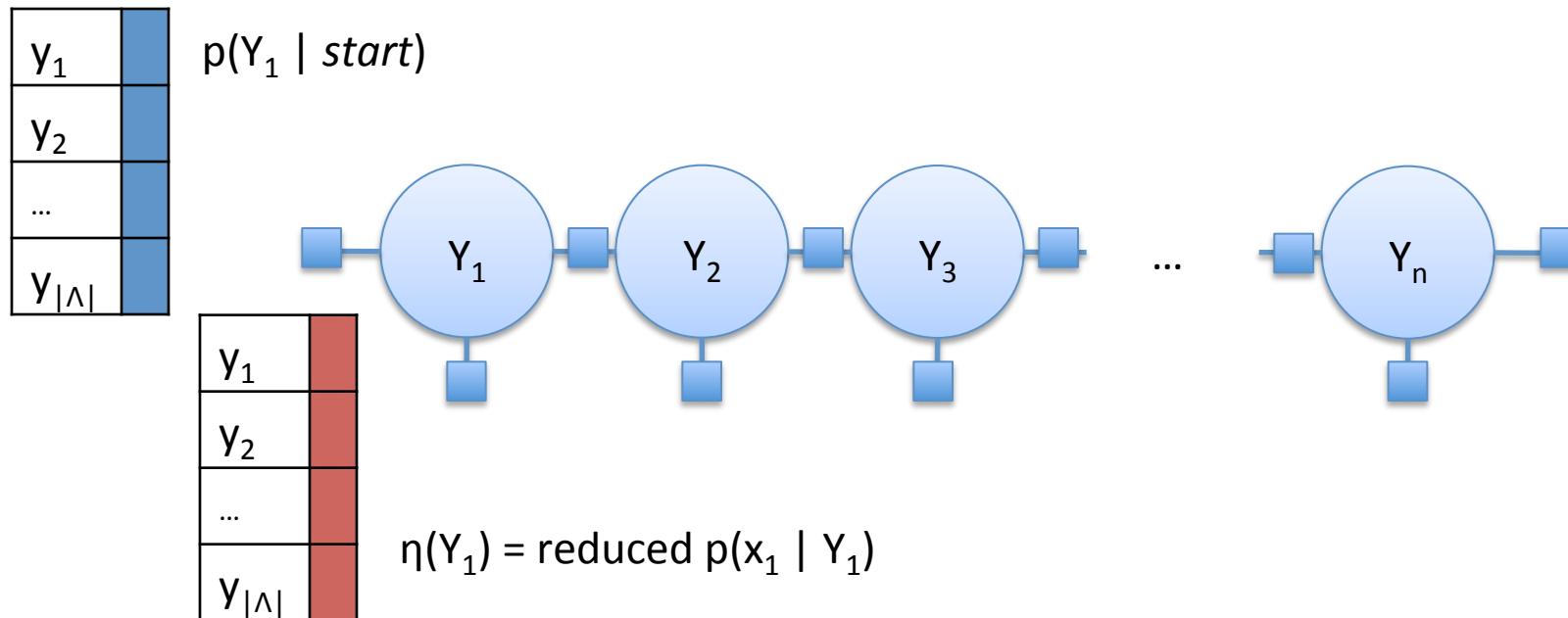
Hidden Markov Model

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



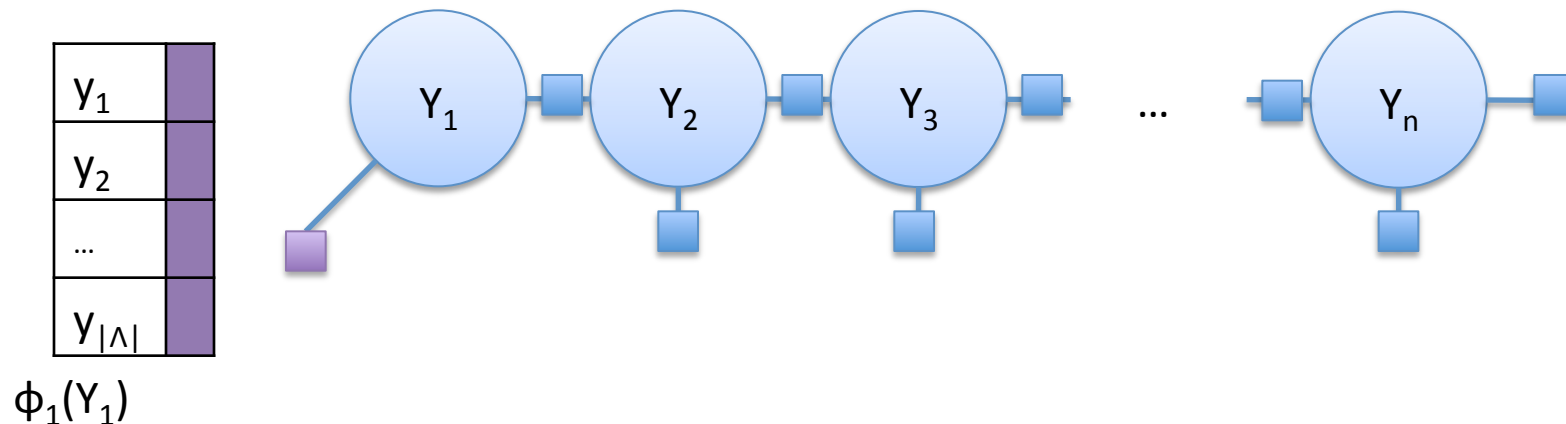
Hidden Markov Model

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



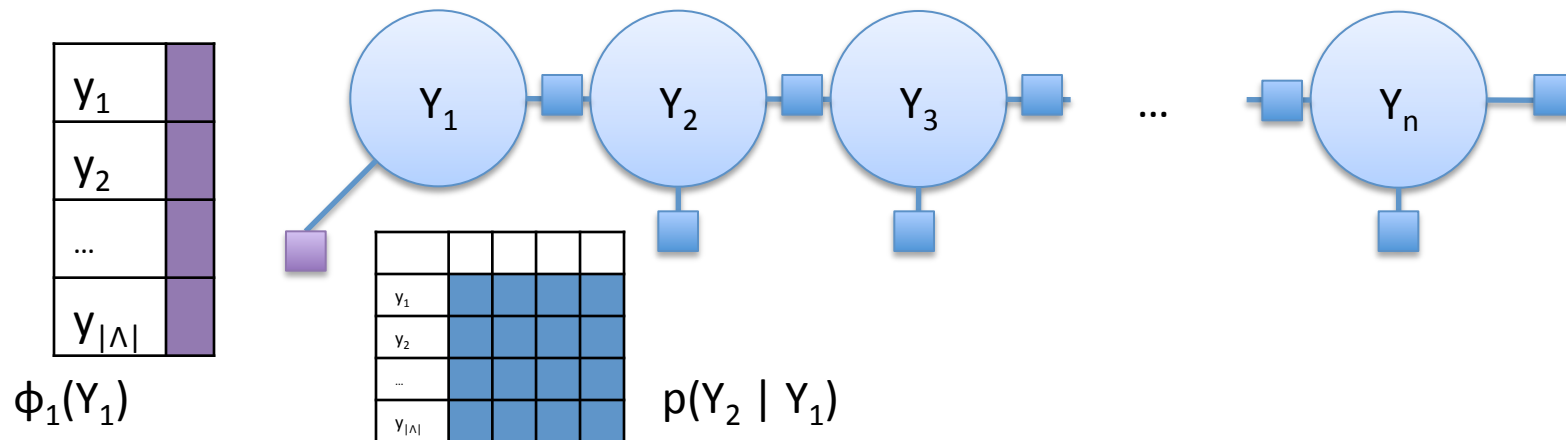
Hidden Markov Model

- 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 .



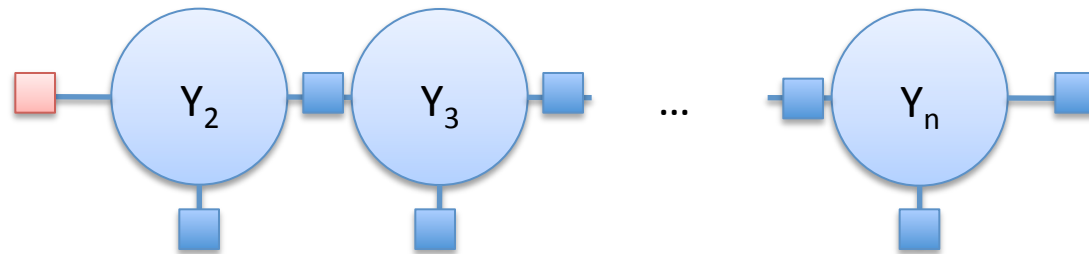
Hidden Markov Model

- 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 .



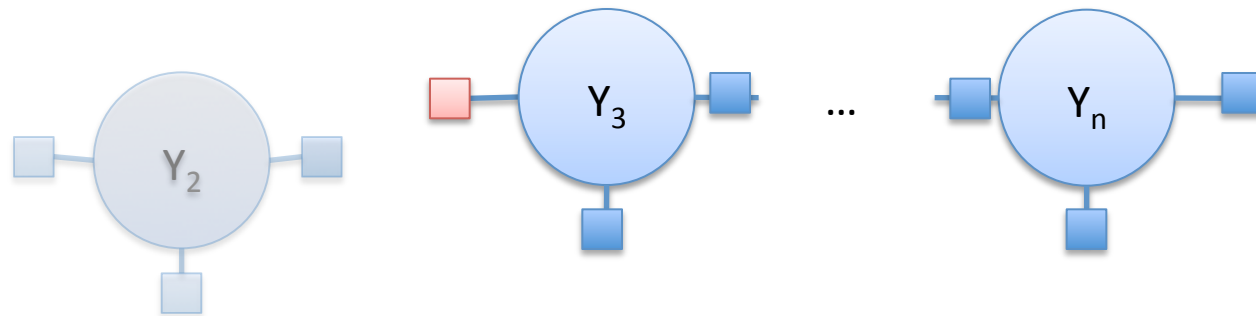
Hidden Markov Model

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



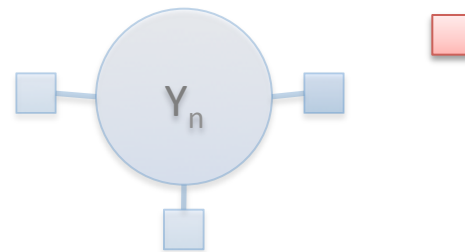
Hidden Markov Model

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



Hidden Markov Model

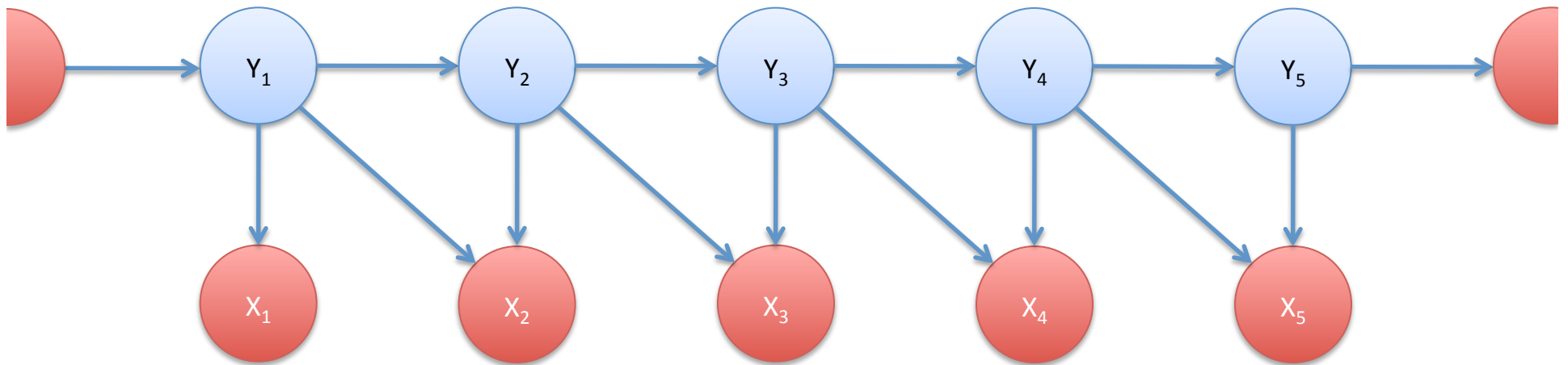
- At the end, we have one final factor with one row, ϕ_{n+1} .
- This is the score of the best sequence.
- Use backtrace to recover values.



Why Think This Way?

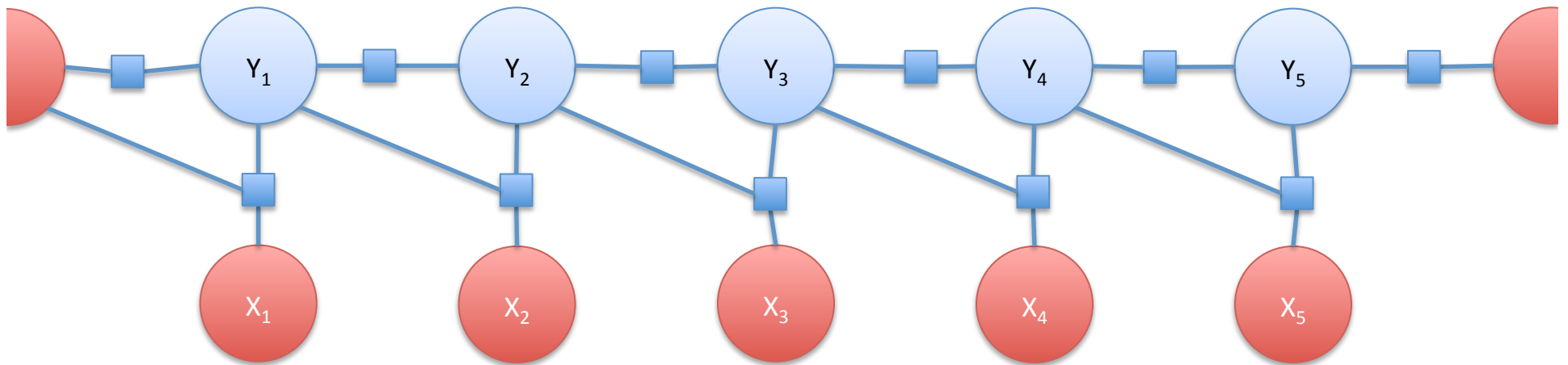
- Easy to see how to generalize HMMs.
 - More evidence
 - More factors
 - More hidden structure
 - More dependencies
- Probabilistic interpretation of factors is *not* central to finding the “best” Y ...
 - Many factors are not conditional probability tables.

Generalization Example 1



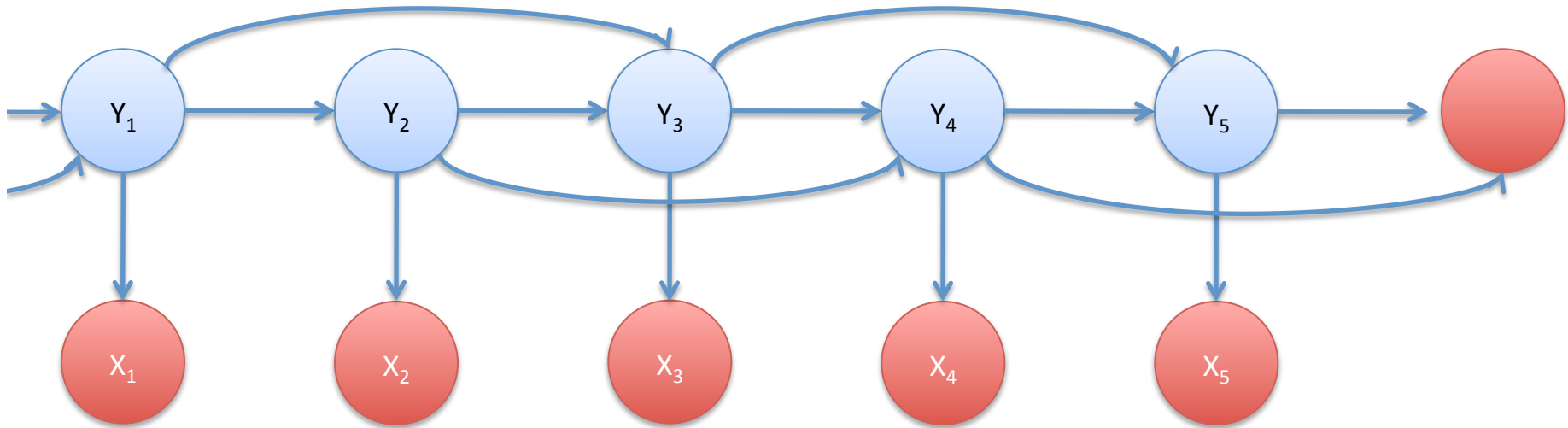
- Each word also depends on previous state.

Generalization Example 1



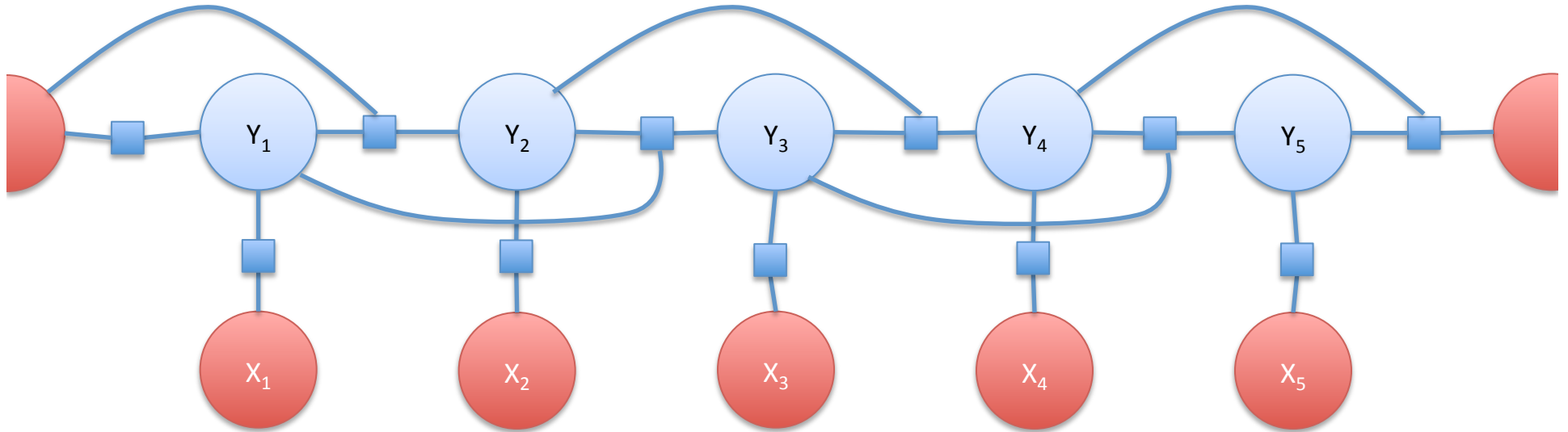
- Each word also depends on previous state.

Generalization Example 2



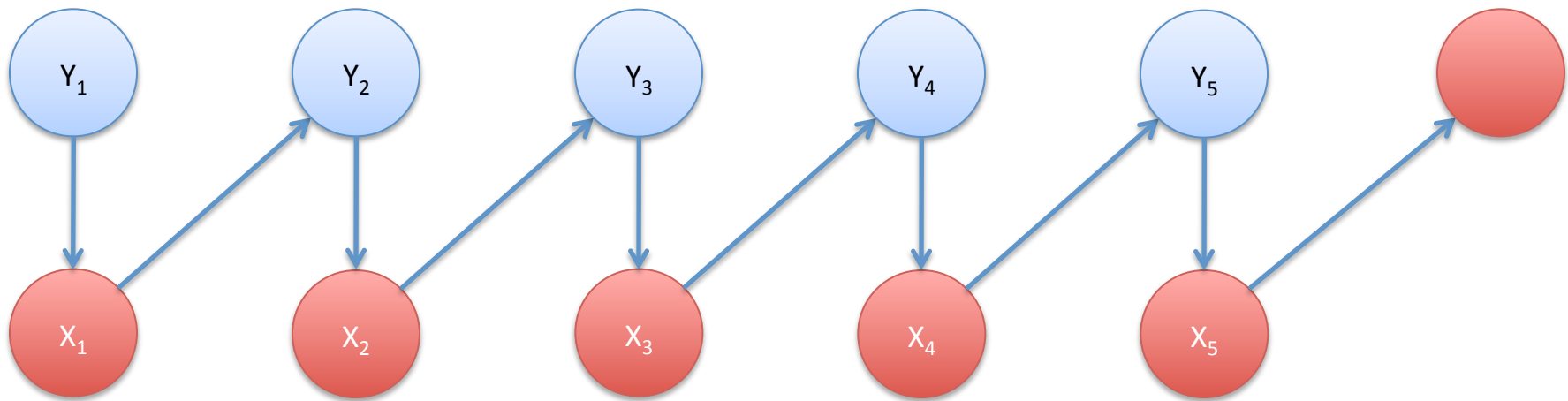
- “Trigram” HMM

Generalization Example 2



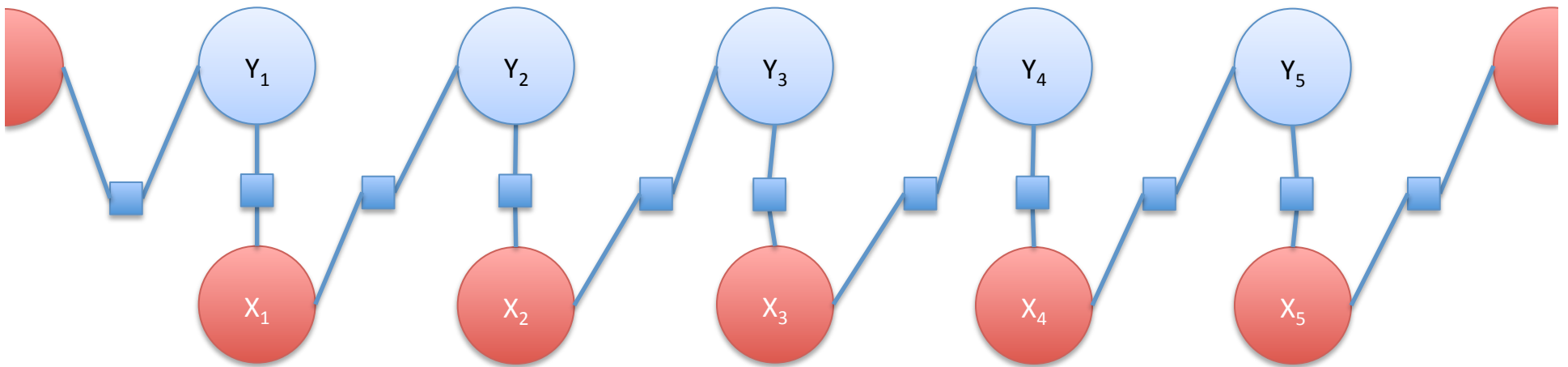
- “Trigram” HMM

Generalization Example 3



- Aggregate bigram model (Saul and Pereira, 1997)

Generalization Example 3



- Aggregate bigram model (Saul and Pereira, 1997)

General Decoding Problem

- Two structured random variables, \mathbf{X} and \mathbf{Y} .
 - Sometimes described as *collections* of random variables.
- “Decode” observed value $\mathbf{X} = \mathbf{x}$ into some value of \mathbf{Y} .
- Usually, we seek to maximize some score.
 - E.g., MAP inference from yesterday.

Linear Models

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

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

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

Generic Noisy Channel as Linear Model

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

- 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

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

HMMs as Linear Models

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

Running Example

	1	2	3	4	5	6	7	8	9	10	
x	=	Britain	sent	warships	across	the	English	Channel	Monday	to	rescue
y	=	B	O	O	O	O	B	I	B	O	O
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, M³Ns ...

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

(What is *Not* A Linear Model?)

- Models with hidden variables

$$\arg \max_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x}) = \arg \max_{\mathbf{y}} \sum_z p(\mathbf{y}, \mathbf{z} \mid \mathbf{x})$$

(also “neural” models)

- Models based on non-linear kernels

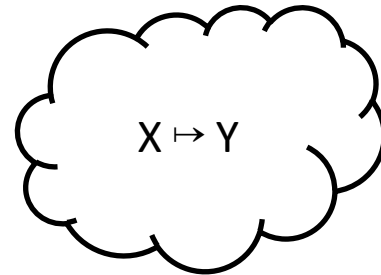
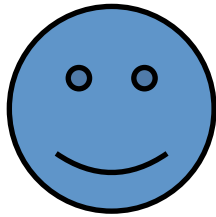
$$\arg \max_{\mathbf{y}} \mathbf{w}^\top \mathbf{g}(\mathbf{x}, \mathbf{y}) = \arg \max_{\mathbf{y}} \sum_{i=1}^N \alpha_i K(\langle \mathbf{x}_i, \mathbf{y}_i \rangle, \langle \mathbf{x}, \mathbf{y} \rangle)$$

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

Inference in a
probabilistic
graphical model!



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:

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

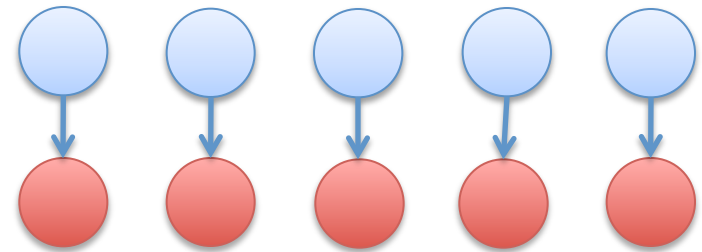
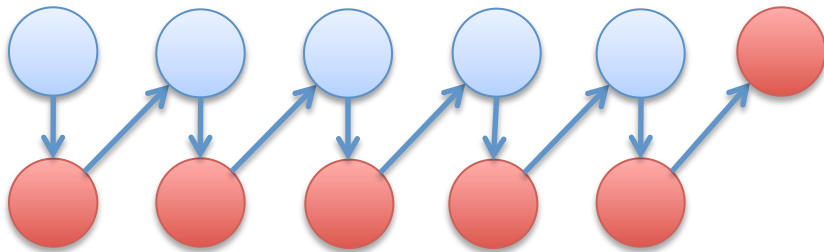
- Markov network:

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

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

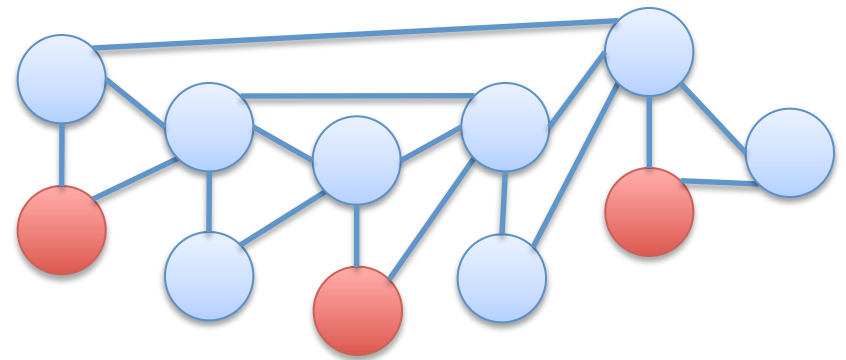
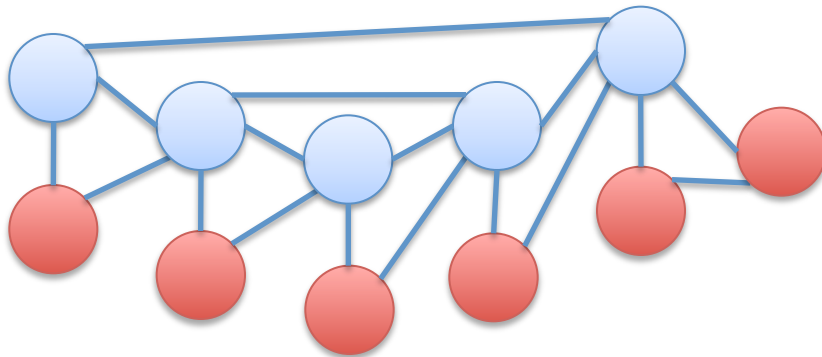
Inference in Graphical Models

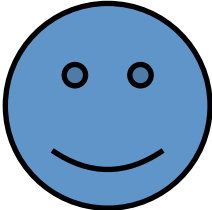
- Remember: more edges make inference more expensive.
 - Fewer edges means stronger independence.
- Really pleasant:



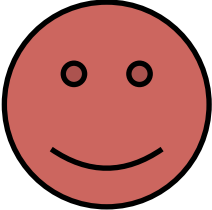
Inference in Graphical Models

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

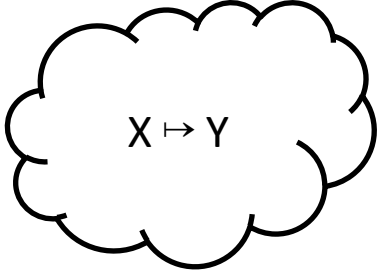




Inference in a
probabilistic
graphical model!



Integer linear
programming!



$X \mapsto Y$











“Parts”

- Assume that feature function \mathbf{g} breaks down into local parts.

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

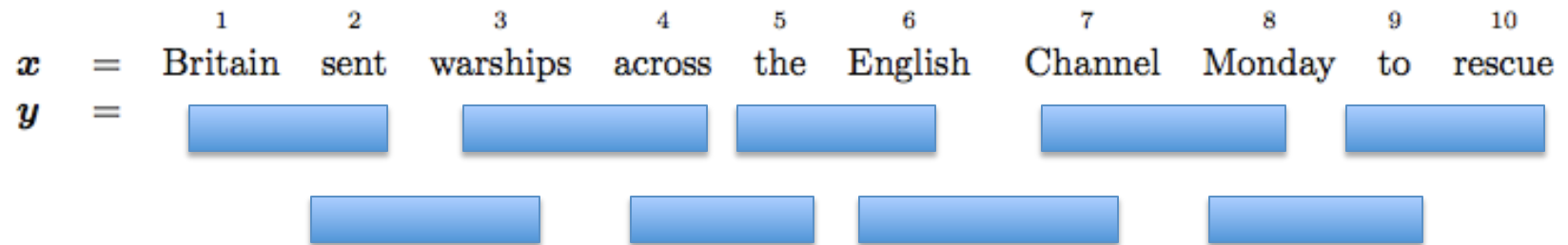
- Each part has an alphabet of possible values.
 - Decoding is choosing values for all parts, with **consistency** constraints.
 - (In the graphical models view, a part corresponds to a factor assignment.)

Example

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

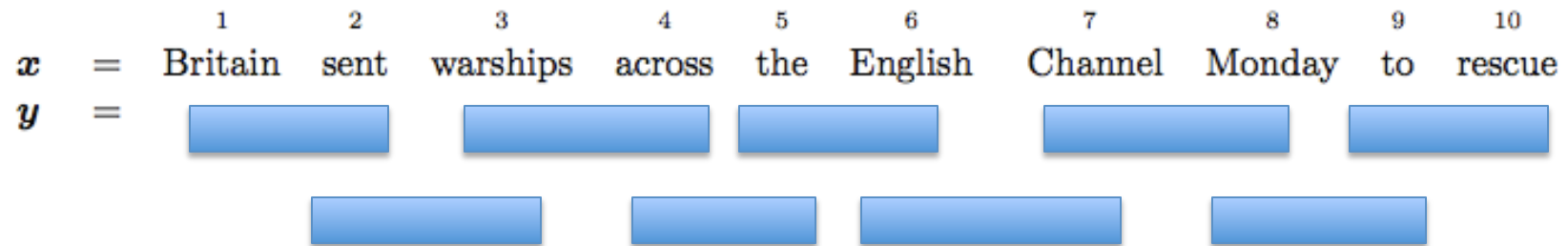
- One part per word, each is in $\{B, I, O\}$
- No features look at multiple parts
 - Fast inference
 - Not very expressive

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

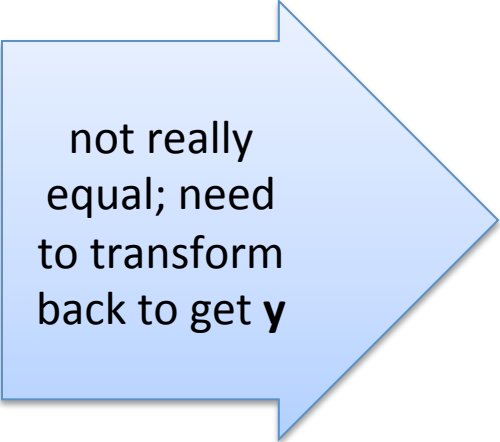
Geometric View



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

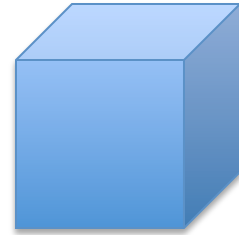
Score is Linear in \mathbf{z}

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

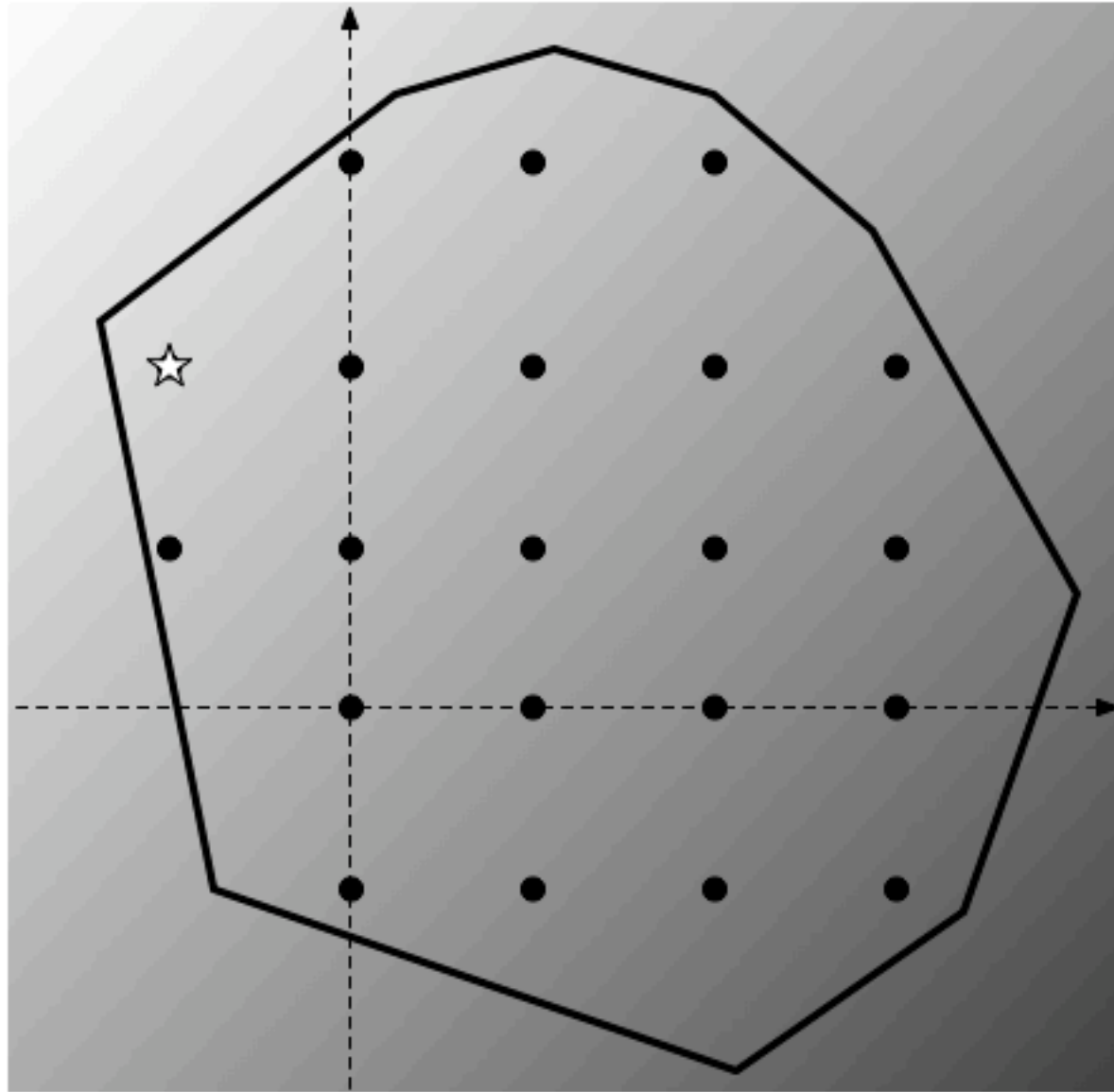


not really
equal; need
to transform
back to get \mathbf{y}

Polyhedra

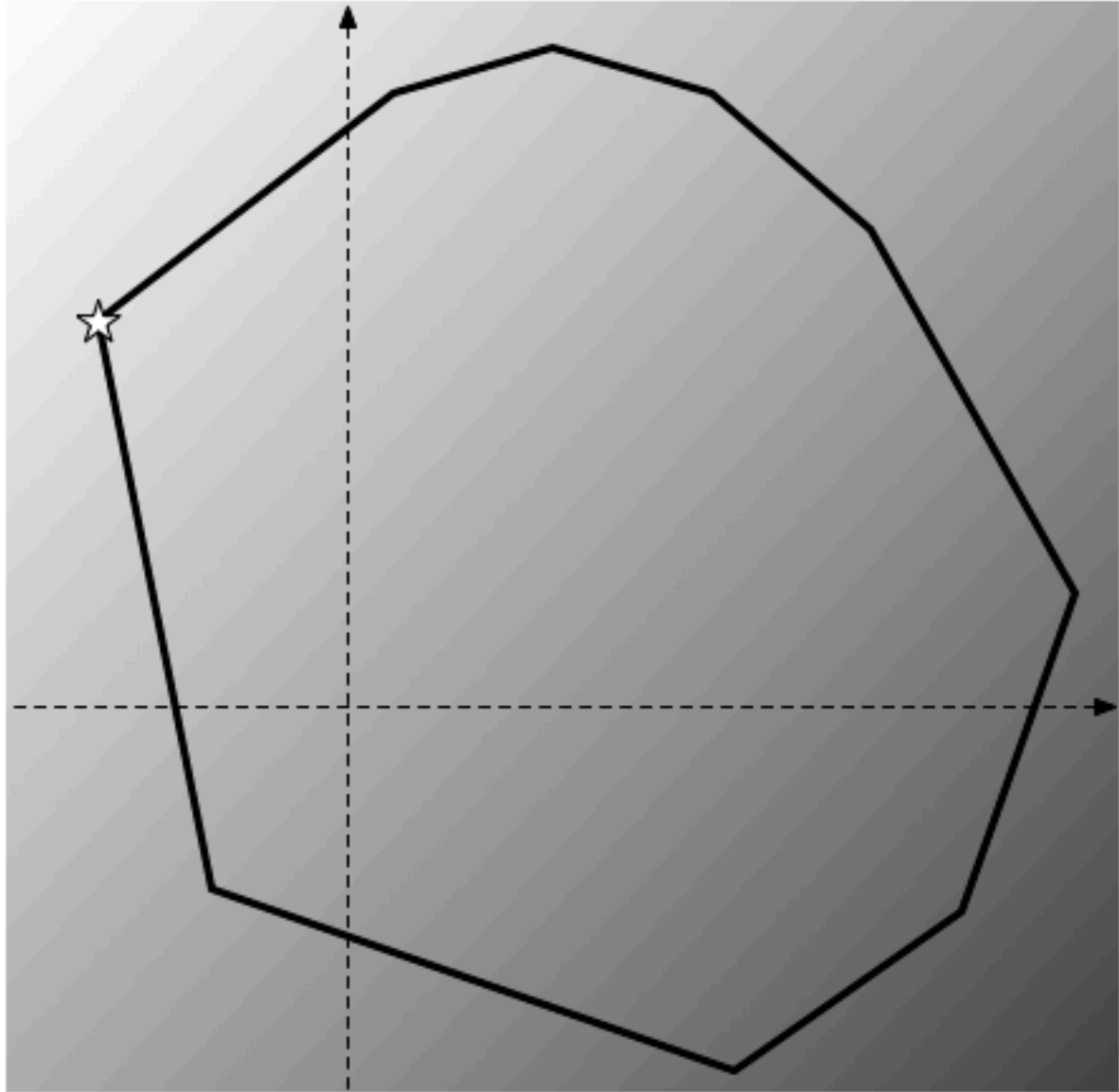


- Not all vertices of the N -dimensional unit cube satisfy the constraints.
 - E.g., can't have $z_{1,BI} = 1$ and $z_{2,BI} = 1$
- Sometimes we can write down a small (polynomial number) of linear constraints on \mathbf{z} .
- Result: linear objective, linear constraints, integer constraints ...



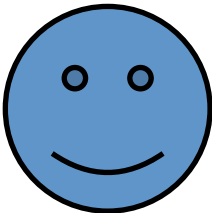
Integer Linear Programming

- 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

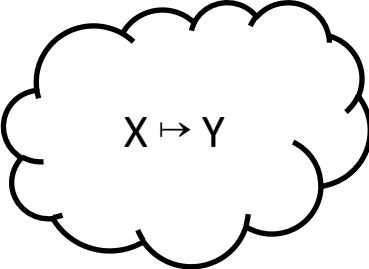
- Graphical models assumed a probabilistic interpretation
 - Though they are not always learned using a probabilistic interpretation!
- The polytope view is agnostic about how you interpret the weights.
 - It only says that the decoding problem is an ILP.




Inference in a
probabilistic
graphical model!



Integer linear
programming!



$X \mapsto Y$



Parsing (with
weights)!

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

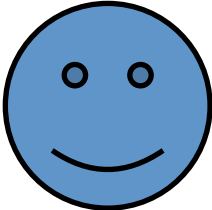
- Every valid \mathbf{y} is a grammatical derivation (parse) for \mathbf{x} .
 - HMM: sequence of “grammatical” states is one allowed by the transition table.
- Augment parsing algorithms with weights and find the best parse.

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

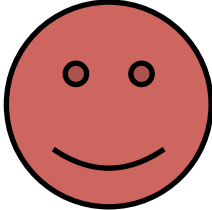
BIO Tagging as a CFG

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

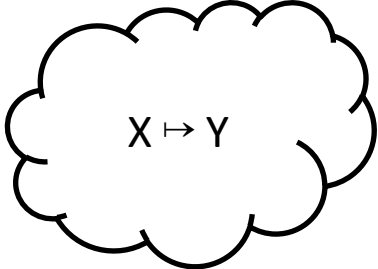
- Weighted (or probabilistic) CKY is a dynamic programming algorithm very similar in structure to classical CKY.




Inference in a
probabilistic
graphical model!



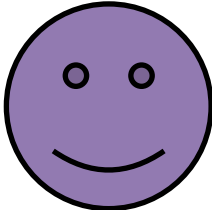
Integer linear
programming!



$X \mapsto Y$



Parsing (with
weights)!



Shortest
(hyper)path!

Best Path

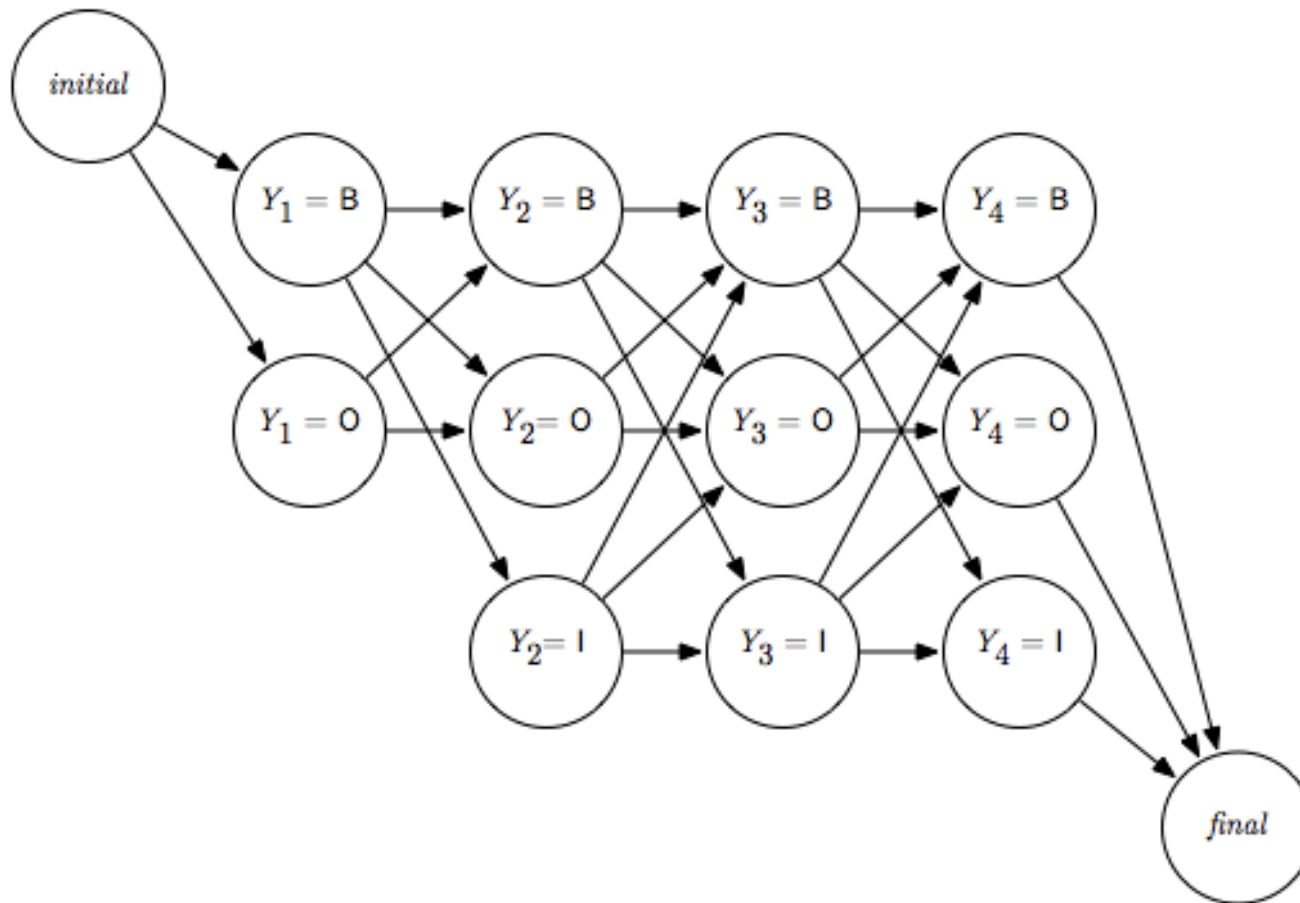
- General idea: take \mathbf{x} and build a **graph**.
- Score of a **path** factors into the **edges**.

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

- Decoding is finding the *best* path.

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

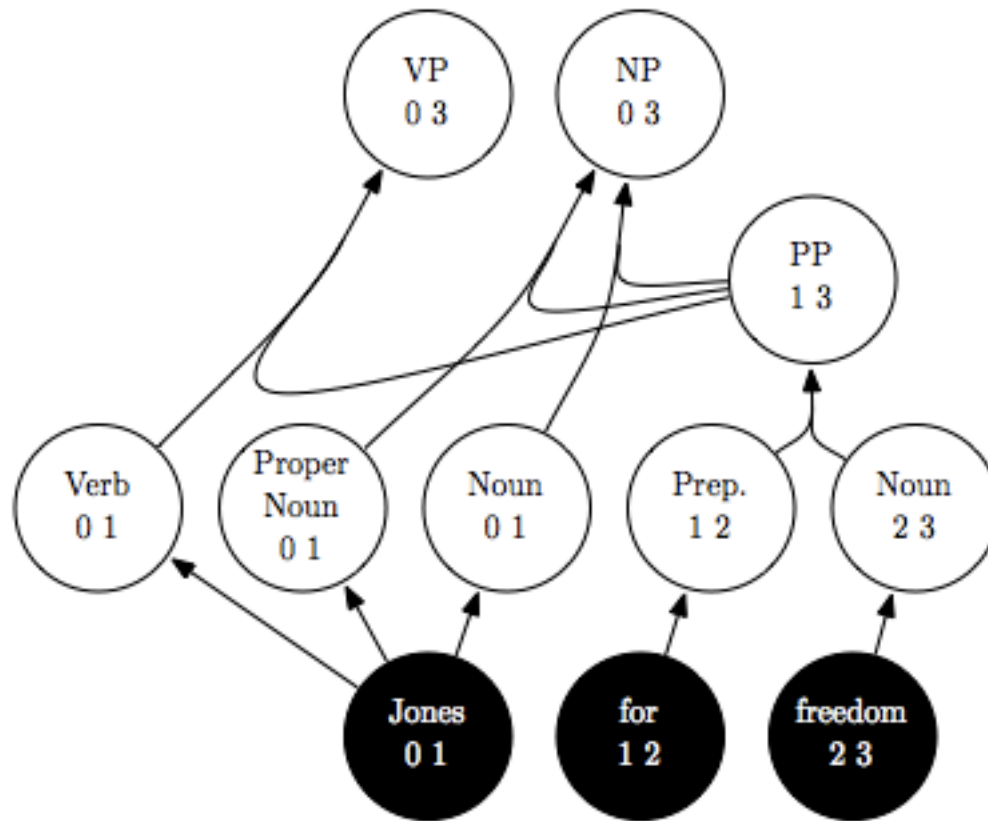
“Lattice” View of Viterbi



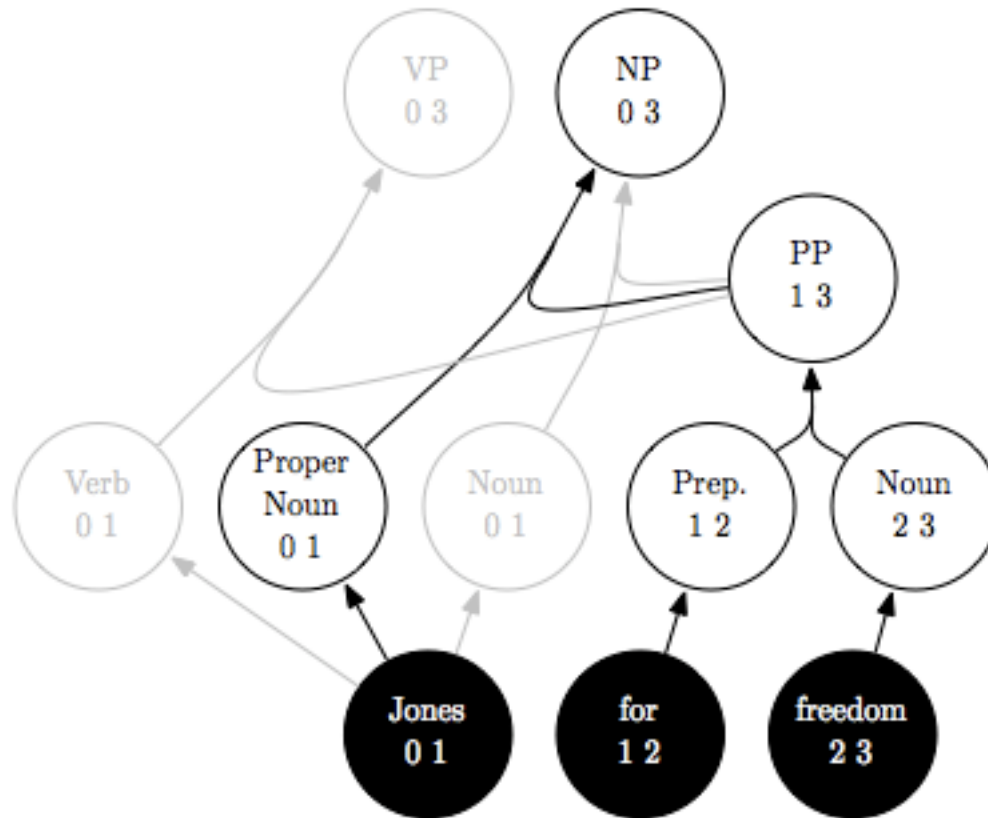
Minimum Cost Hyperpath

- General idea: take \mathbf{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

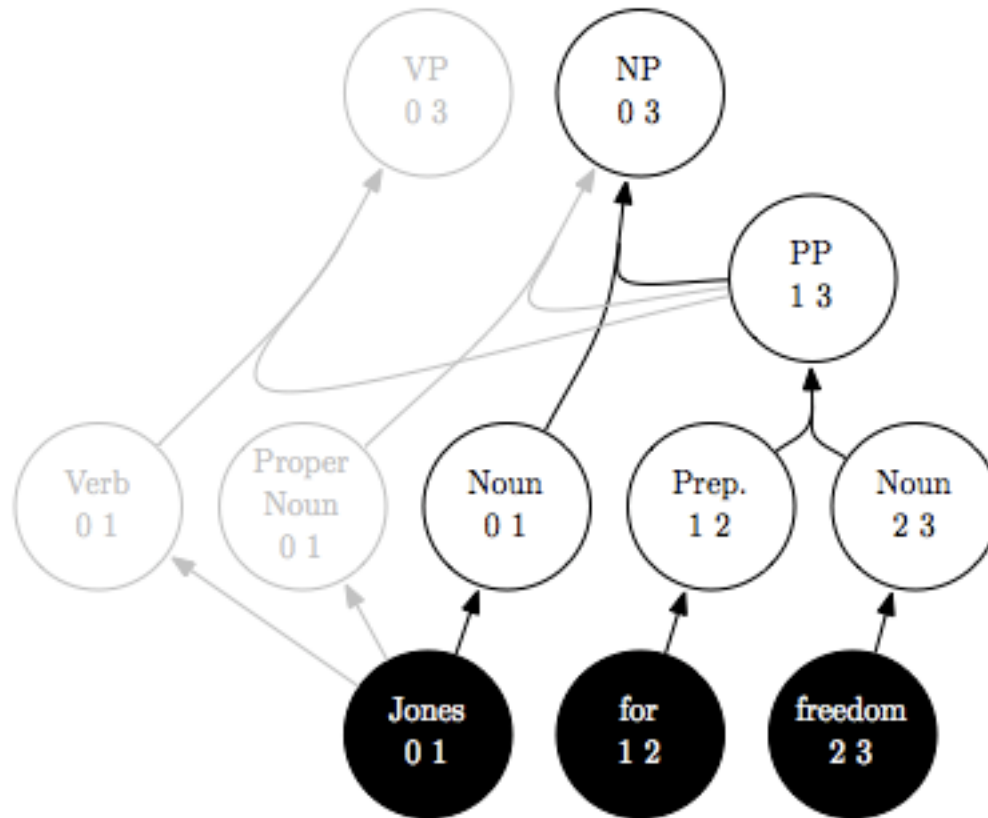


Parsing as a Hypergraph



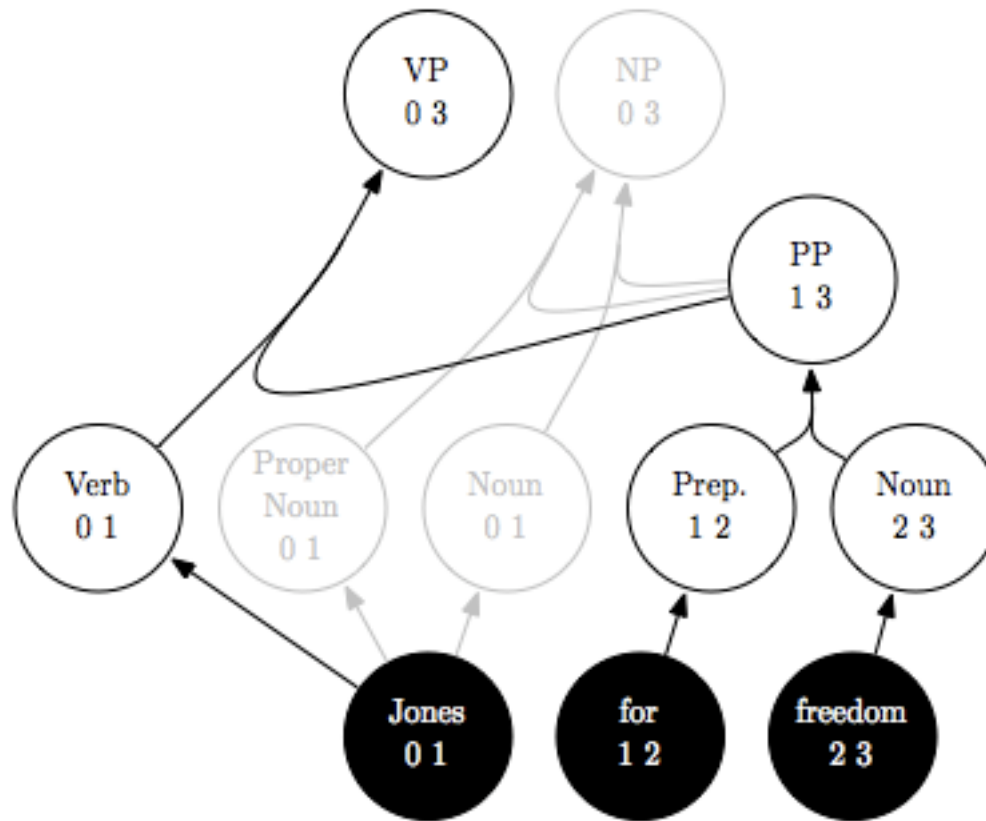
cf. "Dean for democracy"

Parsing as a Hypergraph



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

Parsing as a Hypergraph

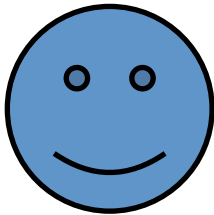


Forced to work on his thesis, sunshine streaming in the window, Mike began to ...

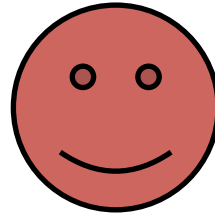
Why Hypergraphs?

- 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.

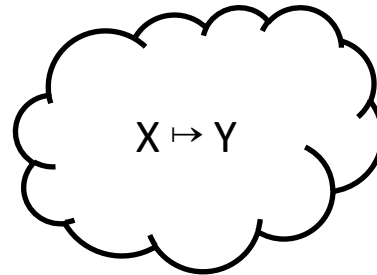
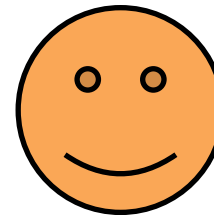
Inference in a probabilistic graphical model!



Integer linear programming!



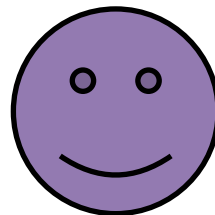
Weighted logic programming!



Parsing (with weights)!



Shortest path!



Logic Programming

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

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

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

label-bigram("B", "B")

label-bigram("B", "I")

label-bigram("B", "O")

label-bigram("I", "B")

label-bigram("I", "I")

label-bigram("I", "O")

label-bigram("O", "B")

label-bigram("O", "O")

$\forall x \in \Sigma,$ labeled-word(x , "B")

$\forall x \in \Sigma,$ labeled-word(x , "I")

$\forall x \in \Sigma,$ labeled-word(x , "O")

$\forall \ell \in \Lambda,$ $v(\ell, 1) =$ labeled-word(x_1, ℓ)

$\forall \ell \in \Lambda,$ $v(\ell, i) = \bigvee_{\ell' \in \Lambda} v(\ell', i - 1) \wedge \text{label-bigram}(\ell', \ell) \wedge \text{labeled-word}(x_i, \ell)$

goal = $\bigvee_{\ell \in \Lambda} v(\ell, n)$

Weighted Deduction

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

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

- Note that axioms can be used more than once in a proof (\mathbf{y}).

Weighted Deduction

- 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.