Probability and Structure in Natural Language Processing

Noah Smith, Carnegie Mellon University

2012 International Summer School in Language and Speech Technologies

Quick Recap

- Yesterday:
  - Bayesian networks and some formal properties
  - Markov networks and some formal properties
  - Exact marginal inference using variable elimination
    - Sum-product version
    - Beginnings of the max-product version

Variable Elimination for Conditional Probabilities

Input: Graphical model on **V**, set of query variables **Y**, evidence **X** = **x**

Output: factor φ and scalar α

1. **Φ** = factors in the model
2. Reduce factors in **Φ** by **X** = **x**
3. Choose variable ordering on **Z** = **V** \ **Y** \ **X**
4. φ = Variable-Elimination(**Φ**, **Z**)
5. $\alpha = \sum_{\mathbf{z} \in Val(\mathbf{Z})} \varphi(\mathbf{z})$
6. Return φ, α

From Marginals to MAP

- Replace factor marginalization steps with *maximization*.
  - Add bookkeeping to keep track of the maximizing values.
- Add a traceback at the end to recover the solution.

- This is analogous to the connection between the forward algorithm and the Viterbi algorithm.
  - Ordering challenge is the same.

Factor Maximization

- Given **X** and Y (Y $\notin$ **X**), we can turn a factor φ(**X**, Y) into a factor ψ(**X**) via maximization:

$$\psi(\boldsymbol{X}) \quad = \quad \max_{Y} \phi(\boldsymbol{X}, Y)$$

- We can refer to this new factor by $\max_Y$ φ.

Factor Maximization

- Given **X** and Y (Y ∉ **X**), we can turn a factor φ(**X**, Y) into a factor ψ(**X**) via maximization:

$$\psi(\boldsymbol{X}) \quad = \quad \max_{Y} \phi(\boldsymbol{X}, Y)$$

A	B	C	φ (A, B, C)
0	0	0	0.9
0	0	1	0.3
0	1	0	1.1
0	1	1	1.7
1	0	0	0.4
1	0	1	0.7
1	1	0	1.1
1	1	1	0.2

"maximizing out" B

A	C	ψ(A, C)	
0	0	1.1	B=1
0	1	1.7	B=1
1	0	1.1	B=1
1	1	0.7	B=0

Distributive Property

- A useful property we exploited in variable elimination:

$$X \notin \mathrm{Scope}(\phi_1) \quad \Rightarrow \quad \sum_X (\phi_1 \cdot \phi_2) = \phi_1 \cdot \sum_X \phi_2$$

- Under the same conditions, factor multiplication distributes over max, too:

$$\max_X (\phi_1 \cdot \phi_2) = \phi_1 \cdot \max_X \phi_2$$

Traceback

Input: Sequence of factors with associated variables: $(\psi_{Z_1}, \ldots, \psi_{Z_k})$

Output: $\mathbf{z}^*$

- Each ψ_Z is a factor with scope including Z and variables eliminated *after* Z.
- Work backwards from i = k to 1:
  - Let $z_i = \arg\max_z \psi_{Z_i}(z, z_{i+1}, z_{i+2}, \ldots, z_k)$
- Return **z**

About the Traceback

- No extra (asymptotic) expense.
  - Linear traversal over the intermediate factors.
- The factor operations for both sum-product VE and max-product VE can be generalized.
  - Example: get the K most likely assignments

Eliminating One Variable (Max-Product Version with Bookkeeping)

Input: Set of factors $\mathbf{\Phi}$, variable Z to eliminate

Output: new set of factors $\mathbf{\Psi}$

1. Let $\mathbf{\Phi}'$ = $\{\varphi \in \mathbf{\Phi} \mid Z \in \text{Scope}(\varphi)\}$
2. Let $\mathbf{\Psi}$ = $\{\varphi \in \mathbf{\Phi} \mid Z \notin \text{Scope}(\varphi)\}$
3. Let τ be $\max_Z \prod_{\varphi \in \mathbf{\Phi}'} \varphi$
   - Let ψ be $\prod_{\varphi \in \mathbf{\Phi}'} \varphi$ (bookkeeping)
4. Return $\mathbf{\Psi} \cup \{\tau\}$, ψ

Variable Elimination (Max-Product Version with Decoding)

Input: Set of factors $\mathbf{\Phi}$, ordered list of variables $\mathbf{Z}$ to eliminate

Output: new factor

1. For each $Z_i \in \mathbf{Z}$ (in order):
   – Let $(\mathbf{\Phi}, \psi_{Z_i})$ = Eliminate-One$(\mathbf{\Phi}, Z_i)$
2. Return $\prod_{\varphi \in \mathbf{\Phi}} \varphi$, Traceback$(\{\psi_{Z_i}\})$

Variable Elimination Tips

- Any ordering will be correct.
- Most orderings will be too expensive.
- There are heuristics for choosing an ordering (you are welcome to find them and test them out).

(Rocket Science: True MAP)

- Evidence: **X** = **x**
- Query: **Y**
- Other variables: **Z** = **V** \ **X** \ **Y**

$$\begin{aligned} \boldsymbol{y}^* &= \arg \max_{\boldsymbol{y} \in \mathrm{Val}(\boldsymbol{Y})} P(\boldsymbol{Y} = \boldsymbol{y} \mid \boldsymbol{X} = \boldsymbol{x}) \\ &= \arg \max_{\boldsymbol{y} \in \mathrm{Val}(\boldsymbol{Y})} \sum_{\boldsymbol{z} \in \mathrm{Val}(\boldsymbol{Z})} P(\boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{Z} = \boldsymbol{z} \mid \boldsymbol{X} = \boldsymbol{x}) \end{aligned}$$

- First, marginalize out **Z**, then do MAP inference over **Y** given **X** = **x**

- This is not usually attempted in NLP, with some exceptions.

Parting Shots

- You will probably never implement the general variable elimination algorithm.
- You will rarely use exact inference.

- There is value in understanding the *problem* that approximation methods are trying to solve, and what an *exact* (if intractable) solution would look like!

Lecture 3:
Structures and 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?

- I 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

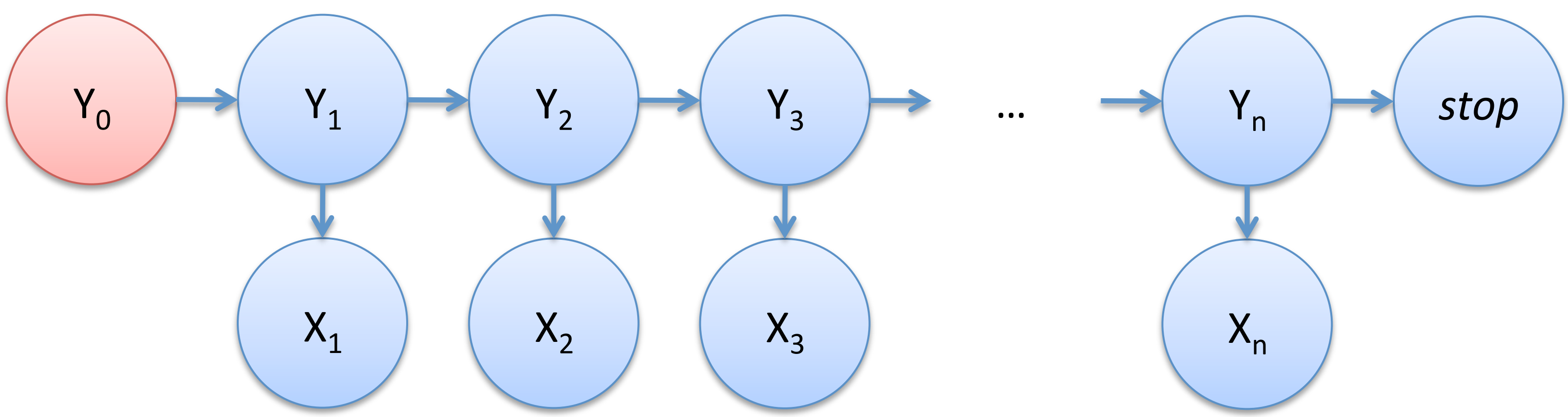
$$p(\boldsymbol{X} = \boldsymbol{x}, \boldsymbol{Y} = \boldsymbol{y}) \quad = \quad \left(\prod_{i=1}^{n} p(x_i \mid y_i) p(y_i \mid y_{i-1}) \right) p(stop \mid y_n)$$

- 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.

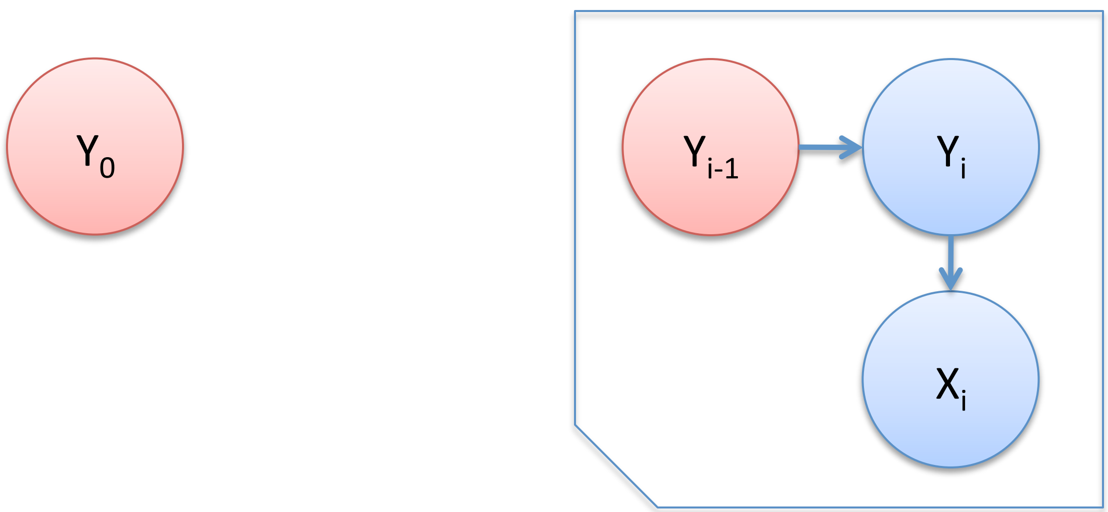
$$p(\boldsymbol{X} = \boldsymbol{x}, \boldsymbol{Y} = \boldsymbol{y}) \quad = \quad \left(\prod_{i=1}^{n} p(x_i \mid y_i) p(y_i \mid y_{i-1}) \right) p(stop \mid y_n)$$



Hidden Markov Model

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

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

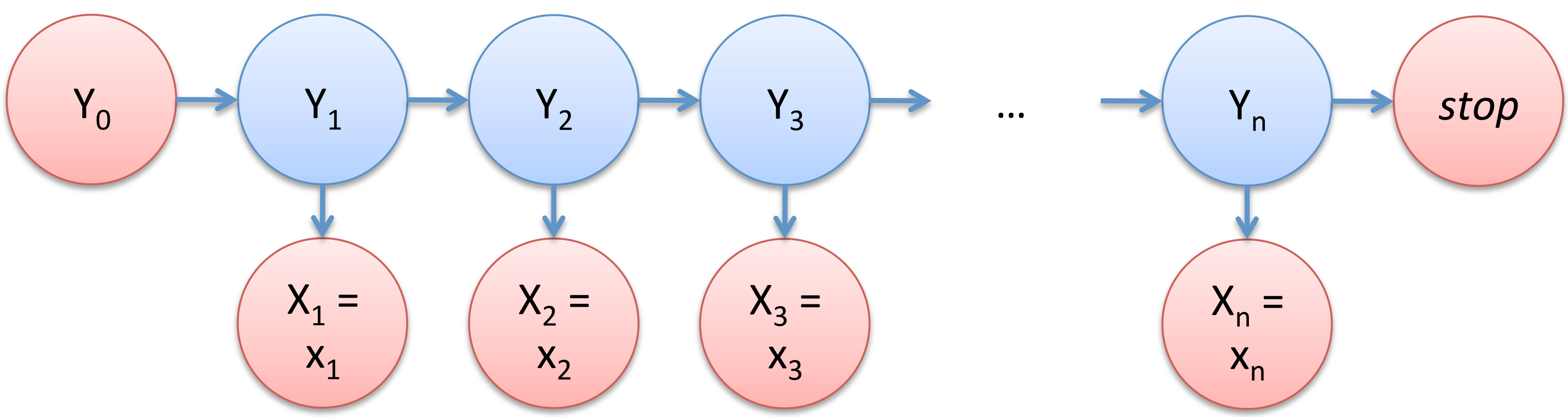


template that gets copied as many times as needed

Hidden Markov Model

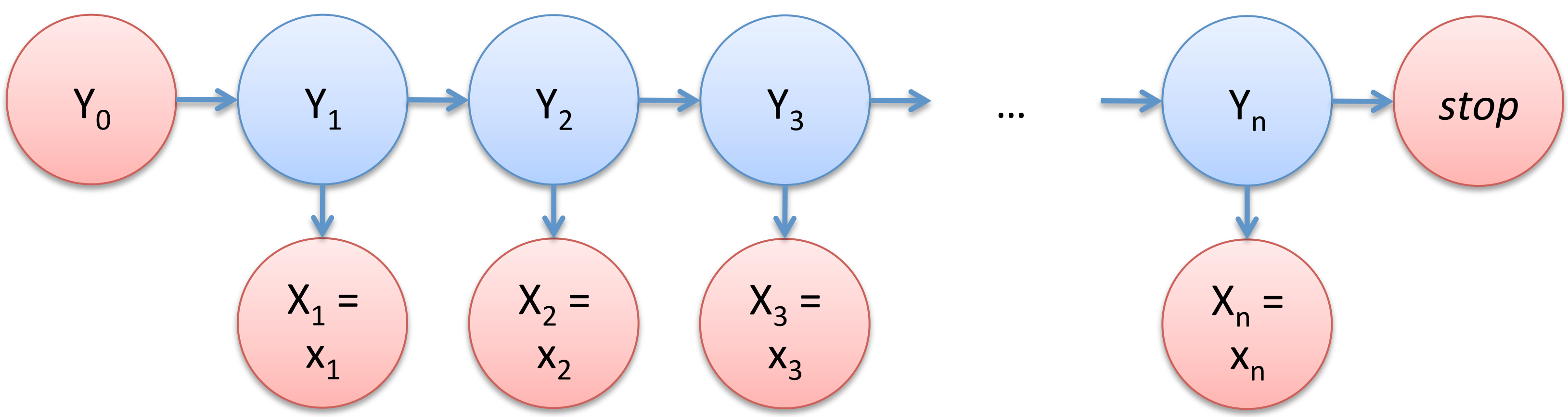
- Given **X**' s value as evidence, the dynamic part becomes unnecessary, since we know n.

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



Hidden Markov Model

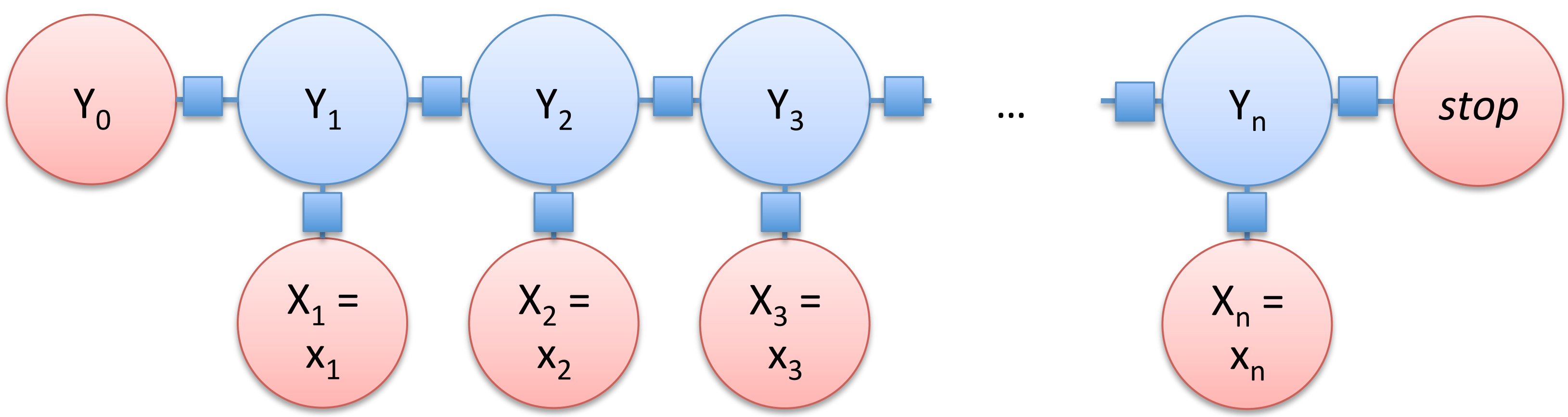
- The usual inference problem is to find the most probable value of **Y** given **X** = **x**.



Hidden Markov Model

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

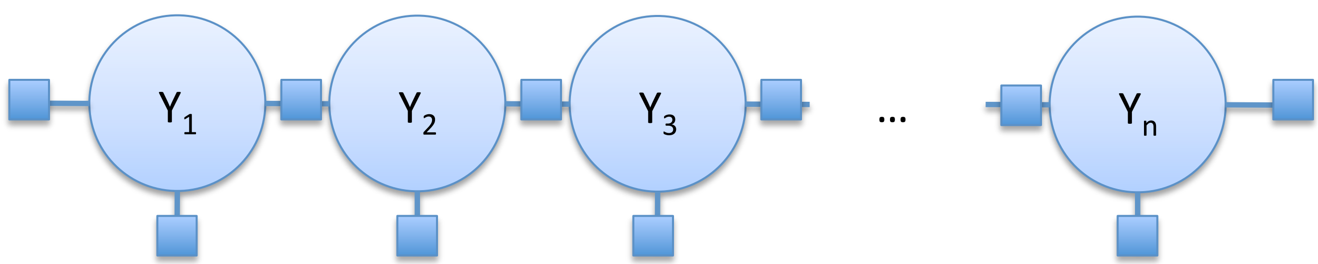
- Factor graph:



Hidden Markov Model

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

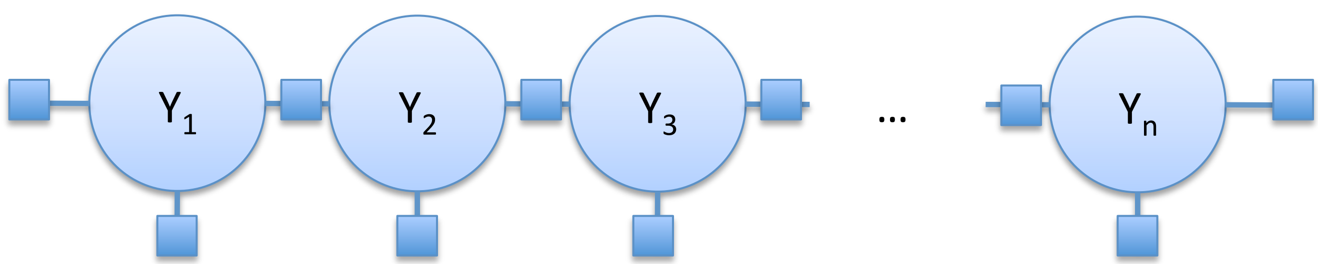
- Factor graph after reducing factors to respect evidence:



Hidden Markov Model

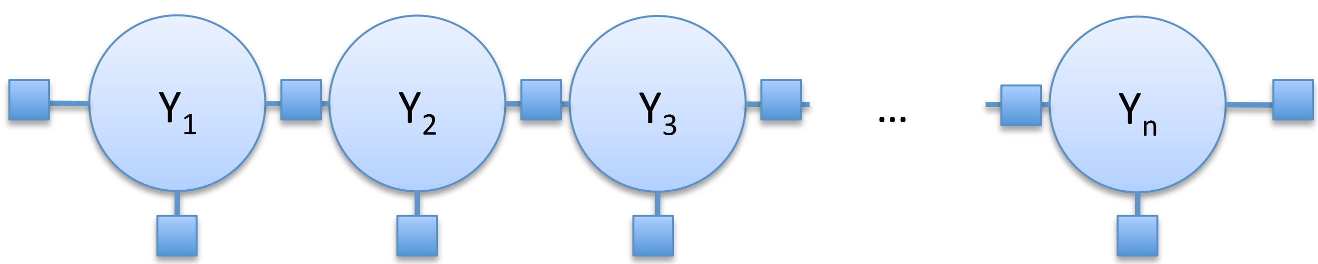
- The usual inference problem is to find the most probable value of **Y** given **X** = **x**.

- Clever ordering should be apparent!



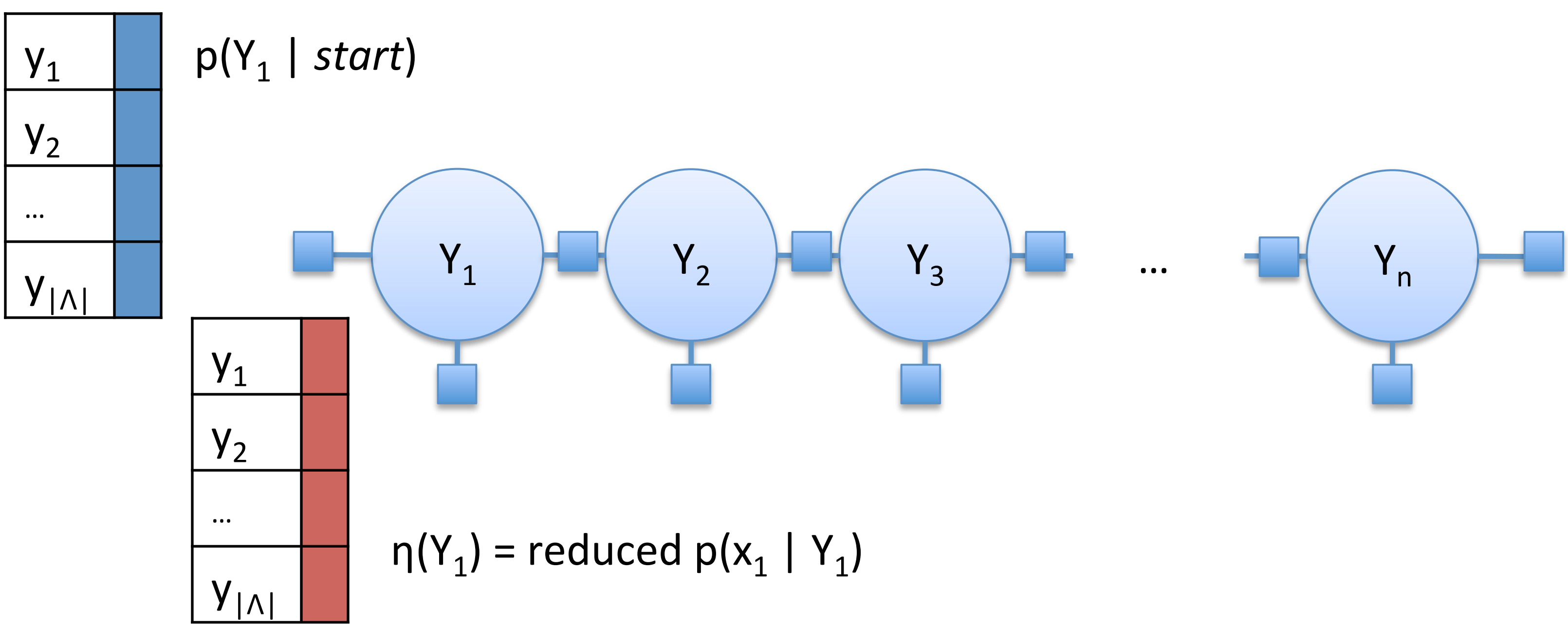
Hidden Markov Model

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



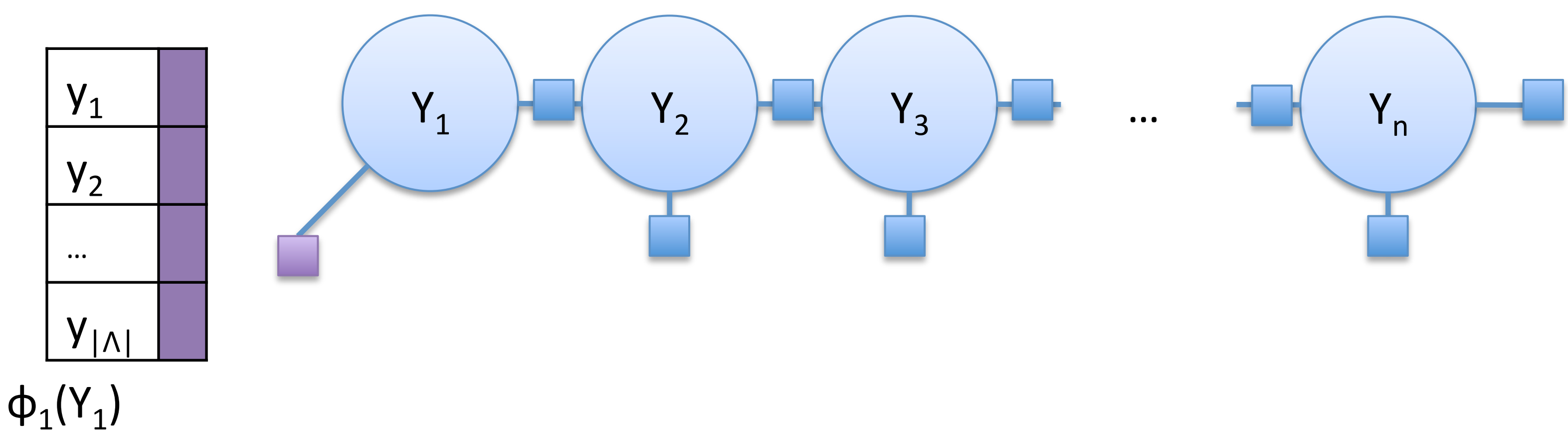
Hidden Markov Model

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



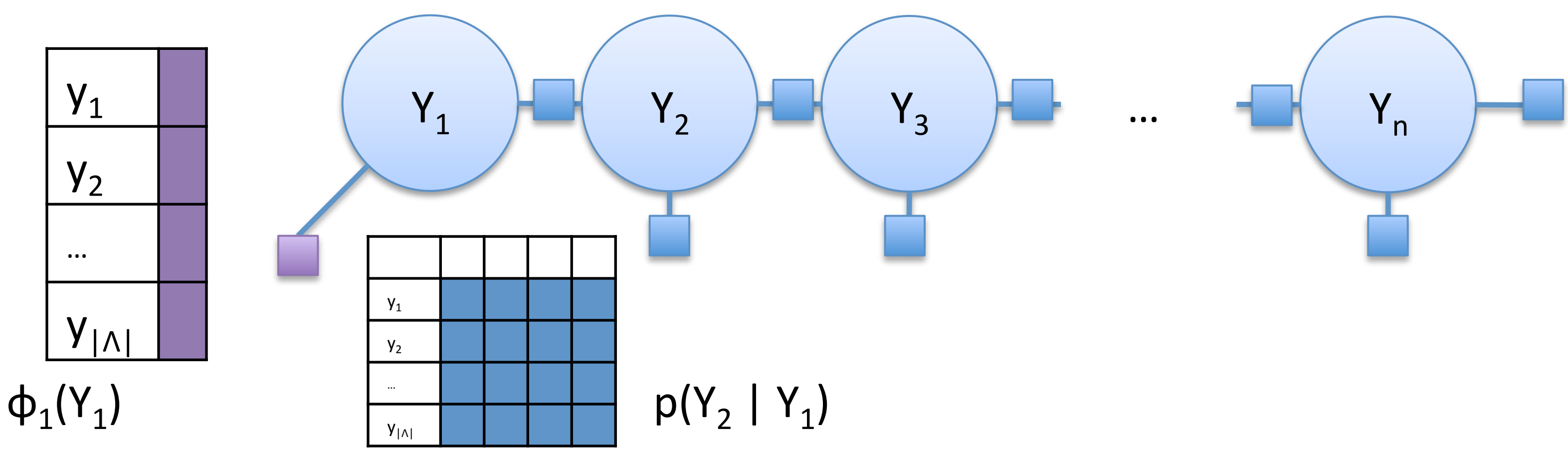
Hidden Markov Model

- 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.



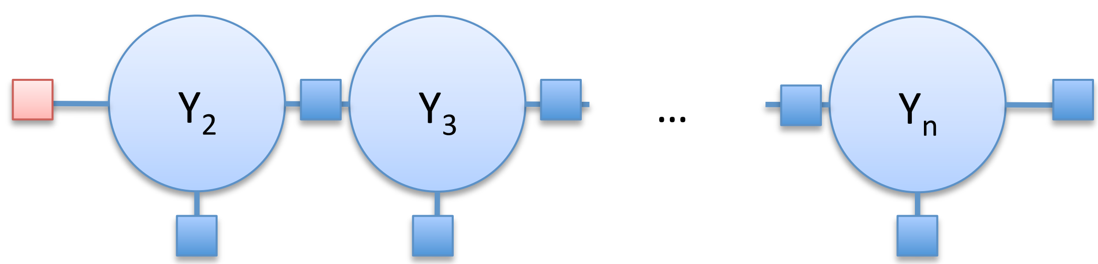
Hidden Markov Model

- 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.



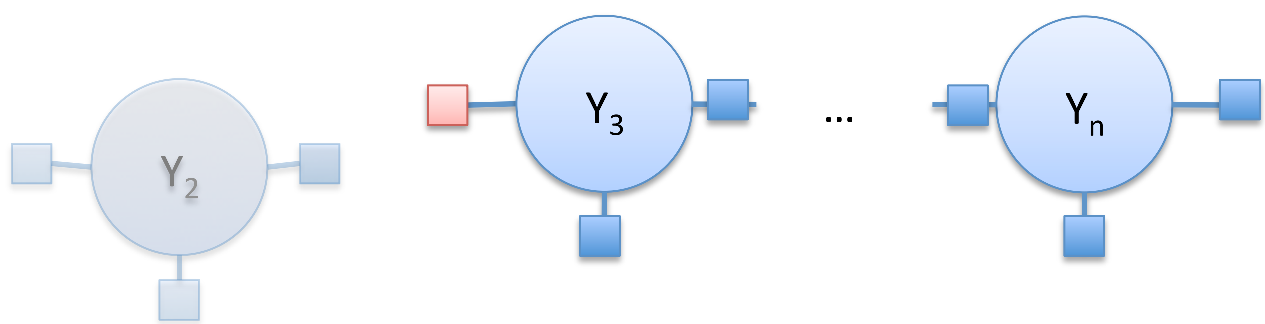
Hidden Markov Model

- 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.



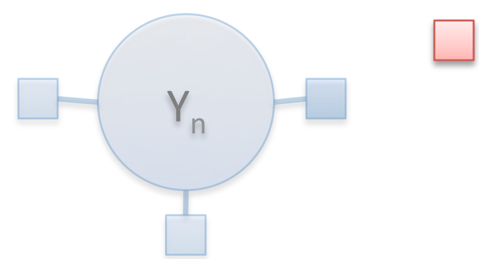
Hidden Markov Model

- 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

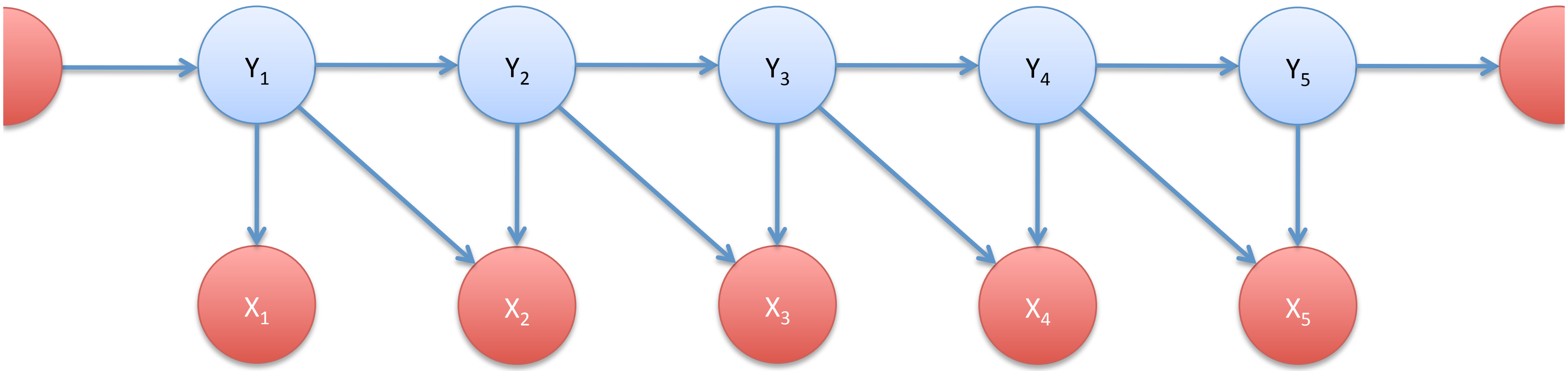
- At the end, we have one final factor with one row, ϕ_{n+1}.
- This is the score of the best sequence.
- Use backtrace to recover values.



Why Think This Way?

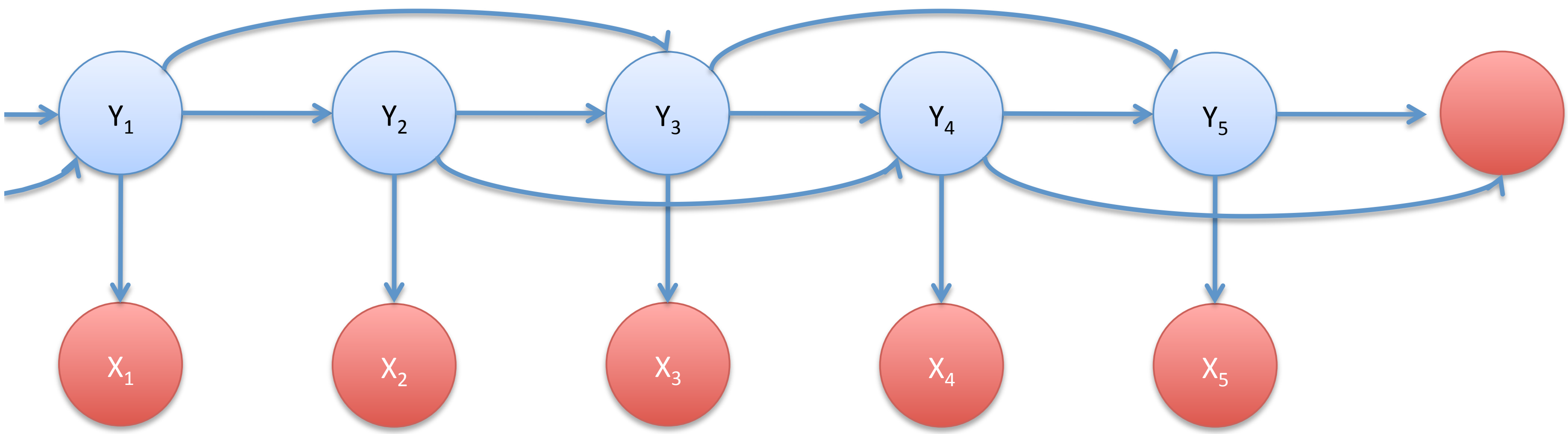
- Easy to see how to generalize HMMs.
  - More evidence
  - More factors
  - More hidden structure
  - More dependencies
- Probabilistic interpretation of factors is *not* central to finding the "best" **Y** …
  - Many factors are not conditional probability tables.

Generalization Example 1



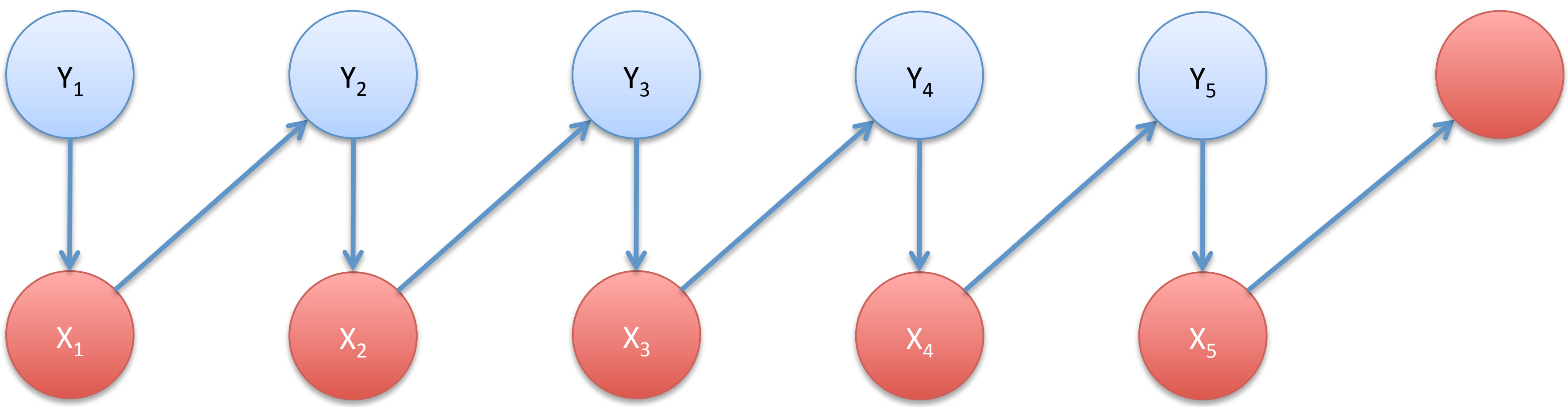
- Each word also depends on previous state.

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. $x_i = \mathit{Britain}$ and $y_i = \mathsf{B}$	1	0
	count of i s.t. $x_i = \mathit{Britain}$ and $y_i = \mathsf{I}$	0	0
	count of i s.t. $x_i = \mathit{Britain}$ and $y_i = \mathsf{O}$	0	1
downcased:	count of i s.t. $lc(x_i) = \mathit{britain}$ and $y_i = \mathsf{B}$	1	0
	count of i s.t. $lc(x_i) = \mathit{britain}$ and $y_i = \mathsf{I}$	0	0
	count of i s.t. $lc(x_i) = \mathit{britain}$ and $y_i = \mathsf{O}$	0	1
	count of i s.t. $lc(x_i) = \mathit{sent}$ and $y_i = \mathsf{O}$	1	1
	count of i s.t. $lc(x_i) = \mathit{warships}$ and $y_i = \mathsf{O}$	1	1
shape:	count of i s.t. $\mathit{shape}(x_i) = \mathit{Aaaaaaa}$ and $y_i = \mathsf{B}$	3	2
	count of i s.t. $\mathit{shape}(x_i) = \mathit{Aaaaaaa}$ and $y_i = \mathsf{I}$	1	1
	count of i s.t. $\mathit{shape}(x_i) = \mathit{Aaaaaaa}$ and $y_i = \mathsf{O}$	0	1
prefix:	count of i s.t. $\mathit{pre}_1(x_i) = B$ and $y_i = \mathsf{B}$	2	1
	count of i s.t. $\mathit{pre}_1(x_i) = B$ and $y_i = \mathsf{I}$	0	0
	count of i s.t. $\mathit{pre}_1(x_i) = B$ and $y_i = \mathsf{O}$	0	1
	count of i s.t. $\mathit{pre}_1(x_i) = s$ and $y_i = \mathsf{O}$	2	2
	count of i s.t. $\mathit{shape}(\mathit{pre}_1(x_i)) = A$ and $y_i = \mathsf{B}$	5	4
	count of i s.t. $\mathit{shape}(\mathit{pre}_1(x_i)) = A$ and $y_i = \mathsf{I}$	1	1
	count of i s.t. $\mathit{shape}(\mathit{pre}_1(x_i)) = A$ and $y_i = \mathsf{O}$	0	1
	$[\![\mathit{shape}(\mathit{pre}_1(x_1)) = A \wedge y_1 = \mathsf{B}]\!]$	1	0
	$[\![\mathit{shape}(\mathit{pre}_1(x_1)) = A \wedge y_1 = \mathsf{O}]\!]$	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. $x_i = \mathit{sent}$ 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:

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

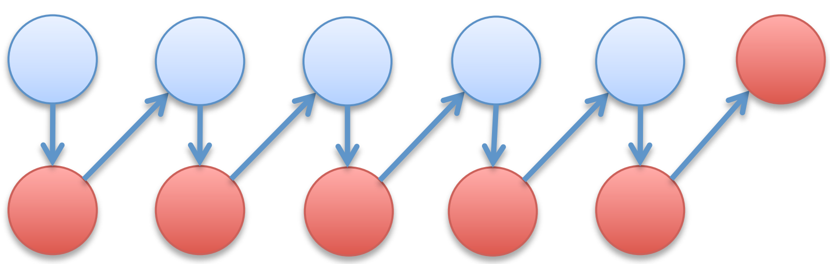
- Markov network:

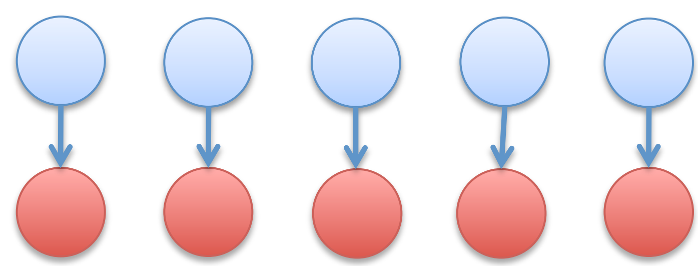
$$\sum_C \log \phi_C \left(\{x_i\}_{i \in C}, \{y_j\}_{j \in C}\right)$$

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

Inference in Graphical Models

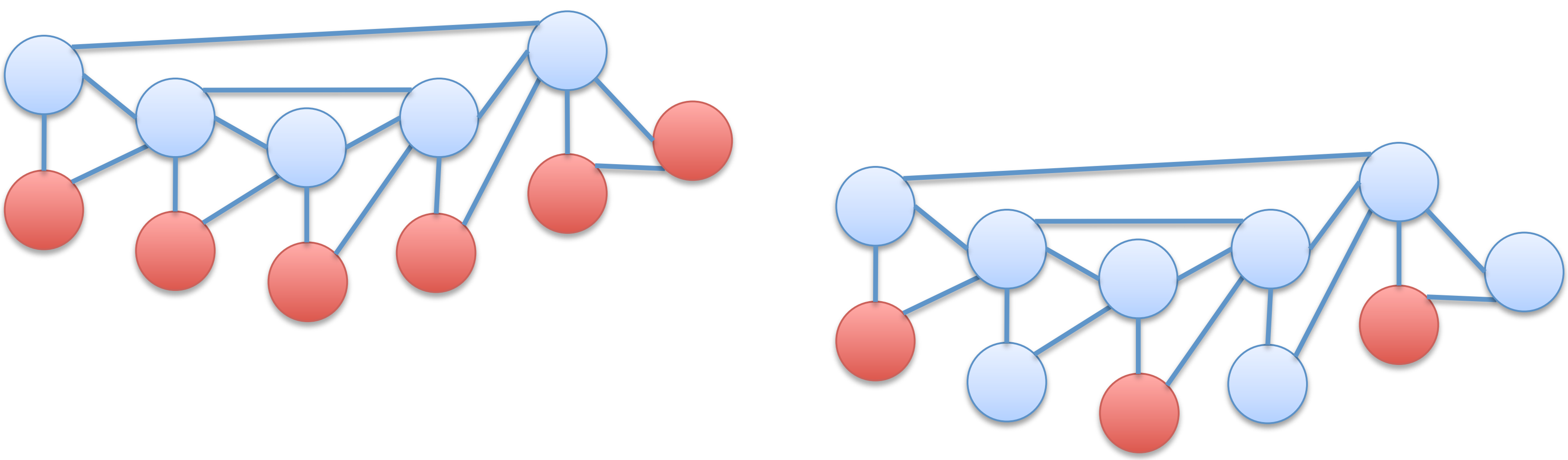
- Remember: more edges make inference more expensive.
  - Fewer edges means stronger independence.
- Really pleasant:





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.

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

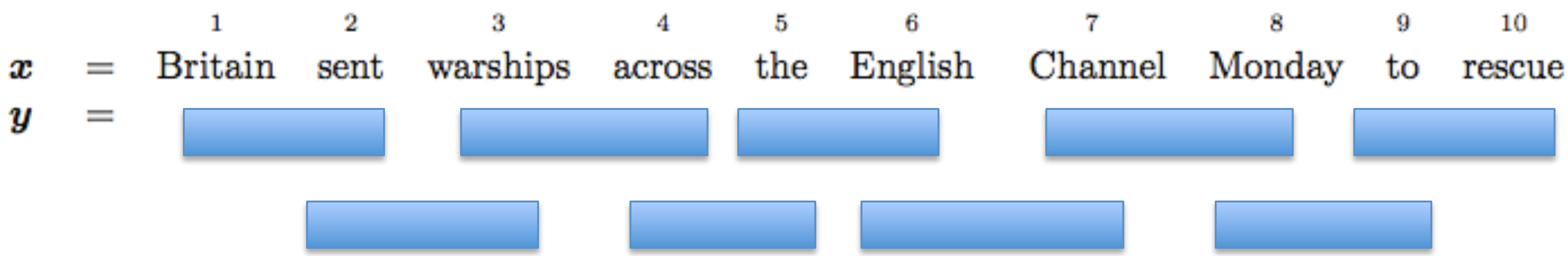
- Each part has an alphabet of possible values.
  – Decoding is choosing values for all parts, with consistency constraints.
  – (In the graphical models view, a part is a clique.)

Example

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

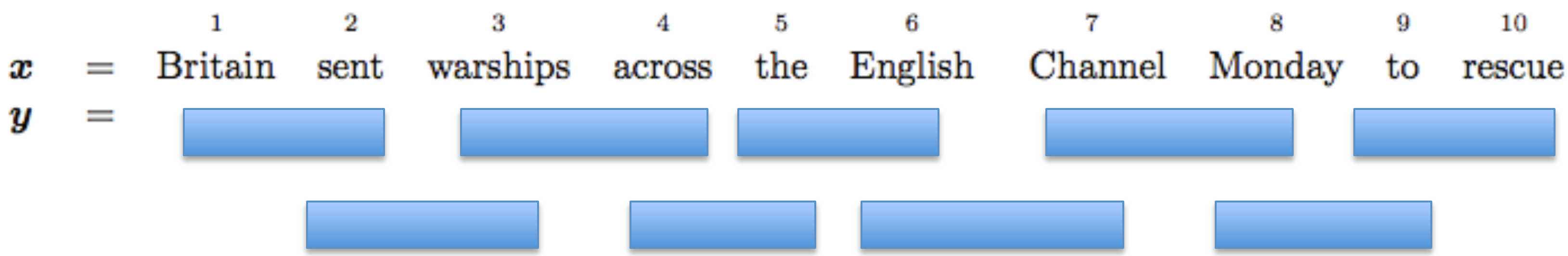
- One part per word, each is in $\{B, I, O\}$
- No features look at multiple parts
  - Fast inference
  - Not very expressive

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



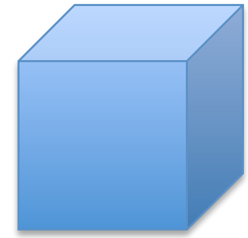
- Let $z_{i,\boldsymbol{\pi}}$ be 1 if part *i* takes value **π** and 0 otherwise.
- **z** is a vector in $\{0, 1\}^N$
  - N = total number of localized part values
  - Each **z** is a vertex of the unit cube

Score is Linear in **z**

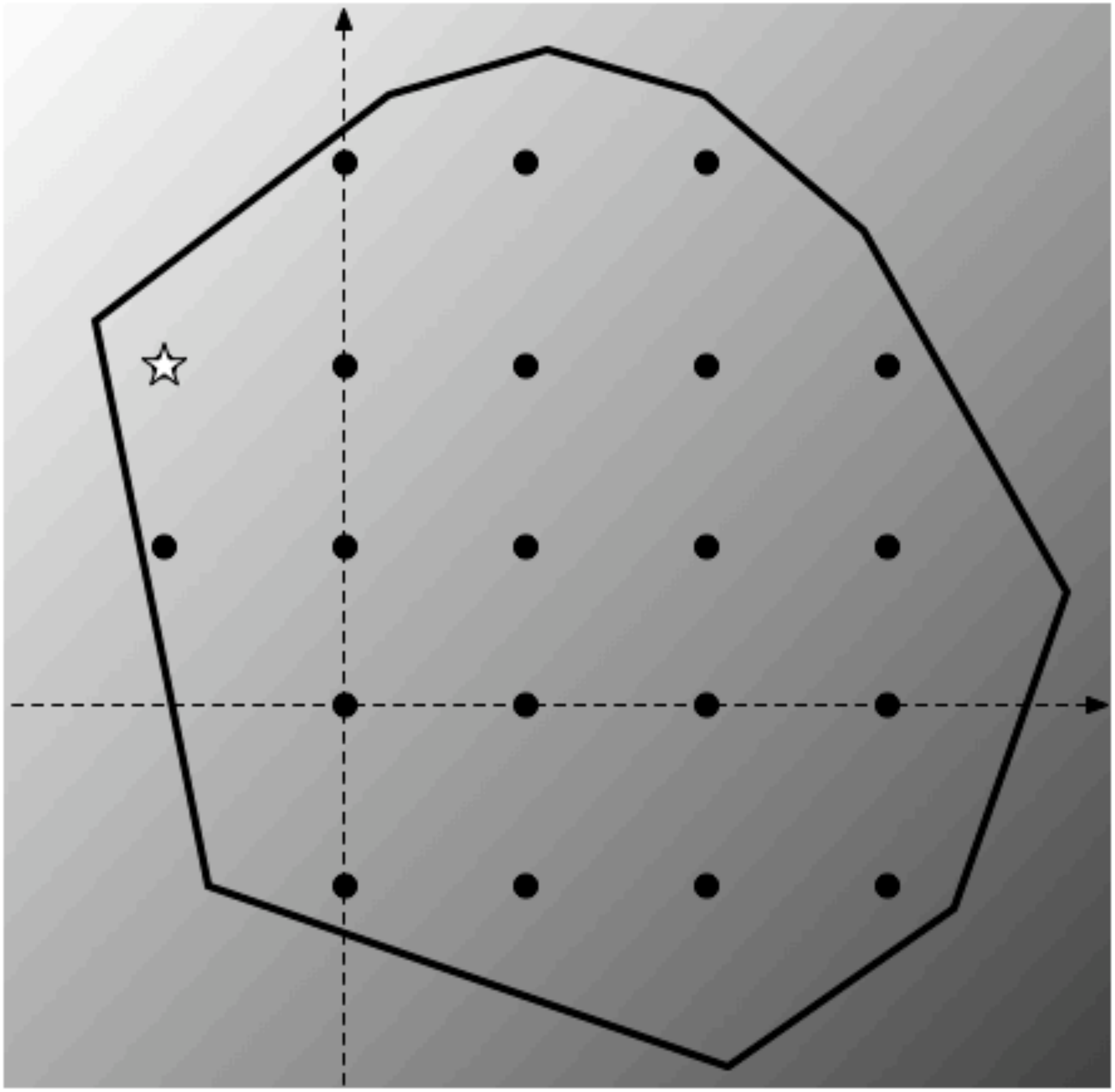
not really equal; need to transform back to get **y**

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

Polyhedra

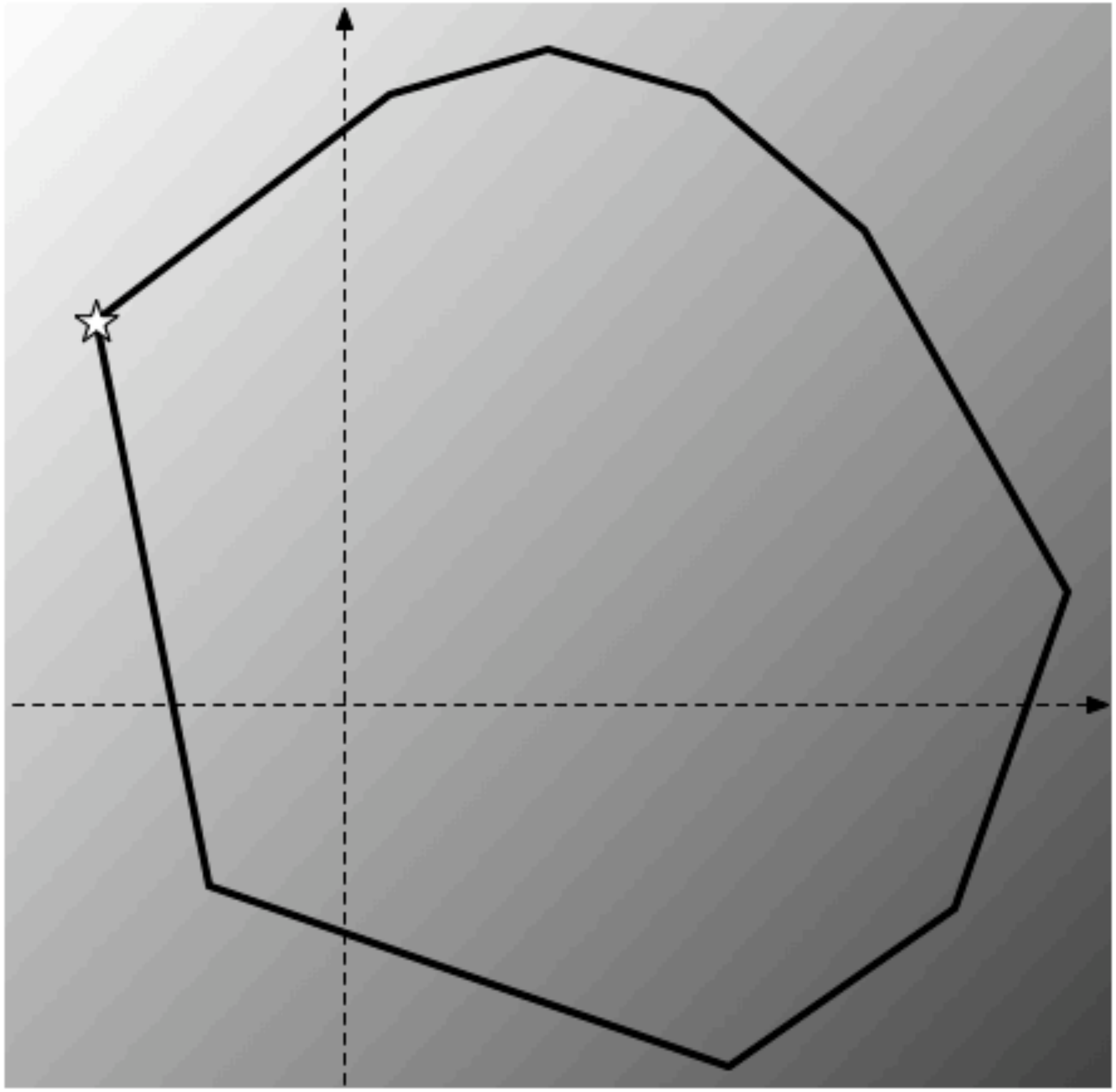


- Not all vertices of the *N*-dimensional unit cube satisfy the constraints.
  - E.g., can't have $z_{1,BI} = 1$ and $z_{2,BI} = 1$
- Sometimes we can write down a small (polynomial number) of linear constraints on **z**.
- Result: linear objective, linear constraints, integer constraints …



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.