

# Special Topics: Structured Prediction

Karl Stratos

June 22, 2018

## Motivation: Part-of-Speech (POS) Tagging

**Task.** Given a sentence, output a sequence of POS tags.

Vocabulary  $V$ , set of POS tags  $L$

$$V = \{\text{prim, that, Arya, fastidiously, 1988, ...}\}$$
$$L = \{\text{DT, NN, VBD, JJ, ...}\}$$

## Motivation: Part-of-Speech (POS) Tagging

**Task.** Given a sentence, output a sequence of POS tags.

Vocabulary  $V$ , set of POS tags  $L$

$$V = \{\text{prim, that, Arya, fastidiously, 1988, ...}\}$$
$$L = \{\text{DT, NN, VBD, JJ, ...}\}$$

For a sentence  $\mathbf{x} = (x_1 \dots x_m) \in V^N$  of length  $m$ , define the set of possible POS sequences

$$\mathbf{GEN}(\mathbf{x}) = \{(y_1 \dots y_m) \in L^m\}$$

How many possible POS sequences are there?

# Exponentially Many Possible Structures

$$|L|^m$$

the/DT dog/DT saw/DT the/DT cat/DT  
the/DT dog/DT saw/DT the/DT cat/NN  
the/DT dog/DT saw/DT the/NN cat/DT  
the/DT dog/DT saw/NN the/DT cat/DT  
the/DT dog/NN saw/DT the/DT cat/DT  
the/NN dog/DT saw/DT the/DT cat/DT  
the/DT dog/DT saw/DT the/DT cat/VBD  
the/DT dog/DT saw/DT the/VBD cat/DT  
the/DT dog/DT saw/VBD the/DT cat/DT  
the/DT dog/VBD saw/DT the/DT cat/DT  
the/VBD dog/DT saw/DT the/DT cat/DT

⋮

## Exponentially Many Possible Structures

If  $|L| = 45$ , then

<b>N</b>	1	2	3	4	...	50
<b>GEN</b> ( $x$ )	45	2025	91125	4100625	...	$4.6e+82$

# Exponentially Many Possible Structures

If  $|L| = 45$ , then

$N$	1	2	3	4	...	50
$ \text{GEN}(x) $	45	2025	91125	4100625	...	$4.6e+82$

🔍

[All](#) [News](#) [Images](#) [Videos](#) [Maps](#) [More](#) [Settings](#) [Tools](#)

About 21,000,000 results (0.62 seconds)

At this level, it is estimated that there are between  $10^{78}$  to  $10^{82}$  atoms in the known, observable universe. In layman's terms, that works out to between ten quadrillion vigintillion and one-hundred thousand quadrillion vigintillion atoms. Jul 30, 2009

**COSMIC EPOCHS**

13.7 billion years ago: Big Bang

100 million years: Dark Ages

400 million years: First Stars

1 billion years: First Galaxies

3 billion years: First Planets

4.5 billion years: Earth forms

65 million years: Dinosaurs

100 years: Modern Humans

[How Many Atoms Are There in the Universe? - Universe Today](https://www.universetoday.com/36302/atoms-in-the-universe/)  
<https://www.universetoday.com/36302/atoms-in-the-universe/>

## Structured Prediction

- ▶ The goal of structured prediction is to navigate an **exponentially large search space** of possible label structures.
- ▶ This is done by either imposing **structural constraints** or by **approximation**.
- ▶ Today we will use hidden Markov models (HMMs) to illustrate some of these concepts.
- ▶ The techniques generalize to models beyond HMMs, including conditional random fields (CRFs).

# Overview

## Derivation of an HMM

Parameter Estimation from Labeled Data

Computation with an HMM

- Marginalization and Inference

- Forward Algorithm

- Viterbi Algorithm

- Practical Issues

Beam Search



## Sequence Labeling with a Probabilistic Model

Vocabulary  $V$ , set of POS tags  $L$

$$V = \{\text{prim, that, Arya, fastidiously, 1988, ...}\}$$
$$L = \{\text{DT, NN, VBD, JJ, ...}\}$$

## Sequence Labeling with a Probabilistic Model

Vocabulary  $V$ , set of POS tags  $L$

$$V = \{\text{prim, that, Arya, fastidiously, 1988, ...}\}$$
$$L = \{\text{DT, NN, VBD, JJ, ...}\}$$

Want to define a **joint** distribution  $p(x_1 \dots x_m, y_1 \dots y_m)$  over

1. Any sentence  $x_1 \dots x_m \in V^m$
2. A corresponding sequence of POS tags  $y_1 \dots y_m \in L^m$

## Sequence Labeling with a Probabilistic Model

Vocabulary  $V$ , set of POS tags  $L$

$$V = \{\text{prim, that, Arya, fastidiously, 1988, ...}\}$$
$$L = \{\text{DT, NN, VBD, JJ, ...}\}$$

Want to define a **joint** distribution  $p(x_1 \dots x_m, y_1 \dots y_m)$  over

1. Any sentence  $x_1 \dots x_m \in V^m$
2. A corresponding sequence of POS tags  $y_1 \dots y_m \in L^m$

Why? Then we can infer for any given  $x_1 \dots x_m$

$$\begin{aligned} y_1^* \dots y_m^* &= \arg \max_{y_1 \dots y_m \in L^m} p(y_1 \dots y_m | x_1 \dots x_m) \\ &= \arg \max_{y_1 \dots y_m \in L^m} p(x_1 \dots x_m, y_1 \dots y_m) \end{aligned}$$

## A Left-to-Right Generative Process

By the chain rule, we may assume that

$$\begin{aligned} & p(x_1 \dots x_m, y_1 \dots y_m) \\ &= p(y_1) \times p(x_1|y_1) \times p(y_2|x_1, y_1) \times p(x_2|x_1, y_1, y_2) \cdots \\ &\quad \times p(y_m | \{x_i, y_i\}_{i=1}^{m-1}) \times p(x_m | \{x_i, y_i\}_{i=1}^{m-1}, y_m) \\ &\quad \times p(\text{STOP} | \{x_i, y_i\}_{i=1}^m) \end{aligned}$$

## A Left-to-Right Generative Process

By the chain rule, we may assume that

$$\begin{aligned} p(x_1 \dots x_m, y_1 \dots y_m) \\ &= p(y_1) \times p(x_1|y_1) \times p(y_2|x_1, y_1) \times p(x_2|x_1, y_1, y_2) \cdots \\ &\quad \times p(y_m | \{x_i, y_i\}_{i=1}^{m-1}) \times p(x_m | \{x_i, y_i\}_{i=1}^{m-1}, y_m) \\ &\quad \times p(\text{STOP} | \{x_i, y_i\}_{i=1}^m) \end{aligned}$$

Design a tractable model by making **independence assumptions**.

- ▶ What kind of assumption is reasonable for POS tagging?

# First-Order HMM Assumptions

1. At any position  $i$ , the word depends on the current tag only.

$$p(x_i | \{x_j, y_j\}_{j=1}^{i-1}, y_i) = p(x_i | y_i)$$

2. At any position  $i$ , the tag depends on the previous tag only.

$$p(y_i | \{x_j, y_j\}_{j=1}^{i-1}) = p(y_i | y_{i-1})$$

## Model Parameters

- ▶  $|V| \times |L|$  “emission” probabilities

$o(x|y)$  = probability of emitting word  $x$  given tag  $y$

- ▶  $|L|^2 + 2|L|$  “transition” probabilities

$t(y'|y)$  = probability of transitioning from tag  $y$  to  $y'$

$t(y|*)$  = probability of starting with tag  $y$

$t(\text{STOP}|y)$  = probability of ending with tag  $y$

Used to calculate

$$p(x_1 \dots x_m, y_1 \dots y_m) = \prod_{i=1}^{m+1} t(y_i|y_{i-1}) \times \prod_{i=1}^m o(x_i|y_i)$$

where  $y_0 = *$  and  $y_{m+1} = \text{STOP}$  are special symbols.

# Overview

Derivation of an HMM

Parameter Estimation from Labeled Data

Computation with an HMM

- Marginalization and Inference

- Forward Algorithm

- Viterbi Algorithm

- Practical Issues

Beam Search



## Labeled Data

- ▶ Consists of  $N$  annotated sentences  $(x^{(1)}, y^{(1)}) \dots (x^{(N)}, y^{(N)})$  where  $l_i = |x^{(i)}| = |y^{(i)}|$  and  $y_0^{(i)} = *$ ,  $y_{l_i+1}^{(i)} = \text{STOP}$ .
- ▶ Define **count** $(y, y')$  for  $y, y' \in L \cup \{*, \text{STOP}\}$ :

$$\mathbf{count}(y, y') = \sum_{i=1}^N \sum_{\substack{j=1: \\ y_{j-1}^{(i)}=y \\ y_j^{(i)}=y'}}^{l_i+1} 1$$

- ▶ Define **count** $(x, y)$  for  $x \in V$ ,  $y \in L$ :

$$\mathbf{count}(x, y) = \sum_{i=1}^N \sum_{\substack{j=1: \\ x_j^{(i)}=x \\ y_j^{(i)}=y}}^{l_i} 1$$

## Parameter Estimation

- ▶ For all  $y, y'$  with  $\mathbf{count}(y, y') > 0$ , set

$$t(y'|y) = \frac{\mathbf{count}(y, y')}{\sum_{y' \in L} \mathbf{count}(y, y')}$$

Otherwise  $t(y'|y) = 0$ .

## Parameter Estimation

- ▶ For all  $y, y'$  with  $\mathbf{count}(y, y') > 0$ , set

$$t(y'|y) = \frac{\mathbf{count}(y, y')}{\sum_{y' \in L} \mathbf{count}(y, y')}$$

Otherwise  $t(y'|y) = 0$ .

- ▶ For all  $x, y$  with  $\mathbf{count}(x, y) > 0$ , set

$$o(x|y) = \frac{\mathbf{count}(x, y)}{\sum_{x \in V} \mathbf{count}(x, y)}$$

Otherwise  $o(x|y) = 0$ .

## Justification

**Claim.** The solution of

$$o^*, t^* = \underset{\substack{o, t: o(x|y), t(y'|y) \geq 0 \\ \sum_{y'} t(y'|y) = \sum_x o(x|y) = 1}}{\arg \max} \sum_{i=1}^N \log p(x^{(i)}, y^{(i)})$$

where  $p(x, y)$  is the distribution of an HMM is given by

$$o^*(x|y) = \frac{\mathbf{count}(x, y)}{\sum_x \mathbf{count}(x, y)} \quad t^*(y'|y) = \frac{\mathbf{count}(y, y')}{\sum_{y'} \mathbf{count}(y, y')}$$

# Overview

Derivation of an HMM

Parameter Estimation from Labeled Data

Computation with an HMM

Marginalization and Inference

Forward Algorithm

Viterbi Algorithm

Practical Issues

Beam Search

## Setting

- ▶ We now assume that we have parameters  $o(x|y)$  and  $t(y'|y)$ .

## Setting

- ▶ We now assume that we have parameters  $o(x|y)$  and  $t(y'|y)$ .
- ▶ They define the joint probability distribution

$$p(x_1 \dots x_m, y_1 \dots y_m) = \prod_{i=1}^{m+1} t(y_i|y_{i-1}) \times \prod_{i=1}^m o(x_i|y_i)$$

over any words  $x_1 \dots x_m \in V^m$  and POS tags  $y_1 \dots y_m \in L^m$ .

## Setting

- ▶ We now assume that we have parameters  $o(x|y)$  and  $t(y'|y)$ .
- ▶ They define the joint probability distribution

$$p(x_1 \dots x_m, y_1 \dots y_m) = \prod_{i=1}^{m+1} t(y_i|y_{i-1}) \times \prod_{i=1}^m o(x_i|y_i)$$

over any words  $x_1 \dots x_m \in V^m$  and POS tags  $y_1 \dots y_m \in L^m$ .

- ▶ **Given a fixed sentence**  $x_1 \dots x_m \in V^m$ , we often wish to perform two critical calculations (next slide).



## Marginalization and Inference

1. What is the probability of  $x_1 \dots x_m$  under the HMM?

$$\sum_{y_1 \dots y_m \in L^m} p(x_1 \dots x_m, y_1 \dots y_m)$$

## Marginalization and Inference

1. What is the probability of  $x_1 \dots x_m$  under the HMM?

$$\sum_{y_1 \dots y_m \in L^m} p(x_1 \dots x_m, y_1 \dots y_m)$$

2. What is the most probable  $y_1 \dots y_m \in L^m$  under the HMM?

$$\arg \max_{y_1 \dots y_m \in L^m} p(x_1 \dots x_m, y_1 \dots y_m)$$

# Number of Possible Tag Sequences

- ▶ **Exponential in the length of the sentence**
- ▶ Enumerating all  $|L|^m$  candidates is clearly not practical.
- ▶ We will exploit the HMM assumptions to perform marginalization/inference **exactly** and with **polynomial complexity**.

# Overview

Derivation of an HMM

Parameter Estimation from Labeled Data

Computation with an HMM

- Marginalization and Inference

  - Forward Algorithm

  - Viterbi Algorithm

  - Practical Issues

Beam Search

## Left-to-Right Incremental Marginalization

- ▶ **Idea.** No need to consider all  $|L|^m$  candidates because of the left-to-right generative process and independence assumptions under the HMM

## Left-to-Right Incremental Marginalization

- ▶ **Idea.** No need to consider all  $|L|^m$  candidates because of the left-to-right generative process and independence assumptions under the HMM
- ▶ **Forward algorithm.** For  $i = 1 \dots m$ , for all  $y \in L$ ,

$$\pi(i, y) := \sum_{y_1 \dots y_i \in L^i: y_i = y} p(x_1 \dots x_i, y_1 \dots y_i)$$

We will see that computing each  $\pi(i, y)$  takes  $O(|L|)$  time using **dynamic programming**.

## Left-to-Right Incremental Marginalization

- ▶ **Idea.** No need to consider all  $|L|^m$  candidates because of the left-to-right generative process and independence assumptions under the HMM
- ▶ **Forward algorithm.** For  $i = 1 \dots m$ , for all  $y \in L$ ,

$$\pi(i, y) := \sum_{y_1 \dots y_i \in L^i: y_i = y} p(x_1 \dots x_i, y_1 \dots y_i)$$

We will see that computing each  $\pi(i, y)$  takes  $O(|L|)$  time using **dynamic programming**.

- ▶ Total runtime?

## Base Case ( $i = 1$ )

$$\begin{aligned}\pi(1, y) &:= \sum_{y_1 \in L: y_1=y} p(x_1, y_1) \\ &= t(y|*) \times o(x_1|y)\end{aligned}$$



## Main Body ( $i > 1$ )

$$\pi(i, y') := \sum_{y_1 \dots y_i: y_i = y'} p(x_1 \dots x_i, y_1 \dots y_i)$$

## Main Body ( $i > 1$ )

$$\begin{aligned}\pi(i, \mathbf{y}') &:= \sum_{y_1 \dots y_i: y_i = \mathbf{y}'} p(x_1 \dots x_i, y_1 \dots y_i) \\ &= \sum_{y_1} \dots \sum_{y_{i-1}} p(x_1 \dots x_i, y_1 \dots y_{i-1} \mathbf{y}')\end{aligned}$$

## Main Body ( $i > 1$ )

$$\begin{aligned}\pi(i, \mathbf{y}') &:= \sum_{y_1 \dots y_i: y_i = \mathbf{y}'} p(x_1 \dots x_i, y_1 \dots y_i) \\ &= \sum_{y_1} \dots \sum_{y_{i-1}} p(x_1 \dots x_i, y_1 \dots y_{i-1}, \mathbf{y}') \\ &= \sum_{y_1} \dots \sum_{y_{i-1}} p(x_1 \dots x_{i-1}, y_1 \dots y_{i-1}) \times t(\mathbf{y}' | y_{i-1}) \times o(x_i | \mathbf{y}')\end{aligned}$$

## Main Body ( $i > 1$ )

$$\begin{aligned}\pi(i, \mathbf{y}') &:= \sum_{y_1 \dots y_i: y_i = \mathbf{y}'} p(x_1 \dots x_i, y_1 \dots y_i) \\ &= \sum_{y_1} \dots \sum_{y_{i-1}} p(x_1 \dots x_i, y_1 \dots y_{i-1} \mathbf{y}') \\ &= \sum_{y_1} \dots \sum_{y_{i-1}} p(x_1 \dots x_{i-1}, y_1 \dots y_{i-1}) \times t(\mathbf{y}' | y_{i-1}) \times o(x_i | \mathbf{y}') \\ &= \sum_{\mathbf{y}} \sum_{y_1 \dots y_{i-2}} p(x_1 \dots x_{i-1}, y_1 \dots y_{i-2} \mathbf{y}) \times t(\mathbf{y}' | \mathbf{y}) \times o(x_i | \mathbf{y}')\end{aligned}$$

## Main Body ( $i > 1$ )

$$\begin{aligned}\pi(i, \mathbf{y}') &:= \sum_{y_1 \dots y_i: y_i = \mathbf{y}'} p(x_1 \dots x_i, y_1 \dots y_i) \\ &= \sum_{y_1} \dots \sum_{y_{i-1}} p(x_1 \dots x_i, y_1 \dots y_{i-1} \mathbf{y}') \\ &= \sum_{y_1} \dots \sum_{y_{i-1}} p(x_1 \dots x_{i-1}, y_1 \dots y_{i-1}) \times t(\mathbf{y}' | y_{i-1}) \times o(x_i | \mathbf{y}') \\ &= \sum_{\mathbf{y}} \sum_{y_1 \dots y_{i-2}} p(x_1 \dots x_{i-1}, y_1 \dots y_{i-2} \mathbf{y}) \times t(\mathbf{y}' | \mathbf{y}) \times o(x_i | \mathbf{y}') \\ &= \sum_{\mathbf{y}} \pi(i-1, \mathbf{y}) \times t(\mathbf{y}' | \mathbf{y}) \times o(x_i | \mathbf{y}')\end{aligned}$$

## Final Marginalization

Obtain the probability of  $x_1 \dots x_m$  under the HMM by

$$\sum_{y_1 \dots y_m} p(x_1 \dots x_m, y_1 \dots y_m) = \sum_{y \in L} \pi(m, y)$$

# Overview

Derivation of an HMM

Parameter Estimation from Labeled Data

Computation with an HMM

- Marginalization and Inference

- Forward Algorithm

- Viterbi Algorithm

- Practical Issues

Beam Search

## Left-to-Right Incremental Maximization

- ▶ **Same Idea.** Use the properties of the HMM.



## Left-to-Right Incremental Maximization

- ▶ **Same Idea.** Use the properties of the HMM.
- ▶ **Viterbi algorithm.** For  $i = 1 \dots m$ , for all  $y \in L$ ,

$$\pi(i, y) := \max_{y_1 \dots y_i \in L^i: y_i = y} p(x_1 \dots x_i, y_1 \dots y_i)$$

## Left-to-Right Incremental Maximization

- ▶ **Same Idea.** Use the properties of the HMM.

- ▶ **Viterbi algorithm.** For  $i = 1 \dots m$ , for all  $y \in L$ ,

$$\pi(i, y) := \max_{y_1 \dots y_i \in L^i: y_i = y} p(x_1 \dots x_i, y_1 \dots y_i)$$

- ▶ The *only* difference from the forward alg: “ $\sum$ ”  $\mapsto$  “**max**”

$$\pi(1, y) = t(y|\ast) \times o(x_1|y)$$

$$\pi(i, y') = \max_{y \in L} \pi(i-1, y) \times t(y'|y) \times o(x_i|y')$$

## Left-to-Right Incremental Maximization

- ▶ **Same Idea.** Use the properties of the HMM.

- ▶ **Viterbi algorithm.** For  $i = 1 \dots m$ , for all  $y \in L$ ,

$$\pi(i, y) := \max_{y_1 \dots y_i \in L^i: y_i = y} p(x_1 \dots x_i, y_1 \dots y_i)$$

- ▶ The *only* difference from the forward alg: “ $\sum$ ”  $\mapsto$  “**max**”

$$\pi(1, y) = t(y|*) \times o(x_1|y)$$

$$\pi(i, y') = \max_{y \in L} \pi(i-1, y) \times t(y'|y) \times o(x_i|y')$$

- ▶ But how do we extract the actual **tag sequence**?

$$y_1^* \dots y_m^* = \arg \max_{y_1 \dots y_m \in L^m} p(x_1 \dots x_m, y_1 \dots y_m)$$

## Backtracking

- ▶ Keep an *additional* chart to record the **path**:

$$\beta(i, y') = \mathbf{arg\ max}_{y \in L} \pi(i - 1, y) \times t(y'|y) \times o(x_i|y')$$

for  $i = 2 \dots m$ .

## Backtracking

- ▶ Keep an *additional* chart to record the **path**:

$$\beta(i, y') = \mathbf{arg\ max}_{y \in L} \pi(i-1, y) \times t(y'|y) \times o(x_i|y')$$

for  $i = 2 \dots m$ .

- ▶ After running Viterbi, we can “backtrack”

$$y_m^* = \mathbf{arg\ max}_{y \in L} \pi(m, y)$$

$$y_{m-1}^* = \beta(m, y_m^*)$$

⋮

$$y_1^* = \beta(2, y_2^*)$$

and return  $y_1^* \dots y_m^*$ .

# Overview

Derivation of an HMM

Parameter Estimation from Labeled Data

Computation with an HMM

- Marginalization and Inference

- Forward Algorithm

- Viterbi Algorithm

- Practical Issues

Beam Search

## Log Space

- ▶ For numerical stability, always operate in **log space**.

## Log Space

- ▶ For numerical stability, always operate in **log space**.
- ▶ For Viterbi, it's a simple change:

$$\pi(1, y) = \log t(y|\star) + \log o(x_1|y)$$

$$\pi(i, y') = \max_{y \in L} \pi(i-1, y) + \log t(y'|y) + \log o(x_i|y')$$



## Log Space

- ▶ For numerical stability, always operate in **log space**.
- ▶ For Viterbi, it's a simple change:

$$\pi(1, y) = \log t(y|\ast) + \log o(x_1|y)$$

$$\pi(i, y') = \max_{y \in L} \pi(i-1, y) + \log t(y'|y) + \log o(x_i|y')$$

- ▶ For the forward algorithm, we need a helper function:

$$\text{logsum}(\log(c_1) \dots \log(c_n))$$

returns  $\log(c_1 + \dots + c_n)$  **without exponentiating**  $\log(c_i)$ !

# Log Space: Forward Algorithm

- ▶ Original:

$$\pi(1, y) = t(y|\ast) \times o(x_1|y)$$

$$\pi(i, y') = \sum_{y \in L} \pi(i-1, y) \times t(y'|y) \times o(x_i|y')$$

- ▶ Log space:

$$\pi(1, y) = \log t(y|\ast) + \log o(x_1|y)$$

$$\pi(i, y') = \operatorname{logsum}_{y \in L} \pi(i-1, y) + \log t(y'|y) + \log o(x_i|y')$$

## Trick to Sum Logs

**Input:**  $\log a \geq \log b$

**Output:**  $\log(a + b)$

- ▶ If  $\log a < -\infty$ : return  $-\infty$ .
- ▶ If  $\log b - \log a < -20$ : return  $\log a$ .
- ▶ If  $\log b - \log a \geq -20$ : return

$$\log a + \log(1 + \exp(\log b - \log a))$$

## Justification of the Trick

$$\begin{aligned}\log(a + b) &= \log\left(a \left(1 + \frac{b}{a}\right)\right) \\ &= \log(a) + \log\left(1 + \exp(\log b - \log a)\right)\end{aligned}$$

- ▶ Even if  $\exp(\log a)$  and  $\exp(\log b)$  underflow to zero,  $\exp(\log b - \log a)$  does not.

$$\log a = -99999$$

$$\log b = -100000$$

$$\log b - \log a = -1$$

# Debugging

- ▶ How do you debug the forward/Viterbi algorithm?
- ▶ The (only) surest check:
  1. Generate a small synthetic HMM, say with  $|V| = 10$ ,  $|L| = 5$ .
  2. Generate a short random sentence, say length 7.
  3. **Brute-force**: enumerate all  $5^7$  possible sequences for exact marginalization and inference.
  4. Run your forward/Viterbi.
  5. Make sure 4 is precisely the same as 3.
  6. Repeat 2–5 many times.

# Overview

Derivation of an HMM

Parameter Estimation from Labeled Data

Computation with an HMM

- Marginalization and Inference

- Forward Algorithm

- Viterbi Algorithm

- Practical Issues

Beam Search

# Heads-Up

- ▶ We will now talk about an extremely general technique called **beam search**.
  - ▶ Applicable to many models other than HMMs
  
- ▶ Possibly the most practical trick in structured prediction

## Score Function Under an HMM

- ▶ Given a fixed input sequence  $x = (x_1 \dots x_m)$ , an HMM defines the “score” of a candidate sequence  $y = (y_1 \dots y_m)$  as

$$\text{score}_x(y) = \prod_{i=1}^m \text{score}_x(y_i | y_1 \dots y_{i-1})$$

where each local score is **restricted** to only depend on the previous label  $y_{i-1}$  and current input  $x_i$ .

$$\text{score}_x(y_i | y_1 \dots y_{i-1}) := t(y_i | y_{i-1}) \times o(x_i | y_i)$$



## Score Function Under an HMM

- ▶ Given a fixed input sequence  $x = (x_1 \dots x_m)$ , an HMM defines the “score” of a candidate sequence  $y = (y_1 \dots y_m)$  as

$$\text{score}_x(y) = \prod_{i=1}^m \text{score}_x(y_i | y_1 \dots y_{i-1})$$

where each local score is **restricted** to only depend on the previous label  $y_{i-1}$  and current input  $x_i$ .

$$\text{score}_x(y_i | y_1 \dots y_{i-1}) := t(y_i | y_{i-1}) \times o(x_i | y_i)$$

- ▶ **With this restriction**, we can efficiently and exactly compute

$$\arg \max_{y_1 \dots y_m} \text{score}(y_1 \dots y_m) \quad (\text{Viterbi})$$

$$\sum_{y_1 \dots y_m} \text{score}(y_1 \dots y_m) \quad (\text{forward})$$

## General Score Function

- ▶ Now suppose we have a local score that can depend arbitrarily on **all previous labels**  $y_1 \dots y_{i-1}$ :

$$\text{score}_x(y_i | y_1 \dots y_{i-1}) = f(x_1 \dots x_m, y_1 \dots y_{i-1})$$

## General Score Function

- ▶ Now suppose we have a local score that can depend arbitrarily on **all previous labels**  $y_1 \dots y_{i-1}$ :

$$\text{score}_x(y_i | y_1 \dots y_{i-1}) = f(x_1 \dots x_m, y_1 \dots y_{i-1})$$

- ▶ Without any Markov assumption, we can't hope to do inference/marginalization efficiently and exactly.

## General Score Function

- ▶ Now suppose we have a local score that can depend arbitrarily on **all previous labels**  $y_1 \dots y_{i-1}$ :

$$\text{score}_x(y_i | y_1 \dots y_{i-1}) = f(x_1 \dots x_m, y_1 \dots y_{i-1})$$

- ▶ Without any Markov assumption, we can't hope to do inference/marginalization efficiently and exactly.
- ▶ But we can **approximate** it.

# Beam Search

- ▶ A hack to approximate a **set** of top- $K$  candidate sequences

$$\mathcal{B} \approx \underset{y_1 \dots y_m}{\text{K-argmax}} \text{score}_x(y_1 \dots y_m)$$

for **any** score function of the form

$$\text{score}_x(y) = \prod_{i=1}^m \text{score}_x(y_i | y_1 \dots y_{i-1})$$

# Uses of the Beam Search

- ▶ The best sequence can be approximated as

$$\arg \max_{(y_1 \dots y_m) \in \mathcal{B}} \text{score}(y_1 \dots y_m)$$

- ▶ The total score of all sequences can be approximated as

$$\sum_{(y_1 \dots y_m) \in \mathcal{B}} \text{score}(y_1 \dots y_m)$$

## Idea

- ▶ Maintain a “beam”  $\mathcal{B}_i$  at each time step  $i = 1 \dots m$  where

$$\mathcal{B}_i \approx \underset{y_1 \dots y_i}{\text{K-argmax}} \text{score}_x(y_1 \dots y_i)$$



# Beam Search Algorithm

- ▶ Base case ( $i = 1$ ):

$$\mathcal{B}_1 = \text{K-argmax}_{y \in L} \text{score}_x(y)$$

- ▶ Main body ( $i > 1$ ):

$$\mathcal{B}_i = \text{K-argmax}_{\substack{(y_1 \dots y_{i-1}) \in \mathcal{B}_{i-1} \\ y_i \in L}} \text{score}_x(y_1 \dots y_{i-1}) \times \text{score}_x(y_i | y_1 \dots y_{i-1})$$



## Leaky Priority Queue

- ▶ A “leaky” priority queue  $q$  with capacity  $K$
- ▶ Accepts a stream of elements  $[\text{thing}, \text{score}]$  but maintains only  $K$  elements with the highest scores seen so far.
- ▶ Both push and pop:  $O(\log K)$  worst-case time complexity
- ▶ Assume a  $O(K \log K)$  operation `dump`:

$$q.\text{dump}() = [q.\text{pop}() \text{ for } K \text{ times}]$$

- ▶ Exercise: try implementing it with a standard priority queue.

## Implementation

- ▶  $q \leftarrow \text{leaky\_priority\_queue}(K)$
- ▶  $q.\text{push}([y_1, \text{score}_x(y_1)]) \quad \forall y_1 \in L$
- ▶ For  $i = 2 \dots m$ :
  - ▶  $\mathcal{B}_{i-1} \leftarrow q.\text{dump}()$
  - ▶ For  $(y, s) \in \mathcal{B}_{i-1}$ :  
$$q.\text{push}([y.\text{append}(y_i), s \times \text{score}_x(y_i|y)]) \quad \forall y_i \in L$$
- ▶ Return  $q.\text{dump}()$ .

## Implementation

- ▶  $q \leftarrow \text{leaky\_priority\_queue}(K)$
- ▶  $q.\text{push}([y_1, \text{score}_x(y_1)]) \quad \forall y_1 \in L$
- ▶ For  $i = 2 \dots m$ :
  - ▶  $\mathcal{B}_{i-1} \leftarrow q.\text{dump}()$
  - ▶ For  $(y, s) \in \mathcal{B}_{i-1}$ :
$$q.\text{push}([y.\text{append}(y_i), s \times \text{score}_x(y_i|y)]) \quad \forall y_i \in L$$
- ▶ Return  $q.\text{dump}()$ .

Runtime complexity:  $O(|L| K \log K m)$

Compare with first-order HMM's forward/Viterbi:  $O(|L|^2 m)$

# Summary

- ▶ The goal of structured prediction is to perform efficient computation over **structured objects**.
- ▶ Today: we focused on HMMs, an important class of model used for structured prediction.
- ▶ Many key ideas illustrated with HMMs (Viterbi/Forward algorithm, beam search) play a central role in more general structured models.