

Hidden Markov Model (HMM)

Viterbi Algorithm

Application to POS-Tagging

Lehrstuhlseminar, Montag, 26. August 2013

Hannah Bast

Chair of Algorithms and Data Structures

Department of Computer Science

University of Freiburg

Hidden Markov Model 1/3

■ Markov Chain

– Given a set S of **states**, e.g. $S = \{\text{☺}, \text{☹}, \text{☹}\}$

– Given probabilities for the initial state

$$\Pr(\text{☺}) = 1/2, \Pr(\text{☹}) = 1/4, \Pr(\text{☹}) = 1/4$$

– Given transition probabilities between states

$$\Pr(\text{☺} \rightarrow \text{☺}) = 1/2, \Pr(\text{☺} \rightarrow \text{☹}) = 1/4, \Pr(\text{☺} \rightarrow \text{☹}) = 1/4$$

$$\Pr(\text{☹} \rightarrow \text{☺}) = 1/4, \Pr(\text{☹} \rightarrow \text{☹}) = 1/2, \Pr(\text{☹} \rightarrow \text{☹}) = 1/4$$

$$\Pr(\text{☹} \rightarrow \text{☺}) = 1/4, \Pr(\text{☹} \rightarrow \text{☹}) = 1/4, \Pr(\text{☹} \rightarrow \text{☹}) = 1/2$$

– Each sequence of states now has a probability

$$\Pr(\text{☺} \rightarrow \text{☺} \rightarrow \text{☺} \rightarrow \text{☺}) = 1/2 \cdot 1/2 \cdot 1/2 \cdot 1/2 = 1/16$$

$$\Pr(\text{☺} \rightarrow \text{☹} \rightarrow \text{☺} \rightarrow \text{☹}) = 1/2 \cdot 1/4 \cdot 1/4 \cdot 1/4 = 1/128$$

■ Markov Chain, remarks

- The probability distribution for the next state depends only on the current state, not on the history before that

This is called the **Markov property**

- The transition probabilities form an $|S| \times |S|$ matrix, which is **row-stochastic** = sum of each row is 1

Note: there is no reason why column sums should be 1

- Typical question in Markov Chain theory (but not in this talk) is, whether the chain has a **stationary distribution**

= there is a probability distribution $p_1, \dots, p_{|S|}$ over the states such that, wherever you start, eventually each state s will be visited with probability approaching p_s

Hidden Markov Model 3/3

■ Hidden Markov Model

- Set H of hidden states, and set S of observable states

Hidden: slept well (S^+), slept badly (S^-)

Observable: 😊, 😐, ☹️

- Markov Model now for the hidden states

$$\Pr(S^+) = 3/4 \qquad \Pr(S^-) = 1/4$$

$$\Pr(S^+ \rightarrow S^+) = 3/4, \qquad \Pr(S^+ \rightarrow S^-) = 1/4$$

$$\Pr(S^- \rightarrow S^+) = 1/3, \qquad \Pr(S^- \rightarrow S^-) = 2/3$$

- Observed states are conditional on the hidden states

$$\Pr(\text{😊} \mid S^+) = 1/2, \quad \Pr(\text{😐} \mid S^+) = 1/4, \quad \Pr(\text{☹️} \mid S^+) = 1/4$$

$$\Pr(\text{😊} \mid S^-) = 0, \quad \Pr(\text{😐} \mid S^-) = 1/2, \quad \Pr(\text{☹️} \mid S^-) = 1/2$$

Viterbi Algorithm 1/5

■ High-level description

- Given a HMM and a sequence of observed states

😊 → 😊 → 😊 → 😐 → 😞 → 😐 → 😊 → 😊

- Find the most likely sequence of hidden states

S⁺ → S⁺ → S⁺ → S⁻ → S⁻ → S⁻ → S⁺ → S⁺

- Formally, given observations $O_1 = o_1, \dots, O_n = o_n$
find sequence of hidden states $H_1 = h_1, \dots, H_n = h_n$
such that the following probability is maximized

$$\Pr(O_1 = o_1, \dots, O_n = o_n, H_1 = h_1, \dots, H_n = h_n)$$

- Note: it would not make much sense to maximize

$$\Pr(O_1 = o_1, \dots, O_n = o_n \mid H_1 = h_1, \dots, H_n = h_n)$$

Viterbi Algorithm 2/5

SHORTCUT:

o_i for $O_i = o_i$
 z_i for $H_i = z_i$

■ Some preparation

- Let us first use the HMM definition to split up the probability in question into simpler terms

$$\Pr(O_1 = o_1, \dots, O_n = o_n, H_1 = h_1, \dots, H_n = h_n)$$

$$= \Pr(o_1, \dots, o_n, z_1, \dots, z_n) \quad \leftarrow \text{MP}$$

$$= \Pr(o_1, z_1) \cdot \Pr(o_2, z_2 | o_1, z_1) \cdot \Pr(o_3, z_3 | o_2, z_2) \cdot \dots$$

$$= \Pr(z_1) \cdot \Pr(o_1 | z_1) \cdot \prod_{i=2}^n \Pr(z_i | z_{i-1}) \cdot \Pr(o_i | z_i)$$

$$= \prod_{i=1}^n \Pr(z_i | z_{i-1}) \cdot \Pr(o_i | z_i)$$

where
 $\Pr(z_0) = 1$
 $\Pr(z_i | z_0) = \Pr(z_i)$
initial prob.

Viterbi Algorithm 3/5

■ Recursive formula

... FIRST TRY

$$M_{n+1} = \max_{z_1 \dots z_{n+1}} \prod_{i=1}^{n+1} \Pr(z_i | z_{i-1}) \cdot \Pr(o_i | z_i)$$

$$\neq \max_{z_{n+1}} \Pr(z_{n+1} | z_n) \cdot \Pr(o_{n+1} | z_{n+1})$$

$$= \max_{z_1 \dots z_n} \underbrace{\prod_{i=1}^n \Pr(z_i | z_{i-1}) \cdot \Pr(o_i | z_i)}_{=: M_n}$$

Viterbi Algorithm 4/5

■ Recursive formula

$$\rightarrow M_n = \max_{z_n} \tilde{M}_n(z_n)$$

$$\tilde{M}_n(z_n) := \max_{z_1 \dots z_{n-1}} \prod_{i=1}^n \Pr(z_i | z_{i-1}) \cdot \Pr(o_i | z_i)$$

$$\tilde{M}_{n+1}(z_{n+1}) = \max_{z_1 \dots z_n} \prod_{i=1}^{n+1} \Pr(z_i | z_{i-1}) \cdot \Pr(o_i | z_i)$$

$$= \Pr(o_{n+1} | z_{n+1}) \cdot \max_{z_n} \{ \Pr(z_{n+1} | z_n)$$

$$\cdot \underbrace{\max_{z_1 \dots z_{n-1}} \prod_{i=1}^n \Pr(z_i | z_{i-1}) \cdot \Pr(o_i | z_i)}_{= \tilde{M}_n(z_n)} \}$$

$$= \Pr(o_{n+1} | z_{n+1}) \cdot \max_{z_n} \{ \Pr(z_{n+1} | z_n) \cdot \tilde{M}_n(z_n) \}$$

$$\begin{aligned} & \max_{a, b} f(a) \cdot g(b) \\ &= \max_a \left\{ f(a) \cdot \max_b g(b) \right\} \end{aligned}$$

■ Time Complexity

- For $i = 1, \dots, n$ compute $\tilde{M}_i(h_i)$ for all $h_i \in H$
 - Each such computation has to compute a maximum over k values, where $k = |H|$ is the number of hidden states
 - Altogether $n \cdot k$ such values have to be computed
 - The time complexity is therefore $O(n \cdot k^2)$
 - Note: this only gives us the **value** of $\max_{h_1, \dots, h_n} \dots$
 - As usual, the h_1, \dots, h_n that achieve this maximum can be retrieved by remembering, in the computation of each $\tilde{M}_i(h_i)$ for which $\tilde{M}_{i-1}(h_{i-1})$ the max was achieved
- See the application example that follows ...

Application to POS-Tagging 1/3

■ POS-Tagging

- Problem: tag each word from a sentence with the right POS (part of speech) tag, for example

Time flies like an arrow

N **V** **P** **D** **N**

N = noun, V = verb, P = preposition, D = determiner

- Usually the first step in any kind of natural language processing (also for Broccoli)
- Solution using a **HMM**:
 - Observed states = the sequence of words
 - Hidden states = the sequence of POS-tags
 - Transition Probabilities = estimated from a large corpus

Application to POS-Tagging 2/3

■ Trivial algorithm

- Note that finding h_1, \dots, h_n that maximize

$$\Pr(O_1 = o_1, \dots, O_n = o_n, H_1 = h_1, \dots, H_n = h_n)$$

can also be computed by trying out all k^n possible combinations for h_1, \dots, h_n , where $k = \#states$

- For the POS-tagging application

$k = \#POS\text{-tags}$, $n = \#\text{words in the sentence}$

so k^n would be feasible (though not particularly fast)

- Indeed, the first POS-taggers were doing this

We will now apply the Viterbi algorithm, however ...

$N = \text{noun}, V = \text{verb}, O = \text{other}$

Application to POS-Tagging 3/3

	START		N	V	O
N	$\frac{1}{2}$	N	$\frac{1}{2}$	$\frac{1}{2}$	0
V	0	V	$\frac{1}{2}$	0	$\frac{1}{2}$
O	$\frac{1}{2}$	O	$\frac{1}{2}$	0	$\frac{1}{2}$

row sum = 1

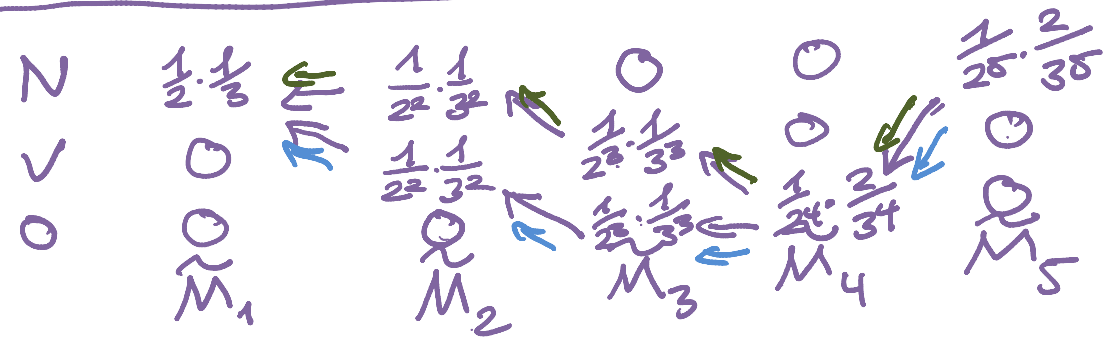
$$\tilde{\pi}_1(z_1) = \text{Pr}(z_1) \cdot \text{Pr}(o_1|z_1)$$

$$\tilde{\pi}_{m+1}(z_{m+1}) = \max_{z_m} \{ \text{Pr}(z_{m+1}|z_m) \cdot \tilde{\pi}_m(z_m) \} \cdot \text{Pr}(o_{m+1}|z_{m+1})$$

time flies like an arrow

N	$\frac{1}{3}$	$\frac{1}{3}$	0	0	$\frac{1}{3}$
V	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0	0
O	0	0	$\frac{1}{3}$	$\frac{2}{3}$	0

row sum = 1



NNVON
NVOON

References

■ Wikipedia

- http://en.wikipedia.org/wiki/Markov_chain
- http://en.wikipedia.org/wiki/Hidden_Markov_model
- http://en.wikipedia.org/wiki/Viterbi_algorithm
- http://en.wikipedia.org/wiki/Part-of-speech_tagging
- http://en.wikipedia.org/wiki/Andrey_Markov
- http://en.wikipedia.org/wiki/Andrew_Viterbi