INF4820: Algorithms for Artificial Intelligence and Natural Language Processing

Hidden Markov Models

Murhaf Fares & Stephan Oepen

Language Technology Group (LTG)

October 27, 2016

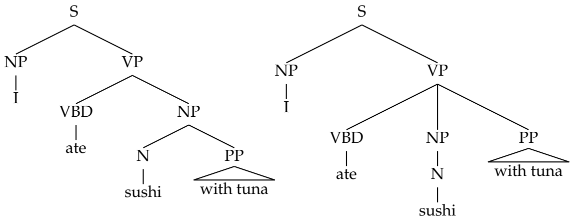
Recap: Probabilistic Language Models

- Basic probability theory: axioms, joint vs. conditional probability, independence, Bayes' Theorem;
- Previous context can help predict the next element of a sequence, for example words in a sentence;
- Rather than use the whole previous context, the Markov assumption says that the whole history can be approximated by the last $n-1$ elements;
- An n-gram language model predicts the n-th word, conditioned on the $n-1$ previous words;
- Maximum Likelihood Estimation uses relative frequencies to approximate the conditional probabilities needed for an n-gram model;
- Smoothing techniques are used to avoid zero probabilities.

Today

Determining

- which string is most likely: ✓
  - *She studies morphosyntax* vs. *She studies more faux syntax*
- which tag sequence is most likely for *flies like flowers*:
  - **NNS VB NNS** vs. **VBZ P NNS**
- which syntactic analysis is most likely:



Parts of Speech

- Known by a variety of names: part-of-speech, POS, lexical categories, word classes, morphological classes, …
- 'Traditionally' defined semantically (e.g. "nouns are naming words"), but more accurately by their distributional properties.

- Open-classes
  - New words created/updated/deleted all the time
- Closed-classes
  - Smaller classes, relatively static membership
  - Usually function words

Open Class Words

- Nouns: dog, Oslo, scissors, snow, people, truth, cups
  - proper or common; countable or uncountable; plural or singular; masculine, feminine or neuter; ...
- Verbs: fly, rained, having, ate, seen
  - transitive, intransitive, ditransitive; past, present, passive; stative or dynamic; plural or singular; ...
- Adjectives: good, smaller, unique, fastest, best, unhappy
  - comparative or superlative; predicative or attributive; intersective or non-intersective; ...
- Adverbs: again, somewhat, slowly, yesterday, aloud
  - intersective; scopal; discourse; degree; temporal; directional; comparative or superlative; ...

Closed Class Words

- Prepositions: *on*, *under*, *from*, *at*, *near*, *over*, ...
- Determiners: *a*, *an*, *the*, *that*, ...
- Pronouns: *she*, *who*, *I*, *others*, ...
- Conjunctions: *and*, *but*, *or*, *when*, ...
- Auxiliary verbs: *can*, *may*, *should*, *must*, ...
- Interjections, particles, numerals, negatives, politeness markers, greetings, existential there ...

(Examples from Jurafsky & Martin, 2008)

POS Tagging

The (automatic) assignment of POS tags to word sequences

- non-trivial where words are ambiguous: *fly* (v) vs. *fly* (n)
- choice of the correct tag is *context-dependent*
- useful in pre-processing for parsing, etc; but also directly for text-to-speech (TTS) system: **con**tent (n) vs. con**tent** (adj)
- difficulty and usefulness can depend on the *tagset*
  - English
    - Penn Treebank (PTB)—45 tags: NNS, NN, NNP, JJ, JJR, JJS

    `http://bulba.sdsu.edu/jeanette/thesis/PennTags.html`
  - Norwegian
    - Oslo-Bergen Tagset—multi-part: ⟨subst appell fem be ent⟩

    `http://tekstlab.uio.no/obt-ny/english/tags.html`

Labelled Sequences

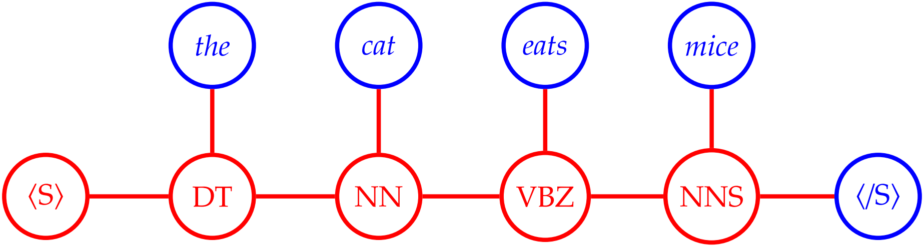
- We are interested in the probability of sequences like:

flies	like	the	wind		flies	like	the	wind
NNS	VB	DT	NN	or	VBZ	P	DT	NN

- In normal text, we see the words, but not the tags.
- Consider the POS tags to be underlying skeleton of the sentence, unseen but influencing the sentence shape.
- A structure like this, consisting of a hidden state sequence, and a related observation sequence can be modelled as a *Hidden Markov Model*.

Hidden Markov Models

The generative story:



$$
\begin{aligned}
P(S, O) = {} & P(\,\mathrm{DT}|\langle S\rangle)\; P(\mathrm{the}|\mathrm{DT})\; P(\mathrm{NN}|\mathrm{DT})\; P(\mathrm{cat}|\mathrm{NN}) \\
& P(\mathrm{VBZ}|\mathrm{NN})\; P(\mathrm{eats}|\mathrm{VBZ})\; P(\mathrm{NNS}|\mathrm{VBZ})\; P(\mathrm{mice}|\mathrm{NNS}) \\
& P(\langle /S\rangle|\mathrm{NNS})
\end{aligned}
$$

Hidden Markov Models

For a bi-gram HMM, with O_1^N:

$$P(S, O) = \prod_{i=1}^{N+1} P(s_i|s_{i-1})P(o_i|s_i) \quad \text{where} \quad s_0 = \langle S \rangle,\ s_{N+1} = \langle /S \rangle$$

- The transition probabilities model the probabilities of moving from state to state.
- The emission probabilities model the probability that a state *emits* a particular observation.

Using HMMs

The HMM models the process of generating the labelled sequence. We can use this model for a number of tasks:

- $P(S, O)$ given S and O
- $P(O)$ given O
- S that maximises $P(S|O)$ given O
- We can also learn the model parameters, given a set of observations.

Our *observations* will be words (w_i), and our *states* PoS tags (t_i)

Estimation

As so often in NLP, we learn an HMM from labelled data:

Transition probabilities

Based on a training corpus of previously tagged text, with tags as our state, the MLE can be computed from the counts of observed tags:

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

Emission probabilities

Computed from relative frequencies in the same way, with the words as observations:

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

Implementation Issues

$$\begin{aligned} P(S,O) &= P(s_1|\langle S\rangle)P(o_1|s_1)P(s_2|s_1)P(o_2|s_2)P(s_3|s_2)P(o_3|s_3)\ldots \\ &= 0.0429 \times 0.0031 \times 0.0044 \times 0.0001 \times 0.0072 \times \ldots \end{aligned}$$

- Multiplying many small probabilities $\rightarrow$ underflow
- Solution: work in log(arithmic) space:
  - $\log(AB) = \log(A) + \log(B)$
  - hence $P(A)P(B) = \exp(\log(A) + \log(B))$
  - $\log(P(S,O)) = -1.368 + -2.509 + -2.357 + -4 + -2.143 + \ldots$

The issues related to MLE / smoothing that we discussed for *n*-gram models also applies here …

Ice Cream and Global Warming

Missing records of weather in Baltimore for Summer 2007

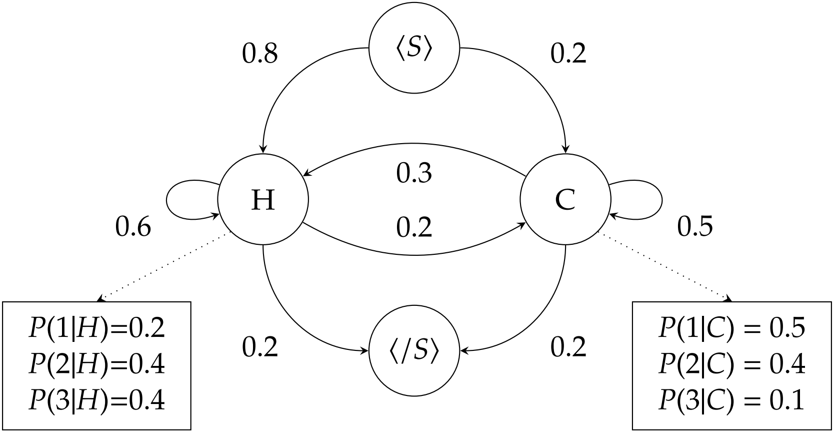
- Jason likes to eat ice cream.
- He records his daily ice cream consumption in his diary.
- The number of ice creams he ate was influenced, but not entirely determined by the weather.
- Today's weather is partially predictable from yesterday's.

A Hidden Markov Model!

with:

- Hidden states: $\{H, C\}$ (plus pseudo-states $\langle S \rangle$ and $\langle /S \rangle$)
- Observations: $\{1, 2, 3\}$

Ice Cream and Global Warming



Using HMMs

The HMM models the process of generating the labelled sequence. We can use this model for a number of tasks:

- $P(S, O)$ given S and O
- $P(O)$ given O
- S that maximises $P(S|O)$ given O
- $P(s_x|O)$ given O
- We can also learn the model parameters, given a set of observations.

Part-of-Speech Tagging

We want to find the tag sequence, given a word sequence. With tags as our states and words as our observations, we know:

$$P(S, O) = \prod_{i=1}^{N+1} P(s_i | s_{i-1}) P(o_i | s_i)$$

We want: $P(S|O) = \dfrac{P(S, O)}{P(O)}$

Actually, we want the state sequence $\hat{S}$ that maximises $P(S|O)$:

$$\hat{S} = \arg\max_S \frac{P(S, O)}{P(O)}$$

Since $P(O)$ always is the same, we can drop the denominator:

$$\hat{S} = \arg\max_S P(S, O)$$

Decoding

Task

What is the most likely state sequence S, given an observation sequence O and an HMM.

HMM		if O = 3 1 3					
$P(H\|\langle S\rangle) = 0.8$	$P(C\|\langle S\rangle) = 0.2$	$\langle S\rangle$	H	H	H	$\langle /S\rangle$	0.0018432
$P(H\|H) = 0.6$	$P(C\|H) = 0.2$	$\langle S\rangle$	H	H	C	$\langle /S\rangle$	0.0001536
$P(H\|C) = 0.3$	$P(C\|C) = 0.5$	$\langle S\rangle$	H	C	H	$\langle /S\rangle$	0.0007680
$P(\langle /S\rangle\|H) = 0.2$	$P(\langle /S\rangle\|C) = 0.2$	$\langle S\rangle$	H	C	C	$\langle /S\rangle$	0.0003200
$P(1\|H) = 0.2$	$P(1\|C) = 0.5$	$\langle S\rangle$	C	H	H	$\langle /S\rangle$	0.0000576
$P(2\|H) = 0.4$	$P(2\|C) = 0.4$	$\langle S\rangle$	C	H	C	$\langle /S\rangle$	0.0000048
$P(3\|H) = 0.4$	$P(3\|C) = 0.1$	$\langle S\rangle$	C	C	H	$\langle /S\rangle$	0.0001200
		$\langle S\rangle$	C	C	C	$\langle /S\rangle$	0.0000500

Dynamic Programming

For (only) two states and a (short) observation sequence of length three, comparing all possible sequences is workable, but . . .

- for N observations and L states, there are L^N sequences
- we do the same partial calculations over and over again

Dynamic Programming:

- records sub-problem solutions for further re-use
- useful when a complex problem can be described recursively
- examples: Dijkstra's shortest path, minimum edit distance, longest common subsequence, Viterbi algorithm

Viterbi Algorithm

Recall our problem:

$$\text{maximise } P(s_1 \dots s_n | o_1 \dots o_n) = P(s_1|s_0)P(o_1|s_1)P(s_2|s_1)P(o_2|s_2)\dots$$

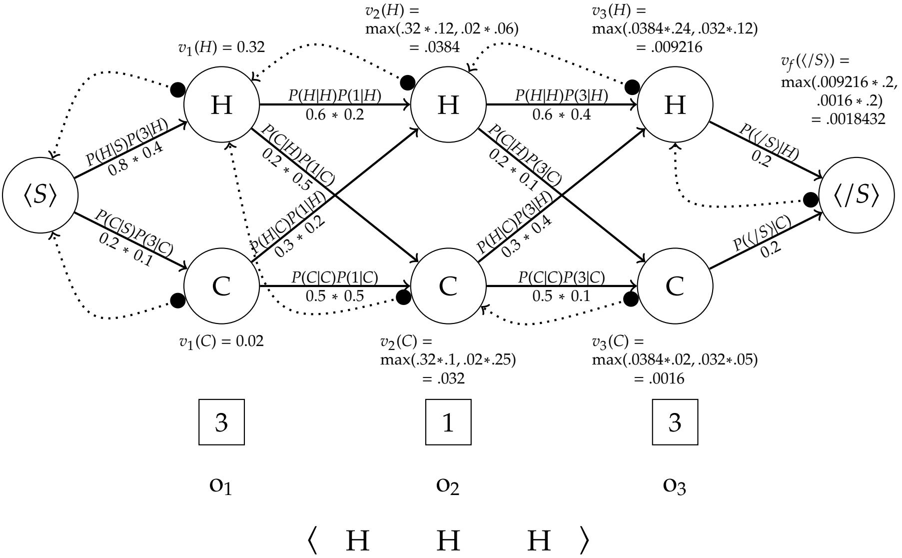
Our recursive sub-problem:

$$v_i(x) = \max_{k=1}^{L} [v_{i-1}(k) \cdot P(x|k) \cdot P(o_i|x)]$$

The variable $v_i(x)$ represents the maximum probability that the i-th state is x, given that we have seen O_1^i.

At each step, we record backpointers showing which previous state led to the maximum probability.

An Example of the Viterbi Algorithm



Pseudocode for the Viterbi Algorithm

Input: *observations* of length N, *state set* of size L
Output: *best-path*
create a path probability matrix $viterbi[N, L+2]$
create a path backpointer matrix $backpointer[N, L+2]$
for each *state s from 1 to L* **do**
    $viterbi[1, s] \leftarrow trans(\langle S\rangle, s) \times emit(o_1, s)$
    $backpointer[1, s] \leftarrow 0$
end
for each *time step i from 2 to N* **do**
    for each *state s from 1 to L* **do**
        $viterbi[i, s] \leftarrow \max_{s'=1}^{L} viterbi[i-1, s'] \times trans(s', s) \times emit(o_i, s)$
        $backpointer[i, s] \leftarrow \arg\max_{s'=1}^{L} viterbi[i-1, s'] \times trans(s', s)$
    end
end
$viterbi[N, L+1] \leftarrow \max_{s=1}^{L} viterbi[s, N] \times trans(s, \langle /S\rangle)$
$backpointer[N, L+1] \leftarrow \arg\max_{s=1}^{L} viterbi[N, s] \times trans(s, \langle /S\rangle)$
return the path by following backpointers from $backpointer[N, L+1]$

Diversion: Complexity and O(N)

Big-O notation describes the complexity of an algorithm.

- it describes the worst-case *order of growth* in terms of the size of the input
- only the largest order term is represented
- constant factors are ignored
- determined by looking at loops in the code

Pseudocode for the Viterbi Algorithm

Input: *observations* of length N, *state set* of length L
Output: *best-path*
create a path probability matrix $viterbi[N, L+2]$
create a path backpointer matrix $backpointer[N, L+2]$
for each *state s from 1 to L* **do** L
  $viterbi[1, s] \leftarrow trans(\langle S \rangle, s) \times emit(o_1, s)$
  $backpointer[1, s] \leftarrow 0$
end
for each *time step i from 2 to N* **do** N
  for each *state s from 1 to L* **do** L
    $viterbi[i, s] \leftarrow \max_{s'=1}^{L} viterbi[i-1, s'] \times trans(s', s) \times emit(o_i, s)$ L
    $backpointer[i, s] \leftarrow \arg\max_{s'=1}^{L} viterbi[i-1, s'] \times trans(s', s)$
  end
end
$viterbi[N, L+1] \leftarrow \max_{s=1}^{L} viterbi[s, N] \times trans(s, \langle /S \rangle)$
$backpointer[N, L+1] \leftarrow \arg\max_{s=1}^{L} viterbi[N, s] \times trans(s, \langle /S \rangle)$
return the path by following backpointers from $backpointer[N, L+1]$ N

$$O(L^2N)$$

Using HMMs

The HMM models the process of generating the labelled sequence. We can use this model for a number of tasks:

- $P(S, O)$ given S and O
- $P(O)$ given O
- S that maximises $P(S|O)$ given O
- $P(s_x|O)$ given O
- We can also learn the model parameters, given a set of observations.

Computing Likelihoods

Task

Given an observation sequence O, determine the likelihood $P(O)$, according to the HMM.

Compute the sum over all possible state sequences:

$$P(O) = \sum_S P(O, S)$$

For example, the ice cream sequence 3 1 3:

$$\begin{aligned} P(3\ 1\ 3) = \quad & P(3\ 1\ 3, \text{cold cold cold}) + \\ & P(3\ 1\ 3, \text{cold cold hot}) + \\ & P(3\ 1\ 3, \text{hot hot cold}) + \dots \quad \Rightarrow O(L^N N) \end{aligned}$$

The Forward Algorithm

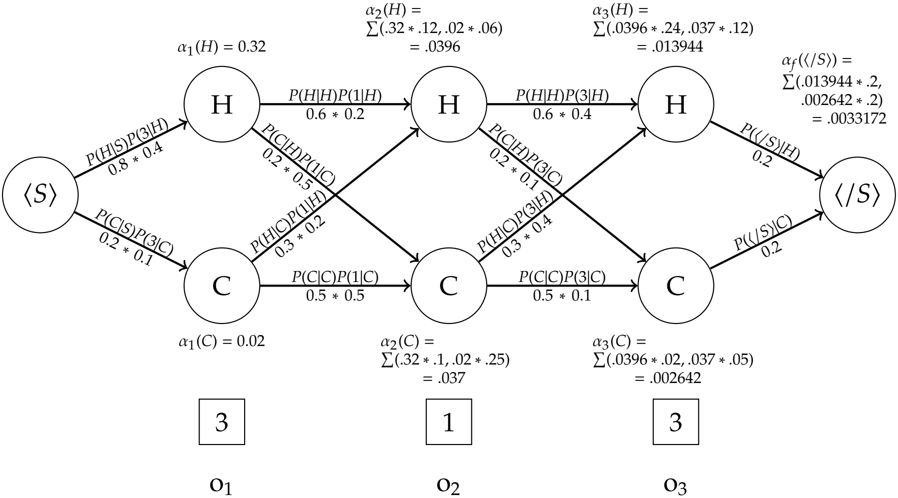
Again, we use dynamic programming—storing and reusing the results of partial computations in a trellis α.

Each cell in the trellis stores the probability of being in state s_x after seeing the first i observations:

$$
\begin{aligned}
\alpha_i(x) &= P(o_1 \ldots o_i, s_i = x) \\
&= \sum_{k=1}^{L} \alpha_{i-1}(k) \cdot P(x|k) \cdot P(o_i|x)
\end{aligned}
$$

Note $\sum$, instead of the max in Viterbi.

An Example of the Forward Algorithm



$$P(3\,1\,3) = 0.0033172$$

Pseudocode for the Forward Algorithm

Input: *observations* of length N, *state set* of length L
Output: *forward-probability*
create a probability matrix $forward[N, L+2]$
for each *state s from 1 to L* **do**
  $forward[1, s] \leftarrow trans(\langle S \rangle, s) \times emit(o_1, s)$
end
for each *time step i from 2 to N* **do**
  for each *state s from 1 to L* **do**
    $forward[i, s] \leftarrow$
    $\sum_{s'=1}^{L} forward[i-1, s] \times trans(s', s) \times emit(o_t, s)$
  end
end
$forward[N, L+1] \leftarrow \sum_{s=1}^{L} forward[N, s] \times trans(s, \langle /S \rangle)$
return $forward[N, L+1]$

Tagger Evaluation

To evaluate a part-of-speech tagger (or any classification system) we:

- train on a labelled training set
- test on a *separate* test set

For a POS tagger, the standard evaluation metric is tag accuracy:

$$Acc = \frac{\text{number of correct tags}}{\text{number of words}}$$

The other metric sometimes used is *error rate*:

$$\textit{error rate} = 1 - Acc$$

Summary

- Part-of-speech tagging as an example of sequence labelling.
- Hidden Markov Models to model the observation and hidden sequences.
- Learn the parameters of HMM (i.e. transition and emission probabilities) using MLE.
- Use Viterbi for decoding, i.e.: S that maximises $P(S|O)$ given O.
- Use Forward for computing likelihood, i.e.: $P(O)$ given O