

FNLP Lecture 9: Algorithms for HMMs

Henry S. Thompson

Based on slides by Sharon Goldwater

12 February 2019

Recap: HMM

- Elements of HMM:
 - Set of states (tags)
 - Output alphabet (word types)
 - Start state (beginning of sentence)
 - State transition probabilities
 - Output probabilities from each state

More general notation

- Previous lecture:
 - Sequence of tags $T = t_1 \dots t_n$
 - Sequence of words $S = w_1 \dots w_n$
- This lecture:
 - Sequence of states $Q = q_1 \dots q_T$
 - Sequence of outputs $O = o_1 \dots o_T$
 - So t is now a time step, not a tag! And T is the sequence length.

Recap: HMM

- Given a sentence $O = o_1 \dots o_T$ with tags $Q = q_1 \dots q_T$, compute $P(O, Q)$ as:

$$P(O, Q) = \prod_{t=1}^T P(o_t | q_t) P(q_t | q_{t-1})$$

- But we want to find $\operatorname{argmax}_Q P(Q|O)$ without enumerating all possible Q
 - Use Viterbi algorithm to store partial computations.

Today's lecture

- What algorithms can we use to
 - Efficiently compute the most probable tag sequence for a given word sequence?
 - Efficiently compute the likelihood for an HMM (probability it outputs a given sequence s)?
 - Learn the parameters of an HMM given unlabelled training data?
- What are the properties of these algorithms (complexity, convergence, etc)?

Tagging example

Words:

Possible tags:
(ordered by
frequency for
each word)

<s>	one	dog	bit	</s>
<s>	CD	NN	NN	</s>
	NN	VB	VBD	
	PRP			

- Choosing the best tag for each word independently gives the wrong answer (<s> CD NN NN </s>).
- $P(\text{VBD} | \text{bit}) < P(\text{NN} | \text{bit})$, but may yield a better *sequence* (<s> CD NN VB </s>)
 - because $P(\text{VBD} | \text{NN})$ and $P(\text{</s>} | \text{VBD})$ are high.

Viterbi: intuition

Words:

Possible tags:
(ordered by
frequency for
each word)

<s>	one	dog	bit	</s>
<s>	CD	NN	NN	</s>
	NN	VB	VBD	
	PRP			

- Suppose we have already computed
 - a) The best tag sequence for <s> ... bit that ends in NN.
 - b) The best tag sequence for <s> ... bit that ends in VBD.
- Then, the best full sequence would be either
 - sequence (a) extended to include </s>, or
 - sequence (b) extended to include </s>.

Viterbi: intuition

Words:

Possible tags:
(ordered by
frequency for
each word)

<s>	one	dog	bit	</s>
<s>	CD	NN	NN	</s>
	NN	VB	VBD	
	PRP			

- But similarly, to get
 - a) The best tag sequence for <s> ... bit that ends in NN.
- We could extend one of:
 - The best tag sequence for <s> ... dog that ends in NN.
 - The best tag sequence for <s> ... dog that ends in VB.
- And so on...

Viterbi: high-level picture

- Intuition: the best path of length t ending in state Q must include the best path of length $t-1$ to the previous state (call it P). (t now a *time*, not a *tag*):

$$\langle s \rangle / \langle s \rangle \quad \dots \quad o_{t-1} / P \quad o_t / Q$$

- Because otherwise there must be a better path to P that we should have used, thereby getting a better path to Q
 - Remember the Markov assumptions
 - $P(o_t|Q)$ and $P(Q|P)$ are *independent* of everything from $\langle s \rangle$ to Q

Viterbi: high-level picture

- Intuition: the best path of length t ending in state q must include the best path of length $t-1$ to the previous state. (t now a *time step*, not a *tag*). So,
 - Find the best path of length $t-1$ to each state.
 - Consider extending each of those by 1 step, to state q .
 - Take the best of those options as the best path to state q .

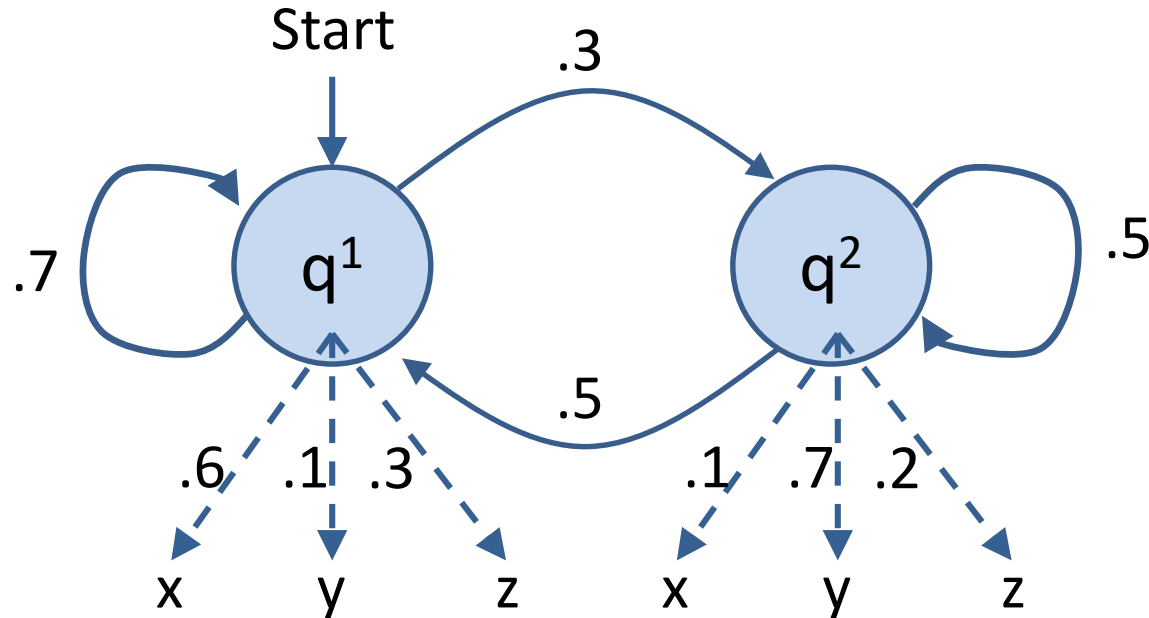
Notation

- Sequence of observations over time o_1, o_2, \dots, o_T
 - here, words in sentence
- Vocabulary size V of possible observations
- Set of possible states q^1, q^2, \dots, q^N (see note next slide)
 - here, tags
- A , an $N \times N$ matrix of transition probabilities
 - a_{ij} : the prob of transitioning from state q^i to q^j . (JM3 Fig 8.7)
- B , an $N \times V$ matrix of output probabilities
 - $b_i(o_t)$: the prob of emitting o_t from state q^i . (JM3 Fig 8.8)

Note on notation

- J&M use q_1, q_2, \dots, q_N for set of states, but *also* use q_1, q_2, \dots, q_T for state sequence over time.
 - So, just seeing q_1 is ambiguous (though usually disambiguated from context).
 - I'll instead use q^i for state names, and q_t for state at time t .
 - So we could have $q_t = q^i$, meaning: the state we're in at time t is q^i .

HMM example w/ new notation



- States $\{q^1, q^2\}$ (or $\{<s>, q^1, q^2\}$)
- Output alphabet $\{x, y, z\}$

Adapted from Manning & Schuetze, Fig 9.2

Transition and Output Probabilities

- Transition matrix **A**:

$$a_{ij} = P(q^j | q^i)$$

	q^1	q^2
$<s>$	1	0
q^1	.7	.3
q^2	.5	.5

- Output matrix **B**:

$$b_i(o) = P(o | q^i)$$

for output o

	x	y	z
q^1	.6	.1	.3
q^2	.1	.7	.2

Joint probability of (states, outputs)

- Let $\lambda = (A, B)$ be the parameters of our HMM.
- Using our new notation, given state sequence $Q = (q_1 \dots q_T)$ and output sequence $O = (o_1 \dots o_T)$, we have:

$$P(O, Q|\lambda) = \prod_{t=1}^T P(o_t|q_t)P(q_t|q_{t-1})$$

- Or:
$$P(O, Q|\lambda) = \prod_{t=1}^T b_{q_t}(o_t) a_{q_{t-1}q_t}$$

- Example:

$$\begin{aligned} P(O = (y, z), Q = (q^1, q^1)|\lambda) &= b_1(y) \cdot b_1(z) \cdot a_{<s>,1} \cdot a_{11} \\ &= (.1)(.3)(1)(.7) \end{aligned}$$

Viterbi: high-level picture

- Want to find $\operatorname{argmax}_Q P(Q|O)$
- Intuition: the best path of length t ending in state q must include the best path of length $t-1$ to the previous state. So,
 - Find the best path of length $t-1$ to each state.
 - Consider extending each of those by 1 step, to state q .
 - Take the best of those options as the best path to state q .

Viterbi algorithm

- Use a **chart** to store partial results as we go
 - $N \times T$ table, where $v(j, t)$ is the probability* of the best state sequence for $o_1 \dots o_t$ that ends in state j .
- Fill in columns from left to right, with
$$v(j, t) = \max_{i=1}^N v(i, t - 1) \cdot a_{ij} \cdot b_j(o_t)$$
- Store a **backtrace** to show, for each cell, which state at $t-1$ we came from.

*Specifically, $v(j, t)$ stores the max of the joint probability $P(o_1 \dots o_t, q_1 \dots q_{t-1}, q_t = j | \lambda)$

Example

- Suppose $O=xzy$. Our initially empty table:

	$o_1=x$	$o_2=z$	$o_3=y$
q^1			
q^2			


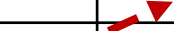
Filling the first column

	$o_1=x$	$o_2=z$	$o_3=y$
q^1	.6		
q^2	0		

$$v(1,1) = a_{<s>1} \cdot b_1(x) = (1)(.6)$$

$$v(2,1) = a_{<s>2} \cdot b_2(x) = (0)(.1)$$

Starting the second column

	$o_1=x$	$o_2=z$	$o_3=y$
q^1	.6		
q^2	0		

$$v(1,2) = \max_{i=1}^N v(i,1) \cdot a_{i1} \cdot b_1(z)$$

$$= \max \begin{cases} v(1,1) \cdot a_{11} \cdot b_1(z) = (.6)(.7)(.3) \\ v(2,1) \cdot a_{21} \cdot b_1(z) = (0)(.5)(.3) \end{cases}$$

Starting the second column

	$o_1=x$	$o_2=z$	$o_3=y$
q^1	.6	.126	
q^2	0		

$$v(1,2) = \max_{i=1}^N v(i,1) \cdot a_{i1} \cdot b_1(z)$$

$$= \max \begin{cases} v(1,1) \cdot a_{11} \cdot b_1(z) = (.6)(.7)(.3) \\ v(2,1) \cdot a_{21} \cdot b_1(z) = (0)(.5)(.3) \end{cases}$$

Finishing the second column

	$o_1=x$	$o_2=z$	$o_3=y$
q^1	.6	.126	
q^2	0		

$$v(2,2) = \max_{i=1}^N v(i,1) \cdot a_{i2} \cdot b_2(z)$$

$$= \max \begin{cases} v(1,1) \cdot a_{12} \cdot b_2(z) = (.6)(.3)(.2) \\ v(2,1) \cdot a_{22} \cdot b_2(z) = (0)(.5)(.2) \end{cases}$$

Finishing the second column

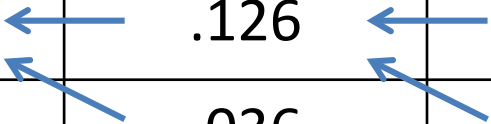
	$o_1=x$	$o_2=z$	$o_3=y$
q^1	.6	.126	
q^2	0	.036	

$$v(2,2) = \max_{i=1}^N v(i,1) \cdot a_{i2} \cdot b_2(z)$$

$$= \max \begin{cases} v(1,1) \cdot a_{12} \cdot b_2(z) = (.6)(.3)(.2) \\ v(2,1) \cdot a_{22} \cdot b_2(z) = (0)(.5)(.2) \end{cases}$$

Third column

	$o_1=x$	$o_2=z$	$o_3=y$
q^1	.6	.126	.00882
q^2	0	.036	.02646



- Exercise: make sure you get the same results!

Best Path

	$o_1=x$	$o_2=z$	$o_3=y$
q^1	.6	.126	.00882
q^2	0	.036	.02646

- Choose best final state: $\max_{i=1}^N v(i, T)$
- Follow backtraces to find best full sequence: $q^1 q^1 q^2$

HMMs: what else?

- As with probabilities in N-gram models and classification, chart probabilities get really tiny really fast, risking underflow
 - So, we use **costs** (negative log probabilities) instead
 - Take minimum over sum of costs, instead of maximum over product of probabilities.
- Using Viterbi, we can find the best tags for a sentence (**decoding**), and get $P(O, Q|\lambda)$.
- We might also want to
 - Compute the **likelihood** $P(O|\lambda)$, i.e., the probability of a sentence regardless of tags (a language model!)
 - **learn** the best set of parameters $\lambda = (A, B)$ given only an *unannotated* corpus of sentences.

Computing the likelihood

- From probability theory, we know that

$$P(O|\lambda) = \sum_Q P(O, Q|\lambda)$$

- There are an exponential number of Q s.
- Again, by computing and storing partial results, we can solve efficiently.
- (Next slides show the algorithm but I'll likely skip them)

Forward algorithm

- Use a table with cells $\alpha(j,t)$: the probability of being in state q^t after seeing $o_1 \dots o_t$ (**forward probability**).

$$\alpha(j, t) = P(o_1, o_2, \dots, o_t, q^t = j | \lambda)$$

- Fill in columns from left to right, with

$$\alpha(j, t) = \sum_{i=1}^N \alpha(i, t-1) \cdot a_{ij} \cdot b_j(o_t)$$

- Same as Viterbi, but sum instead of max (and no backtrace).

Note: because there's a sum, we can't use the trick that replaces probabilities with costs. For implementation info, see <http://digital.cs.usu.edu/~cyan/CS7960/hmm-tutorial.pdf> and <http://stackoverflow.com/questions/13391625/underflow-in-forward-algorithm-for-hmms>

Example

- Suppose $O=xzy$. Our initially empty table:

	$o_1=x$	$o_2=z$	$o_3=y$
q^1			
q^2			

Filling the first column

	$o_1=x$	$o_2=z$	$o_3=y$
q^1	.6		
q^2	0		

$$\alpha(1,1) = a_{<s>1} \cdot b_1(x) = (1)(.6)$$

$$\alpha(2,1) = a_{<s>2} \cdot b_2(x) = (0)(.1)$$

Starting the second column

	$o_1=x$	$o_2=z$	$o_3=y$
q^1	.6	.126	
q^2	0		

$$\begin{aligned}\alpha(1,2) &= \sum_{i=1}^N \alpha(i,1) \cdot a_{i1} \cdot b_1(z) \\ &= \alpha(1,1) \cdot a_{11} \cdot b_1(z) + \alpha(2,1) \cdot a_{21} \cdot b_1(z) \\ &= (.6)(.7)(.3) + (0)(.5)(.3) \\ &= .126\end{aligned}$$

Finishing the second column

	$o_1=x$	$o_2=z$	$o_3=y$
q^1	.6	.126	
q^2	0	.036	

$$\begin{aligned}\alpha(2,2) &= \sum_{i=1}^N \alpha(i,1) \cdot a_{i2} \cdot b_2(z) \\ &= \alpha(1,1) \cdot a_{12} \cdot b_2(z) + \alpha(2,1) \cdot a_{22} \cdot b_2(z) \\ &= (.6)(.3)(.2) + (0)(.5)(.2) \\ &= .036\end{aligned}$$

Third column and finish

	$o_1=x$	$o_2=z$	$o_3=y$
q^1	.6	.126	.01062
q^2	0	.036	.03906

- Add up all probabilities in last column to get the probability of the entire sequence:

$$P(O|\lambda) = \sum_{i=1}^N \alpha(i, T)$$

Learning

- Given *only* the output sequence, learn the best set of parameters $\lambda = (A, B)$.
- Assume 'best' = maximum-likelihood.
- Other definitions are possible, won't discuss here.

Unsupervised learning

- Training an HMM from an annotated corpus is simple.
 - **Supervised** learning: we have examples labelled with the right ‘answers’ (here, tags): no hidden variables in training.
- Training from unannotated corpus is trickier.
 - **Unsupervised** learning: we have no examples labelled with the right ‘answers’: all we see are outputs, state sequence is hidden.

Circularity

- If we know the state sequence, we can find the best λ .
 - E.g., use MLE: $P(q^j|q^i) = \frac{c(q^i \rightarrow q^j)}{c(q^i)}$
- If we know λ , we can find the best state sequence.
 - use Viterbi
- But we don't know either!

Expectation-maximization (EM)

Essentially, a bootstrapping algorithm.

- Initialize parameters $\lambda^{(0)}$
- At each iteration k ,
 - E-step: Compute **expected counts** using $\lambda^{(k-1)}$
 - M-step: Set $\lambda^{(k)}$ using MLE on the expected counts
- Repeat until λ doesn't change (or other stopping criterion).

Expected counts??

Counting transitions from $q^i \rightarrow q^j$:

- Real counts:
 - count 1 each time we see $q^i \rightarrow q^j$ in true tag sequence.
- Expected counts:
 - With current λ , compute probs of all possible tag sequences.
 - If sequence Q has probability p , count p for each $q^i \rightarrow q^j$ in Q .
 - Add up these fractional counts across all possible sequences.

Example

- Notionally, we compute expected counts as follows:

Possible sequence				Probability of sequence
$Q_1 =$	q^1	q^1	q^1	p_1
$Q_2 =$	q^1	q^2	q^1	p_2
$Q_3 =$	q^1	q^1	q^2	p_3
$Q_4 =$	q^1	q^2	q^2	p_4
Observs:	x	z	y	

$$\hat{C}(q^1 \rightarrow q^1) = 2p_1 + p_3$$

Forward-Backward algorithm

- As usual, avoid enumerating all possible sequences.
- **Forward-Backward** (Baum-Welch) algorithm computes expected counts using forward probabilities and **backward probabilities**:

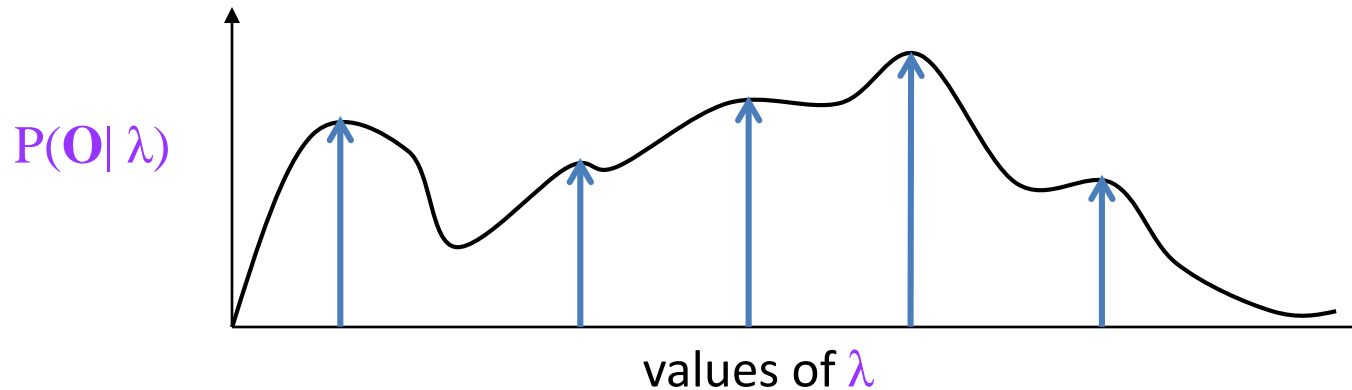
$$\beta(j, t) = P(q^t = j, o_{t+1}, o_{t+2}, \dots o_T | \lambda)$$

– Details, see J&M 6.5

- EM idea is much more general: can use for many latent variable models.

Guarantees

- EM is guaranteed to find a **local** maximum of the likelihood.



- Not guaranteed to find **global** maximum.
- Practical issues: initialization, random restarts, early stopping.

Summary

- HMM: a generative model of sentences using hidden state sequence
- Dynamic programming algorithms to compute
 - Best tag sequence given words (Viterbi algorithm)
 - Likelihood (forward algorithm)
 - Best parameters from unannotated corpus (forward-backward algorithm, an instance of EM)