# CS838-1 Advanced NLP: Hidden Markov Models

Xiaojin Zhu

2007
Send comments to jerryzhu@cs.wisc.edu

## 1 Part of Speech Tagging

Tag each word in a sentence with its part-of-speech, e.g.,

> The/AT representative/NN put/VBD chairs/NNS on/IN the/AT table/NN.

It is useful in information extraction, question answering, and shallow parsing. Tagging is easier than parsing. There are many tag sets, for example (part of the Penn Treebank Tagset)

| Tag | Part of Speech | Examples |
|---|---|---|
| AT | article | the, a, an |
| IN | preposition | atop, for, on, in, with, from |
| JJ | adjective | big, old, good |
| JJR | comparative adjective | bigger, older, better |
| MD | modal verb | may, can, will |
| NN | singular or mass noun | fish, lamp |
| NNP | singular proper noun | Madison, John |
| NNS | plural noun | cameras |
| PN | personal pronoun | he, you, me |
| RB | adverb | else, about, afterwards |
| VB | verb, base form | go |
| VBD | verb, past tense | went |
| VBG | verb, present participle | going |
| VBN | verb, past participle | gone |
| VBP | verb, non-3rd person singular present | go |
| VBZ | verb, 3rd person singular present | goes |

The difficulty: a word might have multiple possible POS:

> I/PN can/MD can/VB a/AT can/NN.

Two kinds of information can be used for POS tagging:

1. Certain tag sequences are more likely than other, e.g., AT JJ NN is quite often, while AT JJ VBP is unlikely: "a new book".

2. A word can have multiple possible POS, but some are more likely than others, e.g., "flour" is more often a noun than a verb.

Question: given a word sequence $x_{1:N}$, how do we compute the most likely POS sequence $z_{1:N}$?

# 2 Hidden Markov Models

An HMM has the following components:

- $K$ states (e.g., tag types)
- initial distribution $\pi \equiv (\pi_1, \ldots, \pi_K)^\top$, a vector of size $K$.
- emission probabilities $p(x|z)$, where $x$ is an output symbol (e.g., a word) and $z$ is a state (e.g., a tag). In this example, $p(x|z)$ can be a multinomial for each $z$. Note it is possible for different states to output the same symbol – this is the source of difficulty. Let $\phi = \{p(x|z)\}$.
- state transition probabilities $p(z_n = j|z_{n-1} = i) \equiv A_{ij}$, where $A$ is a $K \times K$ transition matrix. This is a first-order Markov assumption *on the states*.

The parameters of an HMM is $\theta = \{\pi, \phi, A\}$. An HMM can be plotted as a transition diagram (note it is *not* a graphical model! The nodes are not random variables). However, its graphical model is a linear chain on hidden nodes $z_{1:N}$, with observed nodes $x_{1:N}$.

There is a nice "urn" model that explains HMM as a generative model. We can run an HMM for $n$ steps, and produce $x_{1:N}$, $z_{1:N}$. The joint probability is

$$p(x_{1:N}, z_{1:N}|\theta) = p(z_1|\pi)p(x_1|z_1, \phi) \prod_{n=2}^{N} p(z_n|z_{n-1}, A)p(x_n|z_n, \phi). \quad (1)$$

Since the same output can be generated by multiple states, there usually are more than one state sequence that can generate $x_{1:N}$.

The problem of tagging is $\arg\max_{z_{1:N}} p(z_{1:N}|x_{1:N}, \theta)$, i.e., finding the *most likely state sequence* to explain the observation. As we see later, it is solved with the Viterbi algorithm, or more generally the max-sum algorithm. But we need good parameters $\theta$ first, which can be estimated using maximum likelihood on some (labeled and unlabeled) training data using EM (known as Baum-Welch algorithm for HMM). EM training involves so-called forward-backward or in general sum-product algorithm.

Besides tagging, HMM has been a *huge* success for acoustic modeling in speech recognition, where the observation is acoustic feature vector in a short time window, and the state is the phoneme.

# 3 HMM Training: The Baum-Welch Algorithm

Let $x_{1:N}$ be a single, very long training sequence of observed output (e.g., a long document), and $z_{1:N}$ its hidden labels (e.g., the corresponding tag sequence). It is easy to extend to multiple training documents.

## 3.1 The trivial case: $z_{1:N}$ observed

We find the maximum likelihood estimate of $\theta$ by maximizing the likelihood of observed data. Since both $x_{1:N}$ and $z_{1:N}$ are observed, MLE boils down to frequency estimate. In particular,

- $A_{ij}$ is the fraction of times $z = i$ is followed by $z' = j$ in $z_{1:N}$.
- $\phi$ is the MLE of output $x$ under $z$. For multinomials, $p(x|z)$ is the fraction of times $x$ is produced under state $z$.
- $\pi$ is the fraction of times each state being the first state of a sequence (assuming we have multiple training sequences).

## 3.2 The interesting case: $z_{1:N}$ unobserved

The MLE will maximize (up to a local optimum, see below) the likelihood of observed data

$$p(x_{1:N}|\theta) = \sum_{z_{1:N}} p(x_{1:N}, z_{1:N}|\theta), \tag{2}$$

where the summation is over *all possible label sequences of length* $N$. Already we see this is an exponential sum with $K^N$ label sequences, and brute force will fail. HMM training uses a combination of dynamic programming and EM to handle this issue.

Note the log likelihood involves summing over hidden variables, which suggests we can apply Jensen's inequality to lower bound (2).

$$\begin{aligned}
\log p(x_{1:N}|\theta) &= \log \sum_{z_{1:N}} p(x_{1:N}, z_{1:N}|\theta) && (3)\\
&= \log \sum_{z_{1:N}} p(z_{1:N}|x_{1:N}, \theta^{\text{old}}) \frac{p(x_{1:N}, z_{1:N}|\theta)}{p(z_{1:N}|x_{1:N}, \theta^{\text{old}})} && (4)\\
&\geq \sum_{z_{1:N}} p(z_{1:N}|x_{1:N}, \theta^{\text{old}}) \log \frac{p(x_{1:N}, z_{1:N}|\theta)}{p(z_{1:N}|x_{1:N}, \theta^{\text{old}})}. && (5)
\end{aligned}$$

We want to maximize the above lower bound instead. Taking the parts that depends on $\theta$, we define an auxiliary function

$$Q(\theta, \theta^{\text{old}}) = \sum_{z_{1:N}} p(z_{1:N}|x_{1:N}, \theta^{\text{old}}) \log p(x_{1:N}, z_{1:N}|\theta). \tag{6}$$

The $p(x_{1:N}, z_{1:N}|\theta)$ term is defined as the product along the chain in (1). By taking the log, it becomes summation of terms. $Q$ is the expectation of individual terms under the distribution $p(z_{1:N}|x_{1:N}, \theta^{\text{old}})$. In fact more variables will be marginalized out, leading to expectation under very simple distributions. For example, the first term in $\log p(x_{1:N}, z_{1:N}|\theta)$ is $\log p(z_1|\pi)$, and its expectation is

$$\sum_{z_{1:N}} p(z_{1:N}|x_{1:N}, \theta^{\text{old}}) \log p(z_1|\pi) = \sum_{z_1} p(z_1|x_{1:N}, \theta^{\text{old}}) \log p(z_1|\pi). \quad (7)$$

Note we go from an exponential sum over possible $z_{1:N}$ sequences to a single variable $z_1$, which has only $K$ values. Introducing the shorthand $\gamma_1(k) = p(z_1 = k|x_{1:N}, \theta^{\text{old}})$, the marginal of $z_1$ given input $x_{1:N}$ and old parameters, the above expectation is $\sum_{k=1}^{K} \gamma_1(k) \log \pi_k$.

In general, we use the shorthand

$$\gamma_n(k) = p(z_n = k|x_{1:N}, \theta^{\text{old}}) \quad (8)$$

$$\xi_n(jk) = p(z_{n-1} = j, z_n = k|x_{1:N}, \theta^{\text{old}}) \quad (9)$$

to denote the node marginals and edge marginals (conditioned on input $x_{1:N}$, under the old parameters). It will be clear that these marginal distributions play an important role. The $Q$ function can be written as

$$Q(\theta, \theta^{\text{old}}) = \sum_{k=1}^{K} \gamma_1(k) \log \pi_k + \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_n(k) \log p(x_n|z_n, \phi) + \sum_{n=2}^{N} \sum_{j=1}^{K} \sum_{k=1}^{K} \xi_n(jk) \log A_{jk}. \quad (10)$$

We are now ready to state the EM algorithm for HMMs, known as the Baum-Welch algorithm. Our goal is to find $\theta$ that maximizes $p(x_{1:N}|\theta)$, and we do so via a lower bound, or $Q$. In particular,

1. Initialize $\theta$ randomly or smartly (e.g., using limited labeled data with smoothing)
2. E-step: Compute the node and edge marginals $\gamma_n(k), \xi_n(jk)$ for $n = 1 \dots N$, $j, k = 1 \dots K$. This is done using the forward-backward algorithm (or more generally the sum-product algorithm), which is discussed in a separate note.
3. M-step: Find $\theta$ to maximize $Q(\theta, \theta^{\text{old}})$ as in (10).
4. Iterate E-step and M-step until convergence.

As usual, the Baum-Welch algorithm finds a *local optimum* of $\theta$ for HMMs.

# 4 The M-step

The M-step is a constrained optimization problem since the parameters need to be normalized. As before, one can introduce Lagrange multipliers and set the

gradient of the Lagrangian to zero to arrive at

$$\pi_k \propto \gamma_1(k) \tag{11}$$

$$A_{jk} \propto \sum_{n=2}^{N} \xi_n(jk) \tag{12}$$

Note $A_{jk}$ is normalized over $k$. $\phi$ is maximized depending on the particular form of the distribution $p(x_n|z_n, \phi)$. If it is multinomial, $p(x|z = k, \phi)$ is the frequency of $x$ in all output, but each output at step $n$ is weighted by $\gamma_n(k)$.

# 5 The E-step

We need to compute the marginals on each node $z_n$, and on each edge $z_{n-1}z_n$, conditioned on the observation $x_{1:N}$. Recall this is precisely what the sum-product algorithm can do. We can convert the HMM into a linear chain factor graph with variable nodes $z_{1:N}$ and factor nodes $f_{1:N}$. The graph is a chain with the order, from left to right, $f_1, z_1, f_2, z_2, \ldots, f_N, z_N$. The factors are

$$f_1(z_1) = p(z_1|\pi)p(x_1|z_1) \tag{13}$$

$$f_n(z_{n-1}, z_n) = p(z_n|z_{n-1})p(x_n|z_n). \tag{14}$$

We note that $x_{1:N}$ do not appear in the factor graph, instead they are absorbed into the factors.

We pass messages from left to right along the chain, and from right to left. This corresponds to the *forward-backward* algorithm. It is worth noting that since every variable node $z_n$ has only two factor node neighbors, it simply repeats the message it received from one of the neighbor to the other. The left-to-right messages are

$$\mu_{f_n \to z_n} = \sum_{k=1}^{K} f_n(z_{n-1} = k, z_n)\mu_{z_{n-1} \to f_n} \tag{15}$$

$$= \sum_{k=1}^{K} f_n(z_{n-1} = k, z_n)\mu_{f_{n-1} \to z_{n-1}} \tag{16}$$

$$= \sum_{k=1}^{K} p(z_n|z_{n-1} = k)p(x_n|z_n)\mu_{f_{n-1} \to z_{n-1}}. \tag{17}$$

It is not hard to show that

$$\mu_{f_n \to z_n} = p(x_1, \ldots, x_n, z_n), \tag{18}$$

and this message is called $\alpha$ in standard forward-backward literature, corresponding to the forward pass. Similarly, one can show that the right-to-left message

$$\mu_{f_{n+1} \to z_n} = p(x_{n+1}, \ldots, x_N|z_n). \tag{19}$$

This message is called $\beta$ and corresponds to the backward pass.

By multiplying the two incoming messages at any node $z_n$, we obtain the *joint*

$$p(x_{1:N}, z_n) = \mu_{f_n \to z_n} \mu_{f_{n+1} \to z_n} \tag{20}$$

since $x_{1:N}$ are observed (see the sum-product lecture notes, and we have equality because HMM is a directed graph). If we sum over $z_n$, we obtain the marginal likelihood of observed data

$$p(x_{1:N}) = \sum_{z_n=1}^{K} \mu_{f_n \to z_n} \mu_{f_{n+1} \to z_n}, \tag{21}$$

which is useful to monitor the convergence of Baum-Welch (note summing over any $z_n$ will give us this). Finally the desired node marginal is obtained by

$$\gamma_n(k) \equiv p(z_n = k | x_{1:N}) = \frac{p(x_{1:N}, z_n = k)}{p(x_{1:N})}. \tag{22}$$

A similar argument gives $\xi_n(jk)$ using the sum-product algorithm too.

# 6 Use HMM for Tagging

Given input (word) sequence $w_{1:N}$ and HMM $\theta$, tagging can be formulated as finding the most likely state sequence $z_{1:N}$ that maximizes $p(z_{1:N}|x_{1:N}, \theta)$. This is precisely the problem max-sum algorithm solves. In the context of HMMs, the max-sum algorithm is known as the Viterbi algorithm.