Yet Another Introduction to Hidden Markov Models

Anoop Sarkar
School of Computing Science, Simon Fraser University
Burnaby, BC V5A 1S6, Canada
anoop@cs.sfu.ca

Abstract

Yet another introduction to Baum-Welch re-estimation algorithm for learning parameters of a Hidden Markov Model (HMM). The presentation is from an NLP perspective and the focus is on deriving efficient pseudo-code for a fast implementation of the Viterbi, forward and backward algorithms that are used in Baum-Welch re-estimation. We consider only discrete states and discrete observations. In fact, to make the presentation NLP friendly we assume the task will be to train from word sequences with tagging sequences as hidden data, where in the decoding step each word will be eventually tagged with its appropriate tag. The tags are discrete, but can refer to part of speech tags, chunking tags, tags that indicate word boundaries, and other NLP applications that can be represented as a sequence learning problem.

1 Notation

We refer to states or tags as $t \in \{\texttt{1}, \ldots, \texttt{N}\}$ and observations or words as $w \in \{\texttt{1}, \ldots, \texttt{M}\}$. In place of the tag index or word index we sometimes use an actual tag $t = \texttt{DT}$ or word $w = \texttt{the}$ in some examples. A special word $w = \texttt{0}$ and tag $t = \texttt{0}$ is used to indicate the begin and end of a sequence. The sequence of states and associated observations, or tags and associated words is $w_0, t_0, \ldots, w_{T-1}, t_{T-1}$, a sequence of length T. We first consider a single sequence, later generalizing to multiple sequences of varying length.

2 Defining a Hidden Markov Model

A Hidden Markov Model (HMM) θ is the triple $\langle p_{\text{s}}, p_{\text{tt}}, p_{\text{tw}} \rangle$, where

1. $p_{\text{s}}(t_0)$ is the probability that we start with some tag t as t_0,
2. $p_{\text{tt}}(t_i \mid t_{i-1})$ is the transition probability from t_{i-1} to t_i, and
3. $p_{\text{tw}}(w_i \mid t_i)$ is the probability of generating w_i from t_i.

We set $p_{\text{s}}(\texttt{0}) = 1.0$ and $p_{\text{tw}}(\texttt{0} \mid \texttt{0}) = 1.0$. As a result we can now ignore p_{s} and our HMM is represented by the tuple $\langle p_{\text{tt}}, p_{\text{tw}} \rangle$.

3 The Forward Algorithm

Forward step (α):

$$
\begin{aligned}
\Pr(w_0 = \texttt{0} \mid t_0 = \texttt{0}) &= 1.0 \\
\Pr(w_0, \ldots, w_i, t_i) &= \sum_{t_{i-1}} \Pr(w_0, \ldots, w_{i-1}, t_{i-1}) \cdot p_{\text{tt}}(t_i \mid t_{i-1}) \cdot p_{\text{tw}}(w_i \mid t_i) \\
\Pr(w_0, \ldots, w_{T-1}) &= \sum_{t_{T-1}} \Pr(w_0, \ldots, w_{T-1}, t_{T-1}) \\
&= \Pr(w_0, \ldots, w_{T-1}, t_{T-1} = \texttt{0}) \qquad (1)
\end{aligned}
$$

$$\begin{aligned}
\alpha(\mathtt{0},0) &= 1.0 \\
\alpha(t_i,i) &= \sum_{t_{i-1}} \alpha(t_{i-1},i-1)\cdot p_{\text{tt}}(t_i \mid t_{i-1})\cdot p_{\text{tw}}(w_i \mid t_i), \quad 1 \le i \le T-1 \\
\Pr(w_0,\ldots,w_{T-1}) &= \sum_{t} \alpha(t,T-1) \\
&= \alpha(\mathtt{0},T-1) \qquad (2)
\end{aligned}$$

Algorithm 1 α: implements the forward algorithm

Require: $W = w_0,\ldots,w_{T-1}$
Require: *tag-dict*: $\{\mathtt{1},\ldots,\mathtt{M}\} \to \mathcal{P}(\{\mathtt{1},\ldots,\mathtt{N}\})$

$\alpha(\mathtt{0},0) := 1.0$
for $(1 \le i \le T-1)$ **do**
  for $(t_i \in$ *tag-dict*$(w_i))$ **do**
    for $(t_{i-1} \in$ *tag-dict*$(w_{i-1}))$ **do**
      $\alpha(t_i,i) := \alpha(t_i,i) + \alpha(t_{i-1},i-1)\cdot p_{\text{tt}}(t_i \mid t_{i-1})\cdot p_{\text{tw}}(w_i \mid t_i)$
    end for
  end for
end for
$s := \alpha(\mathtt{0},T-1)$
return s

Algorithm 2 α: implements the forward algorithm, 1st optimization

Require: $W = w_0,\ldots,w_{T-1}$
Require: *tag-dict*: $\{\mathtt{1},\ldots,\mathtt{M}\} \to \mathcal{P}(\{\mathtt{1},\ldots,\mathtt{N}\})$

$\alpha(\mathtt{0},0) := 1.0$
for $(1 \le i \le T-1)$ **do**
  for $(t_i \in$ *tag-dict*$(w_i))$ **do**
    for $(t_{i-1} \in$ *tag-dict*$(w_{i-1}))$ **do**
      $\alpha(t_i,i) := \alpha(t_i,i) + \alpha(t_{i-1},i-1)\cdot p_{\text{tt}}(t_i \mid t_{i-1})$
    end for
    $\alpha(t_i,i) := \alpha(t_i,i)\cdot p_{\text{tw}}(w_i \mid t_i)$
  end for
end for
$s := \alpha(\mathtt{0},T-1)$
return s

4 The Backward Algorithm

Backward step (β):

$$\begin{aligned}
\Pr(w_{T-1} = \mathtt{0} \mid t_{T-2}) &= 1.0 \quad \text{for all } t_{T-2} \\
\Pr(w_i,\ldots,w_{T-1} \mid t_{i-1}) &= \sum_{t_i} \Pr(w_{i+1},\ldots,w_{T-1} \mid t_i)\cdot p_{\text{tt}}(t_i \mid t_{i-1})\cdot p_{\text{tw}}(w_i \mid t_i) \\
\Pr(w_0,\ldots,w_{T-1}) &= \sum_{t_0} \Pr(w_1,\ldots,w_{T-1} \mid t_0)
\end{aligned}$$

$$= \Pr(w_1, \ldots, w_{T-1} \mid \mathtt{0}) \tag{3}$$

$$\begin{aligned}
\beta(\mathtt{0}, T-1) &= 1.0 \\
\beta(t_{i-1}, i-1) &= \sum_{t_i} \beta(t_i, i) \cdot p_{\text{tt}}(t_i \mid t_{i-1}) \cdot p_{\text{tw}}(w_i \mid t_i), \quad T-1 \geq i \geq 0 \\
\Pr(w_0, \ldots, w_{T-1}) &= \sum_t \beta(t, 0) \\
&= \beta(\mathtt{0}, 0)
\end{aligned} \tag{4}$$

Algorithm 3 β: implements the backward algorithm

Require: $W = w_0, \ldots, w_{T-1}$
Require: *tag-dict*: $\{\mathtt{1}, \ldots, \mathtt{M}\} \rightarrow \mathcal{P}(\{\mathtt{1}, \ldots, \mathtt{N}\})$

```
β(0, T − 1) := 1.0
for (T − 1 ≥ i ≥ 0) do
  for (t_i ∈ tag-dict(w_i)) do
    for (t_{i−1} ∈ tag-dict(w_{i−1})) do
      β(t_{i−1}, i − 1) := β(t_{i−1}, i − 1) + p_tt(t_i | t_{i−1}) · p_tw(w_i | t_i) · β(t_i, i)
    end for
  end for
end for
```

5 Baum-Welch Re-estimation

Computing γ:

$$\begin{aligned}
\gamma(i, t_i) &= \Pr(t_i \mid w_0, \ldots, w_{T-1}) \quad \text{for } t_i \in \{\mathtt{1}, \ldots, \mathtt{N}\} \\
&= \frac{\Pr(w_0, \ldots, w_i, t_i) \cdot \Pr(w_{i+1}, \ldots, w_{T-1} \mid t_i)}{\sum_{s_i} \Pr(w_0, \ldots, w_i, s_i) \cdot \Pr(w_{i+1}, \ldots, w_{T-1} \mid s_i)} \\
&= \frac{\Pr(w_0, \ldots, w_i, t_i) \cdot \Pr(w_{i+1}, \ldots, w_{T-1} \mid t_i)}{\Pr(w_0, \ldots, w_{T-1})} \\
&= \frac{\alpha(t_i, i) \cdot \beta(t_i, i)}{\alpha(\mathtt{0}, T-1)}
\end{aligned} \tag{5}$$

Computing ξ:

$$\begin{aligned}
\xi(i, t_i, t_{i+1}) &= \Pr(t_i, t_{i+1} \mid w_0, \ldots, w_{T-1}) \quad \text{for } t_i, t_{i+1} \in \{\mathtt{1}, \ldots, \mathtt{N}\} \\
&= \frac{\Pr(w_0, \ldots, w_i, t_i) \cdot p_{\text{tt}}(t_{i+1} \mid t_i) \cdot p_{\text{tw}}(w_{i+1} \mid t_{i+1}) \cdot \Pr(w_{i+2}, \ldots, w_{T-1} \mid t_{i+1})}{\sum_{s_i, s_{i+1}} \Pr(w_0, \ldots, w_i, s_i) \cdot p_{\text{tt}}(s_{i+1} \mid s_i) \cdot p_{\text{tw}}(w_{i+1} \mid s_{i+1}) \cdot \Pr(w_{i+2}, \ldots, w_{T-1} \mid s_{i+1})} \\
&= \frac{\alpha(t_i, i) \cdot p_{\text{tt}}(t_{i+1} \mid t_i) \cdot p_{\text{tw}}(w_{i+1} \mid t_{i+1}) \cdot \beta(t_{i+1}, i+1)}{\sum_{s_i, s_{i+1}} \alpha(s_i, i) \cdot p_{\text{tt}}(s_{i+1} \mid s_i) \cdot p_{\text{tw}}(w_{i+1} \mid s_{i+1}) \cdot \beta(s_{i+1}, i+1)} \\
&= \frac{\alpha(t_i, i) \cdot p_{\text{tt}}(t_{i+1} \mid t_i) \cdot p_{\text{tw}}(w_{i+1} \mid t_{i+1}) \cdot \beta(t_{i+1}, i+1)}{\Pr(w_0, \ldots, w_{T-1})} \\
&= \frac{\alpha(t_i, i) \cdot p_{\text{tt}}(t_{i+1} \mid t_i) \cdot p_{\text{tw}}(w_{i+1} \mid t_{i+1}) \cdot \beta(t_{i+1}, i+1)}{\alpha(\mathtt{0}, T-1)}
\end{aligned} \tag{6}$$

To save space in computing ξ, we compute ξ':

$$\xi'(t_i, t_{i+1}) = \sum_{i=0}^{T-2} \frac{\alpha(t_i, i) \cdot p_{\text{tt}}(t_{i+1} \mid t_i) \cdot p_{\text{tw}}(w_{i+1} \mid t_{i+1}) \cdot \beta(t_{i+1}, i+1)}{\alpha(\mathtt{0}, T-1)}$$
$$\text{for all } t_i, t_{i+1} \in \{\mathtt{1}, \ldots, \mathtt{N}\} \qquad (7)$$

and δ':

$$\delta'(t, w) = \sum_{i=0}^{T-1} \gamma(i, t_i = t) \cdot I(w_i = w)$$
$$\text{where } I(\mathtt{true}) = 1 \text{ and } I(\mathtt{false}) = 0.$$
$$\text{for all } t \in \{\mathtt{1}, \ldots, \mathtt{N}\}, w \in \{\mathtt{1}, \ldots, \mathtt{M}\} \qquad (8)$$

The new estimates for p_{tt} and p_{tw} are:

$$p_{\text{tt}}(t_{i+1} \mid t_i) = \frac{\xi'(t_i, t_{i+1})}{\sum_{s_i} \xi'(s_i, t_{i+1})} \qquad (9)$$

$$p_{\text{tw}}(w \mid t_i) = \frac{\delta'(t_i, w)}{\sum_{s_i} \delta'(s_i, w)} \qquad (10)$$

6 Conclusion

Did you understand it?

References

Jason Eisner. 2004. *Tagging with a Hidden Markov Model. Lecture Notes from Johns Hopkins course 600.465 — Introduction to NLP*

John Langford. 2000. *Optimizing Hidden Markov Model Learning.* unpublished manuscript.

Algorithm 4 Forward-Backward:
1 iteration of iterative algorithm to get new values for p_{tt} and p_{tw},
computes β at the same time

Require: $W = w_0, \ldots, w_{T-1}$
Require: *tag-dict*: $\{\mathtt{1}, \ldots, \mathtt{M}\} \rightarrow \mathcal{P}(\{\mathtt{1}, \ldots, \mathtt{N}\})$

$s := \alpha(W, \textit{tag-dict})$
$\beta(\mathtt{0}, T-1) := 1.0$
for $(T-1 \geq i \geq 0)$ **do**
  for $(t_i \in \textit{tag-dict}(w_i))$ **do**
    $\delta'(t_i, w_i) := \delta'(t_i, w_i) + \frac{1}{s} \cdot \alpha(i, t_i) \cdot \beta(i, t_i)$
    $\delta''(w_i) := \delta''(w_i) + \delta'(t_i, w_i)$
    for $(t_{i-1} \in \textit{tag-dict}(w_{i-1}))$ **do**
      $p := p_{\text{tt}}(t_i \mid t_{i-1}) \cdot p_{\text{tw}}(w_i \mid t_i)$
      $\beta(t_{i-1}, i-1) := \beta(t_{i-1}, i-1) + p \cdot \beta(t_i, i)$
      $\xi'(t_i, t_{i+1}) := \xi'(t_i, t_{i+1}) + \frac{1}{s} \cdot \alpha(t_{i-1}, i-1) \cdot p \cdot \beta(t_i, i)$
      $\xi''(t_{i+1}) := \xi''(t_{i+1}) + \xi'(t_i, t_{i+1})$
    end for
  end for
end for
for $(t_i \in \{\mathtt{1}, \ldots, \mathtt{N}\})$ **do**
  for $(w \in \{\mathtt{1}, \ldots, \mathtt{M}\})$ **do**
    $p_{\text{tw}}(w \mid t_i) := \frac{\delta'(t_i, w)}{\delta''(w_i)}$
  end for
  for $(t_{i+1} \in \{\mathtt{1}, \ldots, \mathtt{N}\})$ **do**
    $p_{\text{tt}}(t_{i+1} \mid t_i) := \frac{\xi'(t_i, t_{i+1})}{\xi''(t_{i+1})}$
  end for
end for

Algorithm 5 Forward-Backward: 1st optimization

Require: $W = w_0, \ldots, w_{T-1}$

Require: *tag-dict*: $\{\texttt{1}, \ldots, \texttt{M}\} \rightarrow \mathcal{P}(\{\texttt{1}, \ldots, \texttt{N}\})$

$s := \alpha(W, \textit{tag-dict})$
$\beta(\texttt{0}, T-1) := 1.0$
for $(T-1 \geq i \geq 0)$ **do**
    for $(t_i \in \textit{tag-dict}(w_i))$ **do**
        $\delta'(t_i, w_i) := \delta'(t_i, w_i) + \frac{1}{s} \cdot \alpha(i, t_i) \cdot \beta(i, t_i)$
        $\delta''(t_i) := \delta''(t_i) + \delta'(t_i, w_i)$
        $m := p_{\text{tw}}(w_i \mid t_i) \cdot \beta(t_i, i)$
        for $(t_{i-1} \in \textit{tag-dict}(w_{i-1}))$ **do**
            $n := p_{\text{tt}}(t_i \mid t_{i-1}) \cdot m$
            $\beta(t_{i-1}, i-1) := \beta(t_{i-1}, i-1) + n$
            $\xi'(t_i, t_{i+1}) := \xi'(t_i, t_{i+1}) + \frac{1}{s} \cdot \alpha(t_{i-1}, i-1) \cdot n$
            $\xi''(t_i) := \xi''(t_i) + \xi'(t_i, t_{i+1})$
        end for
    end for
end for
for $(t_i \in \{\texttt{1}, \ldots, \texttt{N}\})$ **do**
    for $(w \in \{\texttt{1}, \ldots, \texttt{M}\})$ **do**
        $p_{\text{tw}}(w \mid t_i) := \frac{\delta'(t_i, w)}{\delta''(t_i)}$
    end for
    for $(t_{i+1} \in \{\texttt{1}, \ldots, \texttt{N}\})$ **do**
        $p_{\text{tt}}(t_{i+1} \mid t_i) := \frac{\xi'(t_i, t_{i+1})}{\xi''(t_i)}$
    end for
end for

Algorithm 6 Forward-Backward: multiple training sentences

Require: $S = S_0, \ldots, S_{K-1}$, a list of training sentences
Require: *tag-dict*: $\{\texttt{1}, \ldots, \texttt{M}\} \rightarrow \mathcal{P}(\{\texttt{1}, \ldots, \texttt{N}\})$
for $(0 \leq k \leq K-1)$ **do**
  $S_k = w_0, \ldots, w_{T-1}$
  $s := \alpha(S_k, \textit{tag-dict})$
  $L := L \cdot s$
  $\beta(\texttt{0}, T-1) := 1.0$
  for $(T-1 \geq i \geq 0)$ **do**
    for $(t_i \in \textit{tag-dict}(w_i))$ **do**
      $\delta'(t_i, w_i) := \delta'(t_i, w_i) + \frac{1}{s} \cdot \alpha(i, t_i) \cdot \beta(i, t_i)$
      $\delta''(t_i) := \delta''(t_i) + \delta'(t_i, w_i)$
      $m := p_{\text{tw}}(w_i \mid t_i) \cdot \beta(t_i, i)$
      for $(t_{i-1} \in \textit{tag-dict}(w_{i-1}))$ **do**
        $n := p_{\text{tt}}(t_i \mid t_{i-1}) \cdot m$
        $\beta(t_{i-1}, i-1) := \beta(t_{i-1}, i-1) + n$
        $\xi'(t_i, t_{i+1}) := \xi'(t_i, t_{i+1}) + \frac{1}{s} \cdot \alpha(t_{i-1}, i-1) \cdot n$
        $\xi''(t_i) := \xi''(t_i) + \xi'(t_i, t_{i+1})$
      end for
    end for
  end for
end for
for $(t_i \in \{\texttt{1}, \ldots, \texttt{N}\})$ **do**
  for $(w \in \{\texttt{1}, \ldots, \texttt{M}\})$ **do**
    $p_{\text{tw}}(w \mid t_i) := \frac{\delta'(t_i, w)}{\delta''(t_i)}$
  end for
  for $(t_{i+1} \in \{\texttt{1}, \ldots, \texttt{N}\})$ **do**
    $p_{\text{tt}}(t_{i+1} \mid t_i) := \frac{\xi'(t_i, t_{i+1})}{\xi''(t_i)}$
  end for
end for
return L, the likelihood of the training set S

Algorithm 7 Baum-Welch Re-estimation: using only unlabelled data

Require: $S = S_0, \ldots, S_{K-1}$, a list of unlabelled training sentences
Require: *threshold*: small positive number, e.g. 10^{-5}, change in likelihood for stopping condition
for $(w \in S)$ **do**
  $\textit{tag-dict}(w) := \{\texttt{1}, \ldots, \texttt{N}\}$
end for
$L := 0$
repeat
  $L' := L$
  $L := \textit{Forward-Backward}(S, \textit{tag-dict})$
  assert $(L - L') > 0$
until $(L - L') > \textit{threshold}$
return $p_{\text{tt}}, p_{\text{tw}}$