Further details of the Baum-Welch algorithm

Martin Emms

November 15, 2018
Brute-force EM would for each $o^d$ calculate 'responsibility' $\gamma^d(s) = p(s|o^d)$ for all $s$ and from these calculate various expectations (eg. $E^d(i), E^d(ij)$).

Baum-Welch instead first runs $\alpha$ and $\beta$ for $o^d$. By various terms involving $\alpha$ and $\beta$ can derive various per $t$ 'clock tick' mini responsibilities.
real Baum-Welch: summing the 'clock-tick' probs

- brute-force EM would for each $o^d$ calculate 'responsibility' $\gamma^d(s) = p(s|o^d)$ for all $s$ and from these calculate various expectations (eg. $E^d(i)$, $E^d(ij)$)

- Baum-Welch instead first runs $\alpha$ and $\beta$ for $o^d$. By various terms involving $\alpha$ and $\beta$ can derive various per $t$ 'clock tick' mini responsibilities and then summations over these give various expectations ($E^d(i)$, $E^d(ij)$ etc). These are
real Baum-Welch: summing the 'clock-tick' probs

- brute-force EM would for each $o^d$ calculate 'responsibility' $\gamma^d(s) = p(s|o^d)$ for all $s$ and from these calculate various expectations (eg. $E^d(i)$, $E^d(ij)$)

- Baum-Welch instead first runs $\alpha$ and $\beta$ for $o^d$. By various terms involving $\alpha$ and $\beta$ can derive various per $t$ 'clock tick' mini responsibilities and then summations over these give various expectations ($E^d(i)$, $E^d(ij)$ etc).

These are

'occupation' $\gamma^d_t(i) = \text{the cond. prob. of state } i \text{ at } t \text{ given } o^d$

$= \alpha_t(i)\beta_t(i)/P(o^d)$
real Baum-Welch: summing the 'clock-tick' probs

- brute-force EM would for each $o^d$ calculate 'responsibility' $\gamma^d(s) = p(s|o^d)$ for all $s$ and from these calculate various expectations (eg. $E^d(i), E^d(ij)$)

- Baum-Welch instead first runs $\alpha$ and $\beta$ for $o^d$. By various terms involving $\alpha$ and $\beta$ can derive various per $t$ 'clock tick' mini responsibilities and then summations over these give various expectations ($E^d(i), E^d(ij)$ etc).

These are

'occupation' $\gamma^d_t(i) =$ the cond. prob. of state $i$ at $t$ given $o^d$

$$= \frac{\alpha_t(i) \beta_t(i)}{P(o^d)}$$

'transition' $\xi^d_t(i,j) =$ the cond. prob. of transition $ij$ at $t$ given $o^d$

$$= \frac{\alpha_t(i) \times a_{ij} b_j(o_{t+1}^d) \times \beta_{t+1}(j)}{P(o^d)}$$
Further details of the Baum-Welch algorithm

'occupation' $\gamma_t(i)$

$\alpha(t, i)$ $\beta(t, i)$

$\gamma^d_t(i) = \frac{\alpha_t(i) \beta_t(i)}{P(o^d)}$
'transition' $\xi_t(i,j)$

\[ \alpha(t, i) \]

\[ \beta(t+1, j) \]

\[ \begin{align*}
\xi_t(i,j) &= \text{the probability of transition } ij \text{ at } t \text{ given } o^d \\
&= \frac{[\alpha_t(i) \times a_{ij} b_j(o_{t+1}) \times \beta_{t+1}(j)]}{P(o)}
\end{align*} \]
Further details of the Baum-Welch algorithm

re-estimation of transition probs $A$

the re-estimation for the transition probs $a_{ij}$ involves getting the expected count of transition $ij$ and comparing to the expected count of $i$

$$\hat{a}_{ij} = \frac{\sum_d \sum_{t=1}^{T-1} \xi_t^d(i,j)}{\sum_d \sum_{t=1}^{T-1} \gamma_t^d(i)}$$

Note the limit $T - 1$: at the last time tick there is no defined $ij$ transition, nor should any expected state value at $T$ be relevant.
Further details of the Baum-Welch algorithm

picturing the numerator summation for transition probs

\[ \sum_{t=1}^{T-1} \xi_t(i,j) \]

sum over \( t \) = expectation of \( i \) to \( j \) given obs

\( \xi(t' i j) \)
\( \xi(t'' i j) \)
\( \xi(t''' i j) \)
the re-estimation for the obs probs $b_j(k)$ involves getting the expected count being in state $j$ and producing observation symbol $k$ and comparing this to the expected count of being in state $j$

$$\hat{b}(k) = \frac{\sum_{d} \sum_{t=1}^{T} \mathbf{1}_{o_t = k} \gamma_t(j)}{\sum_{d} \sum_{t=1}^{T} \gamma_t(j)}$$

in the numerator just the time ticks where $o_t = k$ are taken, and in the denominator every time tick is taken
Further details of the Baum-Welch algorithm

picturing the numerator summation for the observation probs

\[ \sum_{t=1}^{T} \sum_{o_t=k} \gamma_t(j) \]

sum over t where obs is k = expectation of ob k with state j given obs

\[ \gamma(t', j, k) \quad \gamma(t'', j, k) \quad \gamma(t''', j, k) \]

\[ \cdots \quad o \text{ at } t' = k \quad \cdots \quad o \text{ at } t'' = k \quad \cdots \quad o \text{ at } t''' = k \]
Further details of the Baum-Welch algorithm

picturing the numerator summation for the observation probs

\[ \sum_{t=1}^{T} \gamma_t(j) \mid o_t = k \text{ only where } o_t \text{ is obs } k \]
the re-estimation for start prob $\pi[i]$ involves getting the expected count of being in state $i$ at $t = 1$ and comparing to number of observations $D$

\[ \hat{\pi}[i] = \frac{\sum_d \gamma_1^d(i)}{D} \]
The backward algorithm

recall

'forward probability' $\alpha_t(i) = P(o_1 \ldots o_t, s_t = i)$
Recursion for $\alpha$
base $\alpha_1(i) = \pi(i)b_i(o_1)$
recursive $\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i)a_{ij}b_j(o_t)$, for $t = 2, \ldots, T$

corresponding for $\beta$

'backward probability' $\beta_t(i) = P(o_{t+1} \ldots o_T|s_t = i)$
Recursion for $\beta$
base $\beta_T(i) = 1$
recursive $\beta_t(i) = \sum_{j=1}^{N} a_{ij}b_j(o_{t+1})\beta_{t+1}(j)$, for $t = T - 1, \ldots, 1$
Further details of the Baum-Welch algorithm

**deriving $\beta$**

for $\beta_t(i)$ need $P(o_{t+1} \ldots o_T|s_t = i)$. Let $j$ be some arbitrary state at $t+1$. If had $P(s_{t+1} = j, o_{t+1} \ldots o_T|s_t = i)$, could sum over the $j$ to get desired quantity.

\[
P(s_{t+1} = j, o_{t+1} \ldots o_T|s_t = i) = \frac{P(s_t = i, s_{t+1} = j, o_{t+1} \ldots o_T)}{P(s_t = i)}
\]

\[
= \frac{P(s_t = 1, s_{t+1} = j, o_{t+1})\beta_{t+1}(j)}{P(s_t = i)}
\]

\[
= \frac{P(s_t = 1, s_{t+1} = j) b_j(o_{t+1})\beta_{t+1}(j)}{P(s_t = i)}
\]

\[
= a_{ij} b_j(o_{t+1})\beta_{t+1}(j)
\]

hence

\[
\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1})\beta_{t+1}(j)
\]