**Forward and Backward Probabilities**

word sequence $W = w_1 \ldots w_n$, for time $1 \leq t \leq n$

Forward probability:

(probability of seeing initial sequence $w_1 \ldots w_t$ and having tag $j$ at time $t$)

$$\alpha_{0}(0) = 1 \quad \alpha_{t}(t) = \left(\sum_{i=1}^{m} \alpha_{t-1}(i) a_{ij}(w_t)\right) b_{j}(w_t)$$

Backward probability:

(probability of seeing remaining sequence $w_{t+1} \ldots w_n$ given tag $i$ at time $t$)

$$\beta_{i}(n) = a_{i0} \quad \beta_{t}(t) = \sum_{j=1}^{m} \beta_{t+1}(j+1) a_{ji}(w_t) b_{j}(w_t)$$

$$P(w_1 \ldots w_n) = \beta_{0}(0) = \sum_{i=1}^{m} \alpha_{n}(i) a_{i0}$$

---

**New Parameters for Forward-Backward**

Probability of having tag $i$ at time $t$ given $w_1 \ldots w_n$

$$\gamma_{i}(t) = \frac{\alpha_{i}(t) \beta_{i}(t)}{\sum_{j=1}^{m} \alpha_{j}(t) \beta_{j}(t)}$$

Probability of having tag $i$ at time $t$ and tag $j$ at time $t + 1$, given $w_1 \ldots w_n$

$$\xi_{ij}(t) = \frac{\gamma_{i}(t) a_{ij}(w_{t+1}) \beta_{j}(t+1)}{\beta_{i}(t)}$$

---

**Forward-Backward Algorithm, E-step**

word sequence: $W = w_1 \ldots w_n$, size of tagset $|Z| = m$ \quad $\alpha_{0}(0) = 1$

for $1 \leq t \leq n$

for $j = 1$ to $m$

$$\alpha_{t}(t) = \sum_{i=1}^{m} \alpha_{t-1}(i) a_{ij}(w_t) b_{j}(w_t)$$

for $i = 1$ to $m$

$$\beta_{i}(n) = a_{i0}$$

for $t = n$ to $1$

$$\gamma_{i}(t) = \frac{\alpha_{i}(t) \beta_{i}(t)}{\sum_{j=1}^{m} \alpha_{j}(t) \beta_{j}(t)}$$

for $i = 1$ to $m$

$$\xi_{ij}(t) = \frac{\gamma_{i}(t) a_{ij}(w_{t+1}) \beta_{j}(t+1)}{\beta_{i}(t)}$$

for $j = 1$ to $m$

$$\xi_{ij}(t) = \frac{\gamma_{i}(t) a_{ij}(w_{t+1}) \beta_{j}(t+1)}{\beta_{i}(t)}$$

---

**Forward-Backward Algorithm, new model**

$$b_{i}(v_k) = \frac{\sum_{t=1}^{n} \delta_{w_t v_k} \gamma_{i}(t)}{\sum_{t=1}^{n} \gamma_{i}(t)}$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{n-1} \xi_{ij}(t)}{\sum_{t=1}^{n} \gamma_{i}(t)}$$

$$\hat{a}_{0j} = \gamma_{j}(1)$$

$$\hat{a}_{i0} = \frac{\gamma_{i}(n)}{\sum_{t=1}^{n} \gamma_{i}(t)}$$

where $\delta_{w_t v_k}$ is an indicator function indicating that the word at time $t$ was $v_k$. 

---

**Agenda**

- Homework
  - HW2 – graded by Thursday
  - HW3 – due Thursday
- Questions, comments, concerns?
- Unsupervised Learning
  - Expectation Maximization
- Supervised Learning – Discriminative Training
  - Perceptron
  - CRFs
Forward-Backward, M-step

corpus of N sentences, \( W_t = w_1^* \cdots w_m^* \), size of tagset \( |T| = m \)

initialize \( \alpha_{ij}, \alpha_{0i}, \alpha_{00} \), and \( b_j(v_k) \) to 0 for all \( i, j, k \)

for \( i = 1 \) to \( m \)

\[
e(t) \leftarrow \sum_{j=1}^{N} \sum_{W_j} \gamma_j(t)
\]

\[
\alpha_{0i} \leftarrow \frac{1}{N} \sum_{j=1}^{N} \gamma_j(t)
\]

\[
\alpha_{00} \leftarrow \frac{1}{N} \sum_{j=1}^{N} \gamma_j(|W_j|)
\]

for \( j = 1 \) to \( m \)

\[
\alpha_{ij} \leftarrow \frac{1}{c_i} \sum_{W_j} \sum_{t=1}^{T-1} \xi_{ij}(t)
\]

for \( k = 1 \) to \( |V| \)

\[
b_k(v_k) \leftarrow \frac{1}{c_i} \sum_{W_j} \sum_{t=1}^{T-1} \xi_{ij}(t)
\]

EM example

EM example: Forward (\( \alpha \))

words

\[\text{fruit} \quad \text{flies} \quad \text{fast}\]

EM example: Backward (\( \beta \))

words

\[\text{fruit} \quad \text{flies} \quad \text{fast}\]

EM example: \( \gamma \)

\[
\gamma_j(t) \leftarrow \frac{\alpha_{ij}(t) \beta_j(t)}{\sum_{j=1}^{m} \alpha_{ij}(t) \beta_j(t)}
\]

EM example: \( \xi \)

\[
\xi_{ij}(t) \leftarrow \gamma_i(t) \alpha_{ij} b_j(w_{t+1}) \beta_j(t+1)
\]

What is \( \beta_0(0) \)?

\[\beta_0(0) = 7.304 \times 10^{-8}\]
**Forward-Backward Algorithm, new model**

\[
\bar{b}_t(v_k) = \frac{\sum_{i=1}^{m} \delta_{w_i,v_k} \gamma_i(t)}{\sum_{i=1}^{m} \gamma_i(t)}
\]

\[
\bar{a}_{ij} = \frac{\sum_{i=1}^{m} \xi_{ij}(t)}{\sum_{i=1}^{m} \gamma_i(t)}
\]

\[
\bar{a}_{0j} = \gamma_j(1)
\]

\[
\bar{a}_{00} = \frac{\gamma_0(n)}{\sum_{i=1}^{m} \gamma_i(t)}
\]

where \( \delta_{w_i,v_k} \) is an indicator function indicating that the word at time \( t \) was \( v_k \).

**Forward-Backward, M-step**

- corpus of \( N \) sentences, \( W_s = w_1^S \ldots w_m^S \), size of tagset \( |T| = m \)
- initialize \( a_{0i}, a_{ij}, a_{0j} \), and \( b_i(v_k) \) to 0 for all \( i, j, k \)
- for \( i = 1 \) to \( m \)
  \[
  c_i(t) \leftarrow \sum_{k=0}^{N} \frac{w_i^S}{} \gamma^+_k(t)
  \]
  \[
  a_{0i} \leftarrow \frac{1}{S} \sum_{k=0}^{N} \gamma^+_k(t)
  \]
  \[
  a_{0j} \leftarrow \frac{1}{N} \sum_{k=0}^{N} \gamma^+_j(|W_s|)
  \]
  \[
  a_{ij} \leftarrow \frac{1}{S} \sum_{k=0}^{N} \sum_{t=1}^{m} \xi_{ij}(t)
  \]
  \[
  b_i(v_k) \leftarrow \frac{1}{S} \sum_{t=1}^{m} \sum_{k=0}^{N} \delta_{w_i,v_k} \gamma^+_k(t)
  \]

**Training versus held-out data**

- Train a model from training data
- Perform EM until convergence
- If training data is used, this doesn’t work (\( \lambda \leftarrow 1 \))
  - Already have maximum likelihood solution for training data
  - If we now try to maximize the likelihood . . .
  - Hold aside some data from training
  - Converge on held-out data
  - Prevents over-training

**Discriminative Training**

- Statistical model training involves maximizing some objective function
- For an HMM, we use maximum likelihood training
  - Maximize the probability of the training set
  - Reduction in errors is the true objective of learning
  - Another option is to try to directly optimize error rate or some other closely related objective
  - Consider not just truth, but also other candidates

**Agenda**

- Homework
- Unsupervised Learning
  - Expectation Maximization
- Supervised Learning – Discriminative Training
  - Perceptron
  - CRFs

**Perceptron**

- One approach that has been around since late 60s is the perceptron
- Basic idea:
  - Find the best scoring analysis (e.g. POS tag sequence)
  - Make its score lower, by penalizing its features
  - Make the score of the truth better, by rewarding its features
  - Go onto the next example
Perceptron Initialization

word sequence: \( W = w_1 \ldots w_n \), for time \( 1 \leq t \leq n \)
input (word) vocabulary: \( v_i \in V \) for \( 1 \leq i \leq k \)
output (tag) vocabulary: \( \tau_j \in \mathcal{T} \) for \( 1 \leq j \leq m \)
Let \( b_j(v_i) = P(\tau_j|\tau_i) = 0 \)
Let \( a_{ij} = P(\tau_j|\tau_i) = 0 \)
Let \( a_{ij} = P(<|\tau_i>) = 0 \)
Let \( \Phi(x,y) = [b_j(v), a_{ij}] \)

Perceptron Example

truth (\( y_0 \)): (PRP Her) (NN cat) (VB loves) (NN fruit) (.)
guess (\( x_0 \)): (AUX Her) (AUX cat) (AUX loves) (AUX fruit) (AUX)

Increment features of truth, decrement features of guess.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Truth</th>
<th>Guess</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P(&lt;</td>
<td>\tau_i&gt;) )</td>
<td>(-1)</td>
</tr>
<tr>
<td>( P(\tau_j</td>
<td>\tau_i) )</td>
<td>(-4)</td>
</tr>
<tr>
<td>( P(\tau_j</td>
<td>\tau_i) )</td>
<td>(1)</td>
</tr>
<tr>
<td>( P(\tau_j</td>
<td>\tau_i) )</td>
<td>(1)</td>
</tr>
<tr>
<td>( P(\tau_j</td>
<td>\tau_i) )</td>
<td>(1)</td>
</tr>
<tr>
<td>( P(\tau_j</td>
<td>\tau_i) )</td>
<td>(1)</td>
</tr>
<tr>
<td>( P(\tau_j</td>
<td>\tau_i) )</td>
<td>(1)</td>
</tr>
</tbody>
</table>

Perceptron Example

truth (\( y_1 \)): (PRP I) (RB always) (VB fast) (IN during) (NNP Lent) (.)
guess (\( x_1 \)): (PRP I) (AUX always) (DT fast) (AUX during) (DT Lent) (.)

Increment features of truth, decrement features of guess.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Truth</th>
<th>Guess</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P(&lt;</td>
<td>\tau_i&gt;) )</td>
<td>(-1)</td>
</tr>
<tr>
<td>( P(&lt;</td>
<td>\tau_i&gt;) )</td>
<td>(-1)</td>
</tr>
<tr>
<td>( P(&lt;</td>
<td>\tau_i&gt;) )</td>
<td>(-1)</td>
</tr>
<tr>
<td>( P(&lt;</td>
<td>\tau_i&gt;) )</td>
<td>(-1)</td>
</tr>
<tr>
<td>( P(&lt;</td>
<td>\tau_i&gt;) )</td>
<td>(-1)</td>
</tr>
<tr>
<td>( P(&lt;</td>
<td>\tau_i&gt;) )</td>
<td>(-1)</td>
</tr>
</tbody>
</table>
Perceptron Example

truth (y_i): (PRP Her) (NN cat) (VB loves) (NN fruit) (. .)

guess (z_i): (PRP Her) (RB cat) (VB loves) (NN fruit) (. .)

Perceptron: Notes

• Because this technique is optimizing (sequence) error rate, it does not involve a normalization factor
• Thus, it will overtrain
  • i.e. it will do very well on the training set, but not so well on new data, like unsmoothed maximum likelihood
  • Techniques exist for controlling overtraining, such as regularization, voting, and averaging
• Perceptron models outperform maximum likelihood–optimized models on a range of tasks
  • POS-tagging, NP-chunking

Agenda: Summary

• Review Forward-Backward algorithm
• Unsupervised learning
  • Apply EM
• Begin discussion of discriminative supervised learning
  • Perceptron
• Midterm – review next lecture