Hidden Markov models
Observable Markov model

• We’ve seen this type of model:
  • e.g., consider the 7-word vocabulary:
    \{
      \text{ship, pass, camp, frock, soccer, mother, tops}
    \}

• What is the probability of the sequence
  \text{ship, ship, pass, ship, tops} \text{?}

• Assuming a \textbf{bigram} model (i.e., 1\textsuperscript{st}-order Markov),
  \begin{align*}
  P(\text{ship}|<s>)P(\text{ship}|\text{ship})P(\text{pass}|\text{ship}) \\
  \cdot P(\text{ship}|\text{pass})P(\text{tops}|\text{ship})
  \end{align*}
Observable Markov model

- This can be conceptualized graphically.
- We start with $N$ states, $s_1, s_2, \ldots, s_N$ that represent unique observations in the world.
- Here, $N = 7$ and each state represents one of the words we can observe.
Observable Markov model

- We have discrete **timesteps**, \( t = 0, t = 1, ... \)

- On the \( t^{th} \) timestep the system is in exactly one of the available states, \( q_t \).
  - \( q_t \in \{s_1, s_2, ..., s_N\} \)

- We could start in any state. The probability of starting with a particular state \( s \) is \( P(q_0 = s) = \pi(s) \).
Observable Markov model

- At each step we must move to a state with some probability.
- Here, an arrow from $q_t$ to $q_{t+1}$ represents $P(q_{t+1}|q_t)$
- $P(\text{ship}|\text{ship})$
- $P(\text{tops}|\text{ship})$
- $P(\text{pass}|\text{ship})$
- $P(\text{frock}|\text{ship}) = 0$
Observable Markov model

• Probabilities on all outgoing arcs must sum to 1.

\[ P(\text{ship}|\text{ship}) + P(\text{tops}|\text{ship}) + P(\text{pass}|\text{ship}) = 1 \]

\[ P(\text{ship}|\text{tops}) + P(\text{tops}|\text{tops}) + P(\text{mother}|\text{tops}) = 1 \]

• ...

CSC401/2511 – Spring 2020
Using the graph

Random walk
Generate sequences by transitioning between states.

Observation likelihood
Given a path, build its probability.
A multivariate system

• What if the probabilities of observing words depended only on some other variable, like mood?

<table>
<thead>
<tr>
<th>word</th>
<th>P(word)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ship</td>
<td>0.1</td>
</tr>
<tr>
<td>pass</td>
<td>0.05</td>
</tr>
<tr>
<td>camp</td>
<td>0.05</td>
</tr>
<tr>
<td>flock</td>
<td>0.6</td>
</tr>
<tr>
<td>soccer</td>
<td>0.05</td>
</tr>
<tr>
<td>mother</td>
<td>0.1</td>
</tr>
<tr>
<td>tops</td>
<td>0.05</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>word</th>
<th>P(word)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ship</td>
<td>0.25</td>
</tr>
<tr>
<td>pass</td>
<td>0.25</td>
</tr>
<tr>
<td>camp</td>
<td>0.05</td>
</tr>
<tr>
<td>flock</td>
<td>0.3</td>
</tr>
<tr>
<td>soccer</td>
<td>0.05</td>
</tr>
<tr>
<td>mother</td>
<td>0.09</td>
</tr>
<tr>
<td>tops</td>
<td>0.01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>word</th>
<th>P(word)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ship</td>
<td>0.3</td>
</tr>
<tr>
<td>pass</td>
<td>0.0</td>
</tr>
<tr>
<td>camp</td>
<td>0.0</td>
</tr>
<tr>
<td>flock</td>
<td>0.2</td>
</tr>
<tr>
<td>soccer</td>
<td>0.05</td>
</tr>
<tr>
<td>mother</td>
<td>0.05</td>
</tr>
<tr>
<td>tops</td>
<td>0.4</td>
</tr>
</tbody>
</table>
A multivariate system

• What if that variable *changes* over time?
  • e.g., I’m *happy* one second and *disgusted* the next.
• Here, *state* ≡ mood
  *observation* ≡ word.

<table>
<thead>
<tr>
<th>word</th>
<th>P(word)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ship</td>
<td>0.1</td>
</tr>
<tr>
<td>pass</td>
<td>0.05</td>
</tr>
<tr>
<td>camp</td>
<td>0.05</td>
</tr>
<tr>
<td>flock</td>
<td>0.6</td>
</tr>
<tr>
<td>soccer</td>
<td>0.05</td>
</tr>
<tr>
<td>mother</td>
<td>0.1</td>
</tr>
<tr>
<td>tops</td>
<td>0.05</td>
</tr>
</tbody>
</table>
Observable multivariate systems

- Imagine you have access to my emotional state somehow.
- All your data are completely observable at every timestep.
- E.g.,

<table>
<thead>
<tr>
<th>$t$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
<td>...</td>
</tr>
<tr>
<td>word</td>
<td>mother</td>
<td>frock</td>
<td>soccer</td>
<td>...</td>
</tr>
</tbody>
</table>

$\equiv$

$\langle\text{mother, frock, soccer}, \langle😊,😊,😊\rangle\rangle$
Observable multivariate systems

• What is the probability of a sequence of words and states?
  • \( P(w_{0:t}, q_{0:t}) = P(q_{0:t})P(w_{0:t}|q_{0:t}) \approx \prod_{i=0}^{t} P(q_i|q_{i-1})P(w_i|q_i) \)

• e.g.,
  \[
P((\text{ship, pass}), (\text{😊,😊})) = P(q_0 = \text{😊})P(\text{ship}|\text{😊})P(\text{😊}|\text{😊})P(\text{pass}|\text{😊})
  \]
Observable multivariate systems

• **Q:** How do you **learn** these probabilities?
  • \( P(w_0:t, q_0:t) \approx \prod_{i=0}^{t} P(q_i|q_{i-1})P(w_i|q_i) \)

• **A:** When all data are observed, basically the same as before.
  • \( P(q_i|q_{i-1}) = \frac{P(q_{i-1}q_i)}{P(q_{i-1})} \) is learned with MLE from training data.
  • \( P(w_i|q_i) = \frac{P(w_i,q_i)}{P(q_i)} \) is also learned with MLE from training data.
Hidden variables

• Q: What if you don’t know the states during testing?
  • e.g., compute $P((\text{ship, ship, pass, frock}))$

• Q: What if you don’t know the states during training?
Examples of hidden phenomena

• We want to represent surface (i.e., observable) phenomena as the output of hidden underlying systems.
  • e.g.,
    • Words are the outputs of hidden parts-of-speech,
    • French phrases are the outputs of hidden English phrases,
    • Speech sounds are the outputs of hidden phonemes.

• in other fields,
  • Encrypted symbols are the outputs of hidden messages,
  • Genes are the outputs of hidden functional relationships,
  • Weather is the output of hidden climate conditions,
  • Stock prices are the outputs of hidden market conditions,
  • …
Definition of an HMM

- A **hidden Markov model** (HMM) is specified by the 5-tuple \(\{S, W, \Pi, A, B\}\):
  - \(S = \{s_1, \ldots, s_N\}\) : set of **states** (e.g., moods)
  - \(W = \{w_1, \ldots, w_K\}\) : output **alphabet** (e.g., words)

\[
\theta = \begin{cases} 
\Pi = \{\pi_1, \ldots, \pi_N\} & \text{: initial state probabilities} \\
A = \{a_{ij}\}, i, j \in S & \text{: state transition probabilities} \\
B = b_i(w), i \in S, w \in W & \text{: state output probabilities}
\end{cases}
\]

yielding
- \(Q = \{q_0, \ldots, q_{T-1}\}, q_i \in S\) : state sequence
- \(\mathcal{O} = \{\sigma_0, \ldots, \sigma_{T-1}\}, \sigma_i \in W\) : output sequence
A hidden Markov production process

• An HMM is a **representation** of a process in the world.
  • We can *synthesize* data, as in Shannon’s game.
• This is how an HMM **generates** new sequences:

  1. \( t := 0 \)
  2. **Start** in state \( q_0 = s_i \) with probability \( \pi_i \)
  3. **Emit** observation symbol \( \sigma_0 = w_k \) with probability \( b_i(\sigma_0) \)
  4. **While** *(not forever)*
     1. **Go** from state \( q_t = s_i \) to state \( q_{t+1} = s_j \) with probability \( a_{ij} \)
     2. **Emit** observation symbol \( \sigma_{t+1} = w_k \) with probability \( b_j(\sigma_{t+1}) \)
     3. \( t := t + 1 \)
Fundamental tasks for HMMs

1. Given a model with particular parameters $\theta = \langle \Pi, A, B \rangle$, how do we efficiently compute the likelihood of a particular observation sequence, $P(\mathcal{O}; \theta)$?

We previously computed the probabilities of word sequences using $N$-grams.

The probability of a particular sequence is usually useful as a means to some other end.
2. Given an observation sequence $\mathcal{O}$ and a model $\theta$, how do we choose a state sequence $Q = \{q_0, \ldots, q_{T-1}\}$ that best explains the observations?

This is the task of inference – i.e., guessing at the best explanation of unknown (‘latent’) variables given our model.

This is often an important part of classification.
Fundamental tasks for HMMs

3. Given a large observation sequence $\mathcal{O}$, how do we choose the best parameters $\theta = \langle \Pi, A, B \rangle$ that explain the data $\mathcal{O}$?

This is the task of training.

As before, we want our parameters to be set so that the available training data is maximally likely, but doing so will involve guessing unseen information.
A pro golfer can only putt the ball 3, 5, 7, and 11 meters.

He is currently 20m from the hole.

If he only sinks the ball if it stops directly in the hole,

*what is the minimum number of strokes to sink the ball?*
Answer: Golfer

- It takes two strokes if the golfer uses >1 dimensions.
Task 1: Computing $P(\mathcal{O}; \theta)$

- We’ve seen the probability of a joint sequence of observations and states:

$$P(\mathcal{O}, Q; \theta) = P(\mathcal{O} | Q; \theta) P(Q; \theta)$$

$$= \pi_{q_0} b_{q_0}(\sigma_0) a_{q_0 q_1} b_{q_1}(\sigma_1) a_{q_1 q_2} b_{q_2}(\sigma_2) \ldots$$

- To get the probability of our observations **without** seeing the state, we must **sum over all possible state sequences**:

$$P(\mathcal{O}; \theta) = \sum_Q P(\mathcal{O}, Q; \theta) = \sum_Q P(\mathcal{O} | Q; \theta) P(Q; \theta).$$
Computing $P(O; \theta)$ naively

• To get the total probability of our observations, we could directly sum over all possible state sequences:

$$P(O; \theta) = \sum_Q P(O|Q; \theta)P(Q; \theta).$$

• For observations of length $T$, each state sequence involves $2T$ multiplications (1 for each state transition, 1 for each observation, 1 for the start state, minus 1).

• There are up to $N^T$ possible state sequences of length $T$ given $N$ states.

$$\therefore \sim (1 + T + T - 1) \cdot N^T$$ multiplications 😞
Computing $P(\mathcal{O}; \theta)$ cleverly

• To avoid this complexity, we use **dynamic programming**; we **remember**, rather than **recompute**, partial results.

• We make a **trellis** which is an array of states vs. time.
  - The element at $(i, t)$ is $\alpha_i(t)$
    
    the probability of being in state $i$ at time $t$

    **after seeing all observations to that point:**

    $P(\sigma_{o:t}, q_t = s_i; \theta)$
Trellis

State

$s_1$

$s_2$

$s_3$

$s_N$

Time, $t$

$0$

$1$

$2$

$T - 1$

Probability of being in state $s_3$ at time $t = 2$

$a_{s_1s_2} b_{s_2}(\sigma_1)$

$a_{s_Ns_3} b_{s_3}(\sigma_2)$
Alternative paths through the trellis

State

\[ s_1 \]
\[ s_2 \]
\[ s_3 \]
\[ s_N \]

Time, \( t \)

0
1
2
\( T - 1 \)

Probability of being in state \( s_3 \) at time \( t = 2 \)
Alternative paths through the trellis

Probability of being in state $s_3$ at time $t = 2$
Alternative paths through the trellis

State

$S_1$

$S_2$

$S_3$

$S_N$

Time, $t$

Probability of being in state $s_3$ at time $t = 2$
Alternative paths through the trellis

Notice that I already computed a path through this node

Probability of being in state $s_3$ at time $t = 2$
Alternative paths through the trellis

Notice that I already computed a path through this node.

Each path through this node will have probability $P(\ldots) a_{s_2 s_3} b_{s_3}(\sigma_2)$.

$\sum P(\ldots) = \alpha_2(1)$
AND SO ON...
To compute the probabilities of the **black** node and the **yellow** node, I need (among others) the probabilities of the **orange** node and the **purple** node:

I compute once, and save them.
The Forward procedure

• To compute

\[ \alpha_i(t) = P(\sigma_{0:t}, q_t = s_i; \theta) \]

we can compute \( \alpha_j(t - 1) \) for possible previous states \( s_j \), then use our knowledge of \( a_{ji} \) and \( b_i(\sigma_t) \).

• We compute the trellis left-to-right (because of the convention of time) and top-to-bottom (‘just because’).

• Remember: \( \sigma_t \) is fixed and known. \( \alpha_i(t) \) is agnostic of the future.
The Forward procedure

- The trellis is computed **left-to-right** and **top-to-bottom**.

- There are three steps in this procedure:
  - **Initialization:** Compute the nodes in the *first column* of the trellis \((t = 0)\).
  - **Induction:** Iteratively compute the nodes in the *rest* of the trellis \((1 \leq t < T)\).
  - **Conclusion:** Sum over the nodes in the *last column* of the trellis \((t = T - 1)\).
Initialization of Forward procedure

\[ \alpha_i(0) := \pi_i b_i(\sigma_0), \]
\[ i := 1..N \]

(Probability of starting in state \( i \) and reading the first word there)
Induction of Forward procedure

\[ \alpha_j(t + 1) := \sum_{i=1}^{N} \alpha_i(t) a_{ij} b_j(\sigma_{t+1}), \]

for \( j := 1..N, t := 0..(T - 2) \)

(Probability of getting to state \( j \) at time \( t + 1 \))
Induction of Forward procedure

\[ s_1 \alpha_1(t) \to a_{1_j} b_j(\sigma_{t+1}) \]
\[ s_2 \alpha_2(t) \to a_{2_j} b_j(\sigma_{t+1}) \]
\[ s_3 \alpha_3(t) \to a_{3_j} b_j(\sigma_{t+1}) \]
\[ s_N \alpha_N(t) \to a_{N_j} b_j(\sigma_{t+1}) \]

\[ s_j \alpha_j(t+1) \]

\[ t \to t + 1 \]
Conclusion of Forward procedure

State: $s_1$, $s_2$, $s_3$, $s_N$

Time, $t$: 0, 1, 2, $T - 1$

Sum over all possible final states.

$$P(\mathcal{O}; \theta) = \sum_{i=1}^{N} \alpha_i(T - 1)$$
The Forward procedure - Example

Let’s compute $P(\langle \text{mother, flock, ship} \rangle)$

We need initial state probabilities $\pi$ and transition probabilities $\alpha_{ij}$

<table>
<thead>
<tr>
<th>$\pi$</th>
<th>$\alpha_{ij}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi = 0.80$</td>
<td>0.4</td>
</tr>
<tr>
<td>$\pi = 0.20$</td>
<td>0.1</td>
</tr>
<tr>
<td>$\pi = 0$</td>
<td>1.0</td>
</tr>
</tbody>
</table>

$t = 0$  $t = 1$  $t = 2$
The Forward procedure - Example

• Let’s compute $P(\langle \text{mother, flock, ship} \rangle)$

### Initialization

Compute the probability of starting in state $i$ and reading the first word there

$$\alpha_i(0) := \pi_i b_i(\sigma_0)$$

- $\alpha(0) = 0.80 \times 0.10 = 0.08$
- $\alpha(0) = 0.20 \times 0.09 = 0.018$
- $\alpha(0) = 0 \times 0.05 = 0$
The Forward procedure - Example

- Let’s compute $P(\langle \text{mother, flock, ship} \rangle)$

Iteratively compute the rest of the nodes in the trellis; i.e., the probability of getting to state $j$ at time $t+1$

$$\alpha_j(t+1) := \sum_{i=1}^{N} \alpha_i(t)a_{ij}b_j(\sigma_{t+1})$$

$$\alpha(t + 1) = 0.08(0.4)(0.6) + 0.018(0)(0.6) + 0(0)(0.6) = 0.0192$$
The Forward procedure - Example

- Let’s compute $P(\langle \text{mother, flock, ship} \rangle)$

```
<table>
<thead>
<tr>
<th>State</th>
<th>$t = 0$</th>
<th>$t = 1$</th>
<th>$t = 2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>mother</td>
<td>0.08</td>
<td>0.0192</td>
<td>0.00076</td>
</tr>
<tr>
<td>flock</td>
<td>0.018</td>
<td>0.0078</td>
<td>0.00283</td>
</tr>
<tr>
<td>ship</td>
<td>0.008</td>
<td>0.0048</td>
<td></td>
</tr>
</tbody>
</table>
```

**Induction**

Iteratively compute the rest of the nodes in the trellis; i.e., the probability of getting to state $j$ at time $t+1$

$$
\alpha_j(t + 1) := \sum_{i=1}^{N} \alpha_i(t) a_{ij} b_j(\sigma_{t+1})
$$
The Forward procedure - Example

- Let’s compute \( P(\langle \text{mother, flock, ship} \rangle) \)

\[
P(0; \theta) = \sum_{i=1}^{N} \alpha_i (T - 1)
\]

\[
P(0; \theta) = 0.00076 + 0.00283 + 0.0048 = 0.00839
\]
The Forward procedure

• The naïve approach needed \((2T) \cdot N^T\) multiplications.

• The Forward procedure (using \textit{dynamic programming}) needs only \(2N^2T\) multiplications. 😊

• The Forward procedure gives us \(P(\mathcal{O}; \theta)\).

• Clearly, but less intuitively, we can also compute the trellis from back-to-front, i.e., \textit{backwards in time}...
Remember the point

• The point was to compute the equivalent of

\[ P(\mathcal{O}; \theta) = \sum_Q P(\mathcal{O}, Q; \theta) \]

where

\[ P(\mathcal{O}, Q; \theta) = P(\mathcal{O}|Q; \theta)P(Q; \theta) \]

\[ = \pi_{q_0} b_{q_0}(\sigma_0) a_{q_0q_1} b_{q_1}(\sigma_1) a_{q_1q_2} b_{q_2}(\sigma_2) \ldots \]

\[ \alpha_i(0) \]
\[ \alpha_i(1) \]
\[ \alpha_i(2) \]

The Forward algorithm stores all possible 1-state sequences (from the start), to store all possible 2-state sequences (from the start), to store all possible 3-state sequences (from the start)…
Remember the point

- But, we can compute these factors in reverse
  \[ P(\mathcal{O}, Q; \theta) = P(\mathcal{O}|Q; \theta)P(Q; \theta) \]

  \[ = \pi_{q_0} \ldots b_{q_{T-3}}(\sigma_{T-3})a_{q_{T-3}q_{T-2}}b_{q_{T-2}}(\sigma_{T-2})a_{q_{T-2}q_{T-1}}b_{q_{T-1}}(\sigma_{T-1}) \]

  \[ \beta_i(T - 2) \]

  \[ \beta_i(T - 3) \]

  \[ \beta_i(T - 4) \]

We can still deal with sequences that evolve *forward* in time, but simply *store temporary results in reverse*...
The Backward procedure

• In the \((i, t)^{th}\) node of the trellis, we store
  \[
  \beta_i(t) = P(\sigma_{t+1:T-1}|\sigma_{0:t}, q_t = s_i; \theta) \\
  = P(\sigma_{t+1:T-1}|q_t = s_i; \theta)
  \]
  which is computed by summing probabilities on outgoing arcs from that node.

\(\beta_i(t)\) is the probability of starting in state \(i\) at time \(t\) then observing everything that comes thereafter.

• The trellis is computed right-to-left and top-to-bottom.
Step 1: Backward initialization

\[ \beta_i(T - 1) := 1, \quad i := 1..N \]

(We’ll see why, soon)
Step 2: Backward induction

\[ \beta_i(t) = \sum_{j=1}^{N} a_{ij} b_j(\sigma_{t+1}) \beta_j(t+1), \]
\[ j := 1..N, \quad t := (T - 2)\ldots 0 \]

(Probability of being in state \( i \) at time \( t \), then reading everything to follow)
Step 3: Backward conclusion

Sum over all possible initial states.

\[ P(\theta; \theta) = \sum_{i=1}^{N} \pi_i b_i(\sigma_0) \beta_i(0) \]
The Backward procedure

• Initialization
  $\beta_i(T - 1) = 1, \quad i := 1..N$

• Induction
  $\beta_i(t) = \sum_{j=1}^{N} a_{ij} b_j(\sigma_{t+1}) \beta_j(t + 1), \quad i := 1..N, \quad t := T - 2..0$

• Conclusion
  $P(\mathcal{O}; \theta) = \sum_{i=1}^{N} \pi_i b_i(\sigma_0) \beta_i(0)$
The Backward procedure – so what?

• The **combination** of Forward and Backward procedures will be vital for solving parameter re-estimation, i.e., **training**.

• Generally, we can **combine** $\alpha$ and $\beta$ at any point in time to represent the probability of an **entire** observation sequence...
Combining $\alpha$ and $\beta$

$$P(\mathcal{O}, q_t = i; \theta) = \alpha_i(t) \beta_i(t)$$

$$\therefore P(\mathcal{O}; \theta) = \sum_{i=1}^{N} \alpha_i(t) \beta_i(t)$$

This requires the current word to be incorporated by $\alpha_i(t)$, but not $\beta_i(t)$.

This isn’t merely for fun – it will soon become useful...
Fundamental tasks for HMMs

2. Given an observation sequence $O$ and a model $\theta$, how do we choose a state sequence $Q^* = \{q_0, \ldots, q_{T-1}\}$ that best explains the observations?

This is the task of inference – i.e., guessing at the best explanation of unknown ('latent') variables given our model.

This is often an important part of classification.
Task 2: Choosing $Q^* = \{q_0 \ldots q_{T-1}\}$

• The purpose of finding the best state sequence $Q^*$ out of all possible state sequences $Q$ is that it tells us what is most likely to be going on ‘under the hood’.

• With the Forward algorithm, we didn’t care about specific state sequences – we were summing over all possible state sequences.
Task 2: Choosing $Q^* = \{q_0 \ldots q_{T-1}\}$

- In other words,
  $$Q^* = \arg\max_Q P(\mathcal{O}, Q; \theta)$$

where

$$P(\mathcal{O}, Q; \theta) = \pi_{q_0} b_{q_0}(\sigma_0) \prod_{t=1}^{T-1} a_{q_{t-1}q_t} b_{q_t}(\sigma_t)$$
Why choose $Q^* = \{q_0 \ldots q_{T-1}\}$?

• Recall the purpose of HMMs:
  • To represent multivariate systems where some variable is unknown/hidden/latent.

• Finding the best hidden-state sequence $Q^*$ allows us to:
  • Identify unseen parts-of-speech given words,
  • Identify equivalent English words given French words,
  • Identify unknown phonemes given speech sounds,
  • Decipher hidden messages from encrypted symbols,
  • Identify hidden relationships from gene sequences,
  • Identify hidden market conditions given stock prices,
  • ...

Example – PoS state sequences

- Will/MD the/DT chair/NN chair/?? the/DT meeting/NN from/IN that/DT chair/NN?

a)
- MD
  - Will
- DT
  - the
- NN
  - chair
- VB
  - chair

b)
- MD
  - Will
- DT
  - the
- NN
  - chair
- NN
  - chair
Recall

- Observation likelihoods depend on the state, which changes over time.

- We **cannot** simply choose the state that maximizes the probability of $o_t$ without considering the state sequence.

<table>
<thead>
<tr>
<th>word</th>
<th>P(word)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ship</td>
<td>0.1</td>
</tr>
<tr>
<td>pass</td>
<td>0.05</td>
</tr>
<tr>
<td>camp</td>
<td>0.05</td>
</tr>
<tr>
<td>flock</td>
<td>0.6</td>
</tr>
<tr>
<td>soccer</td>
<td>0.05</td>
</tr>
<tr>
<td>mother</td>
<td>0.1</td>
</tr>
<tr>
<td>tops</td>
<td>0.05</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>word</th>
<th>P(word)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ship</td>
<td>0.3</td>
</tr>
<tr>
<td>pass</td>
<td>0.25</td>
</tr>
<tr>
<td>camp</td>
<td>0.05</td>
</tr>
<tr>
<td>flock</td>
<td>0.3</td>
</tr>
<tr>
<td>soccer</td>
<td>0.05</td>
</tr>
<tr>
<td>mother</td>
<td>0.09</td>
</tr>
<tr>
<td>tops</td>
<td>0.01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>word</th>
<th>P(word)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ship</td>
<td>0.25</td>
</tr>
<tr>
<td>pass</td>
<td>0.25</td>
</tr>
<tr>
<td>camp</td>
<td>0.05</td>
</tr>
<tr>
<td>flock</td>
<td>0.3</td>
</tr>
<tr>
<td>soccer</td>
<td>0.05</td>
</tr>
<tr>
<td>mother</td>
<td>0.09</td>
</tr>
<tr>
<td>tops</td>
<td>0.01</td>
</tr>
</tbody>
</table>
The Viterbi algorithm

- The Viterbi algorithm is an inductive dynamic-programming algorithm that uses a new kind of trellis.

- We define the probability of the most probable path leading to the trellis node at (state $i$, time $t$) as

\[
\delta_i(t) = \max_{q_0 \ldots q_{t-1}} P(q_0 \ldots q_{t-1}, \sigma_0 \ldots \sigma_{t-1}, q_t = s_i; \theta)
\]

- $\psi_i(t)$: The best possible previous state, if I’m in state $i$ at time $t$. 
Viterbi example

• For illustration, we assume a simpler state-transition topology:

<table>
<thead>
<tr>
<th>word</th>
<th>P(word)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ship</td>
<td>0.1</td>
</tr>
<tr>
<td>pass</td>
<td>0.05</td>
</tr>
<tr>
<td>camp</td>
<td>0.05</td>
</tr>
<tr>
<td>flock</td>
<td>0.6</td>
</tr>
<tr>
<td>soccer</td>
<td>0.05</td>
</tr>
<tr>
<td>mother</td>
<td>0.1</td>
</tr>
<tr>
<td>tops</td>
<td>0.05</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>word</th>
<th>P(word)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ship</td>
<td>0.25</td>
</tr>
<tr>
<td>pass</td>
<td>0.25</td>
</tr>
<tr>
<td>camp</td>
<td>0.05</td>
</tr>
<tr>
<td>flock</td>
<td>0.3</td>
</tr>
<tr>
<td>soccer</td>
<td>0.05</td>
</tr>
<tr>
<td>mother</td>
<td>0.09</td>
</tr>
<tr>
<td>tops</td>
<td>0.01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>word</th>
<th>P(word)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ship</td>
<td>0.3</td>
</tr>
<tr>
<td>pass</td>
<td>0.0</td>
</tr>
<tr>
<td>camp</td>
<td>0.0</td>
</tr>
<tr>
<td>flock</td>
<td>0.2</td>
</tr>
<tr>
<td>soccer</td>
<td>0.05</td>
</tr>
<tr>
<td>mother</td>
<td>0.05</td>
</tr>
<tr>
<td>tops</td>
<td>0.4</td>
</tr>
</tbody>
</table>
Step 1: Initialization of Viterbi

- Initialize with $\delta_i(0) = \pi_i b_i(\sigma_0)$ and $\psi_i(0) = 0$ for all states.
Step 1: Initialization of Viterbi

- For example, let's assume

  \[ \pi_d = 0.8, \pi_h = 0.2 \quad \text{and} \quad \emptyset = \text{ship, frock, tops} \]

  \[
  \begin{array}{c|c}
  \text{0} & \text{0} \\
  \hline
  \text{0.2} \cdot \text{0.3} & \text{0} \\
  \hline
  \text{0.8} \cdot \text{0.1} & \text{0} \\
  \end{array}
  \]

  \[ \sigma_0 = \text{ship} \]
  \[ \sigma_1 = \text{frock} \]
  \[ \sigma_2 = \text{tops} \]

  \( \delta \): max probability
  \( \psi \): backtrace

Observations, \( \sigma_t \)
Step 2: Induction of Viterbi

The best path to state $s_j$ at time $t$, $\delta_j(t)$, depends on the best path to each possible previous state, $\delta_i(t – 1)$, and their transitions to $j$, $a_{ij}$.

$$\delta_j(t) = \max_i \left[ \delta_i(t – 1) a_{ij} \right] b_j(\sigma_t)$$

$$\psi_j(t) = \arg\max_i \left[ \delta_i(t – 1) a_{ij} \right]$$

$\sigma_0 = ship$

$\sigma_1 = f\text{rock}$

$\sigma_2 = t\text{ops}$

Observations, $\sigma_t$
Step 2: Induction of Viterbi

Specifically...

\[
\delta_s(1) = \max_i [\delta_i(0)a_{is}] b_s(\sigma_1)
\]
\[
\psi_s(1) = \arg \max_i [\delta_i(0)a_{is}]
\]

\[
\delta_h(1) = \max_i [\delta_i(0)a_{ih}] b_h(\sigma_1)
\]
\[
\psi_h(1) = \arg \max_i [\delta_i(0)a_{ih}]
\]

\[
\delta_d(1) = \max_i [\delta_i(0)a_{id}] b_d(\sigma_1)
\]
\[
\psi_d(1) = \arg \max_i [\delta_i(0)a_{id}]
\]

\[
\sigma_0 = \text{ship}
\]
\[
\sigma_1 = \text{frock}
\]
\[
\sigma_2 = \text{tops}
\]

Observations, \(\sigma_t\)
Step 2: Induction of Viterbi

| δ_s(0) | a_{sd} = 0, \therefore δ_s(0)a_{sd} = 0 |
| δ_h(0) | a_{hd} = 0, \therefore δ_h(0)a_{hd} = 0 |
| δ_d(0) | a_{dd} = 0.4, \therefore δ_d(0)a_{dd} = 0.032 |

max_i [δ_i(0)a_{id}] b_d(σ_1)

argmax_i [δ_i(0)a_{id}]

σ_0 = ship

σ_1 = frock

σ_2 = tops

Observations, σ_t
Step 2: Induction of Viterbi

\[ \delta_s(1) = \max_i \left[ \delta_i(0)a_{is} \right] b_s(\sigma_1) \]

\[ \delta_d(0)a_{dd} = 0.032, \quad b_d(frock) = 0.6 \]

\[ \therefore \max_i \left[ \delta_i(0)a_{id} \right] b_d(\sigma_1) = 1.92 \times 10^{-2} = 1.92E^{-2} \]

\[ \varphi_n(1) = \arg \max_i \left[ \delta_i(0)a_{in} \right] \]

\[ d \text{ was the most likely previous state} \]

\[ \sigma_0 = \text{ship} \]

\[ \sigma_1 = \text{frock} \]

\[ \sigma_2 = \text{tops} \]

Observations, \( \sigma_t \)
Step 2: Induction of Viterbi

\[
\delta_s(0) = 0, \ a_{sh} = 0, \quad \therefore \delta_s(0)a_{sh} = 0 \\
\delta_h(0) = 0.06, \ a_{hh} = 0.8, \quad \therefore \delta_h(0)a_{hh} = 0.048 \\
\delta_d(0) = 0.08, \ a_{dh} = 0.5, \quad \therefore \delta_d(0)a_{dh} = 0.04
\]

\[
\max_i [\delta_i(0)a_{ih}] \ b_h(\sigma_1) \\
\arg\max_i [\delta_i(0)a_{ih}]
\]

\[
\begin{align*}
\sigma_0 &= \text{ship} \\
\sigma_1 &= \text{frock} \\
\sigma_2 &= \text{tops}
\end{align*}
\]

Observations, \( \sigma_t \)
Step 2: Induction of Viterbi

\[ \delta_h(0)a_{hh} = 0.048, \quad b_h(frock) = 0.2 \]

\[ \therefore \max_i [\delta_i(0)a_{ih}] b_h(\sigma_1) = 9.6 \times 10^{-3} = 9.6 E^{-3} \]

\[ \sigma_0 = \text{ship} \]

\[ \sigma_1 = \text{frock} \]

\[ \sigma_2 = \text{tops} \]
Step 2: Induction of Viterbi

\[
\begin{align*}
\delta_s(0) &= 0, a_{ss} = 1.0, \quad \therefore \delta_s(0)a_{ss} = 0 \\
\delta_h(0) &= 0.06, a_{hs} = 0.2, \quad \therefore \delta_h(0)a_{hs} = 0.012 \\
\delta_d(0) &= 0.08, a_{ds} = 0.1, \quad \therefore \delta_d(0)a_{ds} = 0.008
\end{align*}
\]

Observations, \( \sigma_t \)

\( \sigma_0 = ship \)  \( \sigma_1 = flock \)  \( \sigma_2 = tops \)
Step 2: Induction of Viterbi

\[ \delta_h(0)a_{hh} = 0.012, \quad b_s(frock) = 0.3 \]

\[ \therefore \max_i [\delta_i(0)a_{is}] b_s(\sigma_1) = 3.6 \times 10^{-3} = 3.6E^{-3} \]

\( \sigma_0 = ship \) \quad \sigma_1 = frock \quad \sigma_2 = tops

Observations, \( \sigma_t \)
Jack is looking at Anne. Anne is looking at George. Jack is married but George is not.

Is a married person looking at an unmarried person?

A. Yes
B. No
C. Not enough information
Answer: Marriage

- Jack is looking at Anne. Anne is looking at George. Jack is married but George is not.

Is a married person looking at an unmarried person? **YES.**
Step 2: Induction of Viterbi

Observations, $\sigma_t$

$\sigma_0 = ship$

$\sigma_1 = flock$

$\sigma_2 = tops$

$\delta_s(2) = \max_i [\delta_i(1)a_{is}] b_s(\sigma_2)$

$\psi_s(2) = \arg\max_i [\delta_i(1)a_{is}]$

$\delta_h(2) = \max_i [\delta_i(1)a_{ih}] b_h(\sigma_2)$

$\psi_h(2) = \arg\max_i [\delta_i(1)a_{ih}]$

$\delta_d(2) = \max_i [\delta_i(1)a_{id}] b_s(\sigma_2)$

$\psi_d(2) = \arg\max_i [\delta_i(1)a_{id}]$
Step 2: Induction of Viterbi

\[ \delta_s(1) = 3.6E^{-3}, a_{sd} = 0, \quad \therefore \delta_s(1)a_{sd} = 0 \]

\[ \delta_h(1) = 9.6E^{-3}, a_{hd} = 0, \quad \therefore \delta_h(1)a_{hd} = 0 \]

\[ \delta_d(1) = 1.92E^{-2}, a_{dd} = 0.4, \quad \therefore \delta_d(1)a_{dd} = 0.00768 \]

\[ \psi_h(2) = \text{argmax} \left[ \delta_i(1)a_{ih} \right] \]

\[ \psi_d(2) = \text{argmax} \left[ \delta_i(1)a_{id} \right] \]

\[ \delta_d(2) = \max_i \left[ \delta_i(1)a_{is} \right] b_s(\sigma_2) \]

\[ \sigma_0 = \text{ship} \quad \sigma_1 = \text{frock} \quad \sigma_2 = \text{tops} \]

Observations, \( \sigma_t \)
Step 2: Induction of Viterbi

Observations, $\sigma_t$

$\sigma_0 = ship$

$\sigma_1 = frock$

$\sigma_2 = tops$

$\delta_s(2) = 3.6E^{-3} \cdot 0.01$

$\psi_s(2) = s$

$\delta_h(2) = 9.6E^{-3} \cdot 0.4$

$\psi_h(2) = d$

$\delta_d(2) = 7.68E^{-3} \cdot 0.05$

$\psi_d(2) = d$
Step 3: Conclusion of Viterbi

Choose the best final state:

$$Q_T^* = \arg \max_i \delta_i(T)$$

\[
\begin{array}{c}
0.06 \\
0 \\
\end{array} 
\quad \begin{array}{c}
9.6 \times 10^{-3} \\
h \\
\end{array} 
\quad \begin{array}{c}
1.92 \times 10^{-2} \\
d \\
\end{array} 
\quad \begin{array}{c}
3.84 \times 10^{-4} \\
d \\
\end{array} 
\]

$\sigma_0 = ship$  $\sigma_1 = frock$  $\sigma_2 = tops$

Observations, $\sigma_t$

FIX!!!!!
Step 3: Conclusion of Viterbi

Recursively choose the best previous state:

\[ Q^*_{t-1} = \psi_{Q^*_t}(t) \]

Observations, \( \sigma_t \):

- \( \sigma_0 = \text{ship} \)
- \( \sigma_1 = \text{frock} \)
- \( \sigma_2 = \text{tops} \)
Step 3: Conclusion of Viterbi

Sequence probability:

\[ P(\mathcal{O}, Q^*; \theta) = \max_i \delta_i(T) \]

Observations, \( \sigma_t \):

- \( \sigma_0 = \text{ship} \)
- \( \sigma_1 = \text{frack} \)
- \( \sigma_2 = \text{tops} \)
Aside - Working in the log domain

• Our formulation was
  \[ Q^* = \arg\max_Q P(\mathcal{O}, Q; \theta) \]
  this is equivalent to
  \[ Q^* = \arg\min_Q -\log_2 P(\mathcal{O}, Q; \theta) \]
  where
  \[-\log_2 P(\mathcal{O}, Q; \theta) = -\log_2 \left( \pi_{q_0} b_{q_0}(\varphi_0) \right) - \sum_{t=1}^{T-1} \log_2 \left( a_{q_t-1 q_t} b_{q_t}(\sigma_t) \right)\]
Fundamental tasks for HMMs

3. Given a large observation sequence $O$ for training, but not the state sequence, how do we choose the ‘best’ parameters $\theta = \langle \Pi, A, B \rangle$ that explain the data $O$?

This is the task of **training**.

As with observable Markov models and **MLE**, we want our parameters to be set so that the available training data is maximally likely, but doing so will involve **guessing unseen information**...
Task 3: Choosing $\theta = \langle \Pi, A, B \rangle$

- We want to modify the parameters of our model $\theta = \langle \Pi, A, B \rangle$ so that $P(\mathcal{O}; \theta)$ is maximized for some training data $\mathcal{O}$:

$$\hat{\theta} = \arg\max_{\theta} P(\mathcal{O}; \theta)$$

- Why? E.g., if we later want to choose the best state sequence $Q^*$ for previously unseen test data, the parameters of the HMM should be tuned to similar training data.
Task 3: Choosing $\theta = \langle \Pi, A, B \rangle$

- $\hat{\theta} = \arg\max_{\theta} P(\mathcal{O}; \theta) = \arg\max_{\theta} \sum_{Q} P(\mathcal{O}, Q; \theta)$

- $P(\mathcal{O}, Q; \theta) = P(q_{0:T-1})P(w_{0:t} \mid q_{0:t}) \approx \prod_{i=0}^{t} P(q_i \mid q_{i-1})P(w_i \mid q_i)$

Recall that we could use MLE when $Q$ was known.
Task 3: Choosing $\theta = \langle \Pi, A, B \rangle$

- $P(O, Q; \theta) = P(q_0:t)P(w_0:t | q_0:t) \approx \prod_{t=0}^{t} P(q_i | q_{i-1})P(w_i | q_i)$

- If the training data contained state sequences, we could simply do maximum likelihood estimation, as before:
  
  - $P(q_i | q_{i-1}) = \frac{\text{Count}(q_{i-1} q_i)}{\text{Count}(q_{i-1})}$
  - $P(w_i | q_i) = \frac{\text{Count}(w_i \wedge q_i)}{\text{Count}(q_i)}$

- But we *don’t* know the states; we *can’t* count them.

- However, we *can* use an iterative hill-climbing approach if we can *guess* the counts using a “good” pre-existing model.
Expecting and maximizing

If we knew $\theta$, we could make expectations such as:
- Expected number of times in state $s_i$,
- Expected number of transitions $s_i \rightarrow s_j$.

If we knew:
- Expected number of times in state $s_i$,
- Expected number of transitions $s_i \rightarrow s_j$,
then we could compute the maximum likelihood estimate of

$$\theta = \langle \pi_i, \{a_{ij}\}, \{b_i(w)\} \rangle$$
Expectation-maximization

• **Expectation-maximization** (EM) is an **iterative** training algorithm that alternates between two steps:
  
  • **Expectation (E):** guesses the *expected* counts for the hidden sequence using the current model $\theta_k$.
  
  • **Maximization (M):** computes a new $\theta$ that **maximizes** the likelihood of the data, given the guesses of the E-step. This $\theta_{k+1}$ is then used in the next E-step.

• Continue until convergence or stopping condition...
Baum-Welch re-estimation

- **Baum-Welch (BW):**  
  *n.* a specific version of EM for HMMs.  
  a.k.a. ‘forward-backward’ algorithm.

1. Initialize the model.
2. Compute **expectations** for $\text{Count}(q_{t-1} q_t)$ and  
   $\text{Count}(q_t \wedge w_t)$ given model, training data $\mathcal{O}$.
3. Adjust our **start**, **transition**, and **observation**  
   probabilities to **maximize** the likelihood of $\mathcal{O}$.
4. Go to 2. and repeat until convergence or stopping  
   condition...
Local maxima

- Baum-Welch changes $\theta$ to climb a `hill' in $P(\mathcal{O}; \theta)$.
- How we initialize $\theta$ can have a big effect.
Step 1: BW initialization

- Our initial guess for the parameters, $\theta_0$, can be:
  a) All probabilities are uniform (e.g., $b_i(w_a) = b_i(w_b)$ for all states $i$ and words $w$)

<table>
<thead>
<tr>
<th>word</th>
<th>P(word)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ship</td>
<td>0.143</td>
</tr>
<tr>
<td>pass</td>
<td>0.143</td>
</tr>
<tr>
<td>camp</td>
<td>0.143</td>
</tr>
<tr>
<td>flock</td>
<td>0.143</td>
</tr>
<tr>
<td>soccer</td>
<td>0.143</td>
</tr>
<tr>
<td>mother</td>
<td>0.143</td>
</tr>
<tr>
<td>tops</td>
<td>0.143</td>
</tr>
</tbody>
</table>
Step 1: BW initialization

- Our **initial guess** for the parameters, $\theta_0$, can be:
  
  b) All probabilities are drawn **randomly** (subject to the condition that $\sum_i P(i) = 1$)

<table>
<thead>
<tr>
<th>word</th>
<th>$P(word)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ship</td>
<td>0.1</td>
</tr>
<tr>
<td>pass</td>
<td>0.05</td>
</tr>
<tr>
<td>camp</td>
<td>0.05</td>
</tr>
<tr>
<td>flock</td>
<td>0.6</td>
</tr>
<tr>
<td>soccer</td>
<td>0.05</td>
</tr>
<tr>
<td>mother</td>
<td>0.1</td>
</tr>
<tr>
<td>tops</td>
<td>0.05</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>word</th>
<th>$P(word)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ship</td>
<td>0.25</td>
</tr>
<tr>
<td>pass</td>
<td>0.25</td>
</tr>
<tr>
<td>camp</td>
<td>0.05</td>
</tr>
<tr>
<td>flock</td>
<td>0.3</td>
</tr>
<tr>
<td>soccer</td>
<td>0.05</td>
</tr>
<tr>
<td>mother</td>
<td>0.09</td>
</tr>
<tr>
<td>tops</td>
<td>0.01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>word</th>
<th>$P(word)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ship</td>
<td>0.3</td>
</tr>
<tr>
<td>pass</td>
<td>0.0</td>
</tr>
<tr>
<td>camp</td>
<td>0.0</td>
</tr>
<tr>
<td>flock</td>
<td>0.2</td>
</tr>
<tr>
<td>soccer</td>
<td>0.05</td>
</tr>
<tr>
<td>mother</td>
<td>0.05</td>
</tr>
<tr>
<td>tops</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Table 1: After re-estimation
Step 1: BW initialization

- Our initial guess for the parameters, $\theta_0$, can be:
  - Observation distributions are drawn from prior distributions:
    - $b_i(w_a) = P(w_a)$ for all states $i$.
    - Sometimes this involves pre-clustering, e.g. $k$-means.

All blue dots are words in state BLUE. Their probability distribution is:

<table>
<thead>
<tr>
<th>word</th>
<th>P(word)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ship</td>
<td>0.2</td>
</tr>
<tr>
<td>pass</td>
<td>0.1</td>
</tr>
<tr>
<td>camp</td>
<td>0.03</td>
</tr>
<tr>
<td>flock</td>
<td>0.5</td>
</tr>
<tr>
<td>soccer</td>
<td>0.07</td>
</tr>
<tr>
<td>mother</td>
<td>0.02</td>
</tr>
<tr>
<td>tops</td>
<td>0.08</td>
</tr>
</tbody>
</table>
What to expect when you’re expecting

• If we knew $\theta$, we could estimate **expectations** such as
  • Expected number of times in state $s_i$,
  • Expected number of transitions $s_i \rightarrow s_j$

• If we knew:
  • Expected number of times in state $s_i$,
  • Expected number of transitions $s_i \rightarrow s_j$
then we could compute the maximum likelihood estimate of

$$\theta = \langle \{a_{ij}\}, \{b_i(w)\}, \pi_i \rangle$$
BW E-step (occupation)

- We define

\[ \gamma_i(t) = P(q_t = i|\mathcal{O}; \theta_k) \]

as the probability of **being** in state \( i \) at time \( t \), based on our current model, \( \theta_k \), **given** the **entire** observation, \( \mathcal{O} \).

and rewrite as:

\[
\gamma_i(t) = \frac{P(q_t = i, \mathcal{O}; \theta_k)}{P(\mathcal{O}; \theta_k)} = \frac{\alpha_i(t)\beta_i(t)}{P(\mathcal{O}; \theta_k)}
\]

Remember, \( \alpha_i(t) \) and \( \beta_i(t) \) depend on values from
\( \theta = \langle \pi_i, a_{ij}, b_i(w) \rangle \)
Combining $\alpha$ and $\beta$

$$P(\mathcal{O}, q_t = i; \theta) = \alpha_i(t)\beta_i(t)$$

$$\therefore P(\mathcal{O}; \theta) = \sum_{i=1}^{N} \alpha_i(t)\beta_i(t)$$
BW E-step (transition)

- We define

\[ \xi_{ij}(t) = P(q_t = i, q_{t+1} = j | \mathcal{O}; \theta_k) \]

as the probability of **transitioning** from state $i$ at time $t$ to state $j$ at time $t + 1$ **based on** our current model, $\theta_k$, and **given** the **entire** observation, $\mathcal{O}$. This is:

\[
\xi_{ij}(t) = \frac{P(q_t = i, q_{t+1} = j, \mathcal{O}; \theta_k)}{P(\mathcal{O}; \theta_k)} = \frac{\alpha_i(t) a_{ij} b_j(\sigma_{t+1}) \beta_j(t + 1)}{P(\mathcal{O}; \theta_k)}
\]

Again, these estimates come from our model at iteration $k$, $\theta_k$. 
BW E-step (transition)

\[
\begin{align*}
\alpha_i(t) & \quad \text{at } t - 1 \\
& \quad \text{at } t \\
& \quad \text{at } t + 1 \\
& \quad \text{at } t + 2 \\
\end{align*}
\]

\[
\begin{align*}
\beta_j(t + 1) & \quad \text{at } t + 2 \\
& \quad \text{at } t + 1 \\
& \quad \text{at } t \\
& \quad \text{at } t - 1 \\
\end{align*}
\]

\[
\begin{align*}
\alpha_{ij} b_j(\sigma_{t+1}) & \quad \text{between } S_i \text{ and } S_j \\
& \quad \text{between } S_j \text{ and } S_i \\
\end{align*}
\]
Expecting and maximizing

• If we knew $\theta$, we could estimate **expectations** such as
  • Expected number of times in state $s_i$,
  • Expected number of transitions $s_i \rightarrow s_j$

• If we knew:
  • Expected number of times in state $s_i$,
  • Expected number of transitions $s_i \rightarrow s_j$
then we could compute the **maximum likelihood estimate** of

$$
\theta = \langle \{a_{ij}\}, \{b_{i}(w)\}, \pi_i \rangle
$$
BW M-step

We update our parameters as if we were doing MLE:

I. Initial-state probabilities:
\[ \hat{\pi}_i = \gamma_i(0) \quad \text{for } i := 1..N \]

II. State-transition probabilities:
\[ \hat{\alpha}_{ij} = \frac{\sum_{t=0}^{T-2} \xi_{ij}(t)}{\sum_{t=0}^{T-2} \gamma_i(t)} \quad \text{for } i, j := 1..N \]

III. Discrete observation probabilities:
\[ \hat{b}_j(w) = \frac{\sum_{t=0}^{T-1} \gamma_j(t)|\sigma_t=w}{\sum_{t=0}^{T-1} \gamma_j(t)} \quad \text{for } j := 1..N \text{ and } w \in \mathcal{V} \]
Baum-Welch iteration

- We update our parameters after **each iteration**
  \[
  \theta_{k+1} = \langle \hat{\pi}_i, \hat{\alpha}_{ij}, \hat{b}_j(w) \rangle
  \]
  rinse, and repeat until \( \theta_k \approx \theta_{k+1} \) (until change almost stops).

- Baum proved that
  \[
  P(\mathcal{O}; \theta_{k+1}) \geq P(\mathcal{O}; \theta_k)
  \]
  although this method does **not** guarantee a \textit{global maximum}.
Features of Baum-Welch

• Although we’re not guaranteed to achieve a global optimum, the local optima are often ‘good enough’.

• BW does not estimate the number of states, which must be ‘known’ beforehand.
  • Moreover, some constraints on topology are often imposed beforehand to assist training.
Discrete vs. continuous

• If our observations are drawn from a **continuous** space (e.g., speech acoustics), the probabilities $b_i(X)$ must also be continuous.

• HMMs **generalize** to continuous distributions, or multivariate observations, e.g., $b_i([-14.28, 0.85, 0.21])$. 
Adaptation

• It can take a **LOT** of data to train HMMs.
• Imagine that we’re given a **trained** HMM but not the data.
  • Also imagine that this HMM has been trained with data from **many** sources (e.g., many speakers).

• We want to use this HMM with a **particular new source** for whom we have **some** data (but not enough to fully train the HMM properly from scratch).
  • To be **more accurate for that source**, we want to **change** the original HMM parameters **slightly** given the new data.
HMM interpolation

• For added robustness, we can combine estimates of a generic HMM, $G$, trained with lots of data from many sources with a specific HMM, $S$, trained with a little data from a single source.

$$P_{\text{Interp}}(\sigma) = \lambda P(\sigma; \theta_G) + (1 - \lambda)P(\sigma; \theta_S)$$

• This gives us a model tuned to our target source ($S$), but with some general ‘knowledge’ ($G$) built in.
  • How do we pick $\lambda \in [0..1]$?
EM for interpolated models

- Strategy can be used for any $P(O; \lambda) = \sum_i \lambda_i P_i(O)$
- Introduce latent states $s$ such that $P(s = i; \lambda) = \lambda_i$
- Once in state $i$, $P(O|s = i; \lambda) = P_i(O)$
- Like with HMMs, we estimate $\text{Count}(s = i)$ using EM:
  \[
  \lambda_i^{\text{new}} = \frac{P(s = i, O; \lambda^{old})}{P(O; \lambda^{old})}
  \]

- This is a (simplified) version of what is done for Jelinek-Mercer interpolation, as well as Gaussian Mixture Models (covered in ASR lecture)
Held-out data

• Let $T_\lambda = \{O\}$ be the data used to learn $\lambda$, $T_i$ for $P_i(\cdot)$
• If for most $O \in T_\lambda, j. P_i(O) \geq P_j(O)$, then $\lambda_i \rightarrow 1$
• This can easily occur when $T_i = T_\lambda$, e.g.:
  • If $P_i(\cdot)$ is an MLE $i$-gram model trained on $T_\lambda$, it will outperform $P_{<i}(\cdot)$ (even if also trained on $T_\lambda$)
  • If $P(\sigma; \theta_S)$ was trained on $T_\lambda$ but not $P(\sigma; \theta_G)$
• Less likely to happen when $T_i \cap T_\lambda = \emptyset$
• A disjoint $T_\lambda$ is often called held-out or development data
Aside – Maximum a Posteriori (MAP)

• Given adaptation data $O_a$, the MAP estimate is
  \[ \hat{\theta} = \arg\max_{\theta} P(O_a | \theta) P(\theta) \]

• If we can guess some structure for $P(\theta)$, we can use EM to estimate new parameters (or Monte Carlo).

• For continuous $b_i(\omega)$, we use Dirichlet distribution that defines the hyper-parameters of the model and the Lagrange method to describe the change in parameters $\theta \Rightarrow \hat{\theta}$.
Summary

• Important ideas to know:
  • The definition of an HMM (e.g., its parameters).
  • The purpose of the Forward algorithm.
    • How to compute $\alpha_i(t)$ and $\beta_i(t)$
  • The purpose of the Viterbi algorithm.
    • How to compute $\delta_i(t)$ and $\psi_i(t)$.
  • The purpose of the Baum-Welch algorithm.
    • Some understanding of EM.
    • Some understanding of the equations.
Generative vs. discriminative

- HMMs are **generative** classifiers. You can generate synthetic samples from because they model the phenomenon itself. (e.g. \( P(\mathcal{O}, Q; \theta) \) or \( P(\mathcal{O}; \theta) \))
- Other classifiers (e.g., artificial neural networks and support vector machines) are **discriminative** in that their probabilities are trained specifically to reduce the error in classification. (e.g. \( P(Q | \mathcal{O}; \theta) \))
Reading

• (optional) Manning & Schütze: Section 9.2—9.4.1
  • Note that they use another formulation...

• Rabiner, L. (1990) A Tutorial on Hidden Markov Models and
  Selected Applications in Speech Recognition. In: Readings
  in speech recognition. Morgan Kaufmann.
  (posted on course website)

• Optional software:
  • Hidden Markov Model Toolkit (http://htk.eng.cam.ac.uk/)