Announcements
• Hidden Markov models:

$$p(s_1, \ldots, s_N, x_1, \ldots, x_N) = p(s_1)p(x_1 | s_1) \prod_{t=2}^{N} p(s_t | s_{t-1}) p(x_t | s_t).$$

• Parameters:
  
  – initial state probabilities $p(s_1)$,
  
  – state transition probability matrix $P$, $P_{ij} = p(s_t = i \mid s_{t-1} = j) = p(i \rightarrow j)$,
  
  – Emission probabilities $p(x_t | s_t)$. 
HMM: three fundamental problems

- Inference regarding observation sequence $x_1, \ldots, x_N$
  - Compute likelihood of a model given observations $\Rightarrow$ the forward-backward algorithm
  - Sample from the model

- Inference regarding hidden states
  - Estimate most likely state sequence $\Rightarrow$ the Viterbi algorithm

- Estimating the model parameters $\Rightarrow$ EM (Baum-Welch algorithm)
Forward-backward probabilities

- **Forward probabilities:**

  \[ \alpha_t(s) \triangleq p(x_1, \ldots, x_t, s_t = s) \]

- **Prediction:** current state given the past

  \[ \frac{\alpha_t(s)}{\sum_{s'} \alpha_t(s')} = p(s_t = s | x_1, \ldots, x_t) \]

- **Backward probabilities:** diagnostic (future given the state)

  \[ \beta_t(s) \triangleq p(x_{t+1}, \ldots, x_N | s_t = s) \]
Computing forward probabilities: \( t = 1 \)

\[
\alpha_1(1) = p(x_1, s_1 = 1)
\]
Computing forward probabilities: \( t = 1 \)

\[
\alpha_1(1) = p(x_1, s_1 = 1) = p_0(1)p(x_1 \mid s_1 = 1);
\]
Computing forward probabilities: $t = 1$

$$\alpha_1(1) = p(x_1, s_1 = 1) = p_0(1)p(x_1 | s_1 = 1);$$

$$\alpha_1(2) = p(x_1, s_1 = 2) = p_0(2)p(x_1 | s_1 = 2);$$
Computing forward probabilities: \( t = 2 \)

\[
\alpha_1(1) = p(x_1, s_1 = 1) = p_0(1)p(x_1 | s_1 = 1), \\
\alpha_1(2) = p(x_1, s_1 = 2) = p_0(2)p(x_1 | s_1 = 2)
\]

\[
\alpha_2(1) = p(x_1, x_2, s_2 = 1)
\]
Computing forward probabilities: \( t = 2 \)

\[ \alpha_1(1) = p(x_1, s_1 = 1) = p_0(1)p(x_1 | s_1 = 1), \]
\[ \alpha_1(2) = p(x_1, s_1 = 2) = p_0(2)p(x_1 | s_1 = 2) \]

\[ \alpha_2(1) = p(x_1, x_2, s_2 = 1) = p(x_1, s_2 = 1)p(x_2 | s_2 = 1) \]
Computing forward probabilities: $t = 2$

$$
\alpha_1(1) = p(x_1, s_1 = 1) = p_0(1) p(x_1 | s_1 = 1),
$$

$$
\alpha_1(2) = p(x_1, s_1 = 2) = p_0(2) p(x_1 | s_1 = 2)
$$

$$
\alpha_2(1) = p(x_1, x_2, s_2 = 1) = p(x_1, s_2 = 1) p(x_2 | s_2 = 1)
$$

$$
= [\alpha_1(1) p(1 \rightarrow 1) + \alpha_1(2) p(2 \rightarrow 1)] p(x_2 | s_2 = 1)
$$
Computing forward probabilities: \( t = 2 \)

\[
\alpha_1(1) = p(x_1, s_1 = 1) = p_0(1)p(x_1 | s_1 = 1),
\]
\[
\alpha_1(2) = p(x_1, s_1 = 2) = p_0(2)p(x_1 | s_1 = 2)
\]

\[
\alpha_2(1) = p(x_1, x_2, s_2 = 1) = p(x_1, s_2 = 1)p(x_2 | s_2 = 1)
\]
\[
= [\alpha_1(1)p(1 \rightarrow 1) + \alpha_1(2)p(2 \rightarrow 1)]p(x_2 | s_2 = 1)
\]
\[
\alpha_2(2) = [\alpha_1(2)p(1 \rightarrow 2) + \alpha_1(2)p(2 \rightarrow 2)]p(x_2 | s_2 = 2)
\]
Forward probabilities: recursion

\[ \alpha_t(s) = p(x_1, \ldots, x_t, s_t = s) \]

\[ \alpha_1(s) = p_0(s)p(x_1 | s_1 = s) \]
Forward probabilities: recursion

\[ \alpha_t(s) = p(x_1, \ldots, x_t, s_t = s) \]

\[ \alpha_1(s) = p_0(s)p(x_1 | s_1 = s) \]

\[ \alpha_t(s) = \left[ \sum_{s'} \alpha_{t-1}(s')p(s' \rightarrow s) \right] p(x_t | s_t = s) \]
Backward probabilities

\[ \beta_t(s) \triangleq p(x_{t+1}, \ldots, x_N \mid s_t = s) \]

\[ \beta_N(s) = p(\emptyset \mid s_N = s) \triangleq 1 \]
Backward probabilities

\[ \beta_t(s) \triangleq p(x_{t+1}, \ldots, x_N \mid s_t = s) \]

\[ \beta_N(s) = p(\emptyset \mid s_N = s) \triangleq 1 \]

\[ \beta_t(s) = \sum_{s'} [p(s \rightarrow s') p(x_{t+1} \mid s_{t+1} = s') \beta_{t+1}(s')] \]
Bugs on a Grid

- Naïve algorithm\(^1\):

1. start bug in each state at \( t = 1 \) holding value 0;
2. move each bug forward in time by making copies of it multiplying the value of each copy by the probability of the transition and output emission;
3. go to 2 until all bugs have reached time \( N \);
4. sum up values on all bugs.

\(^1\)from S. Roweis
Bugs on a Grid - Trick

- Clever recursion:
  adds a step between 2 and 3 above which says: at each node, replace all the bugs with a single bug carrying the sum of their values

- This is exactly dynamic programming.
Bugs on a Grid - Trick

- Clever recursion:
  adds a step between 2 and 3 above which says: at each node, replace all the bugs with a single bug carrying the sum of their values

- This is exactly dynamic programming.
Forward-Backward algorithm

\[ \alpha_1(s) = p_0(s)p(x_1 | s_1 = s), \]

\[ \alpha_t(s) = \left[ \sum_{s'} \alpha_{t-1}(s')p(s' \rightarrow s) \right] p(x_t | s_t = s); \]

\[ \beta_N(s) = 1, \]

\[ \beta_t(s) = \sum_{s'} [p(s \rightarrow s')p(x_{t+1} | s_{t+1} = s') \beta_{t+1}(s')]. \]

- We need two passes (forward and backward) to compute \( \alpha_t, \beta_t \) for all \( t = 1, \ldots, N \)

- Time complexity with \( M \) states:
Forward-Backward algorithm

\[ \alpha_1(s) = p_0(s)p(x_1 | s_1 = s), \]
\[ \alpha_t(s) = \left[ \sum_{s'} \alpha_{t-1}(s')p(s' \rightarrow s) \right] p(x_t | s_t = s); \]
\[ \beta_N(s) = 1, \]
\[ \beta_t(s) = \sum_{s'} [p(s \rightarrow s')p(x_{t+1} | s_{t+1} = s') \beta_{t+1}(s')]. \]

- We need two passes (forward and backward) to compute \( \alpha_t, \beta_t \) for all \( t = 1, \ldots, N \)

- Time complexity with \( M \) states: \( O(NM^2) \).
Computing likelihood of observations

\[
\alpha_t(s) = p(x_1, \ldots, x_t, s_t = s) \\
\beta_t(s) = p(x_{t+1}, \ldots, x_N | s_t = s)
\]

- We can compute likelihood directly from the forward/backward probabilities:

\[
p(x_1, \ldots, x_N) = \sum_s p(x_1, \ldots, x_N, s_t = s) \quad \text{(for any } t \in \{1, \ldots, N\} \text{ )}
\]
Computing likelihood of observations

\[ \alpha_t(s) = p(x_1, \ldots, x_t, s_t = s) \]
\[ \beta_t(s) = p(x_{t+1}, \ldots, x_N | s_t = s) \]

- We can compute likelihood directly from the forward/backward probabilities:

\[
p(x_1, \ldots, x_N) = \sum_s p(x_1, \ldots, x_N, s_t = s) \quad \text{(for any } t \in \{1, \ldots, N\} \text{)}
\]
\[
= \sum_s p(x_1, \ldots, x_t, s_t = s)p(x_{t+1}, \ldots, x_N | s_t = s, x_1, \ldots, x_t)
\]
Computing likelihood of observations

\( \alpha_t(s) = p(x_1, \ldots, x_t, s_t = s) \)

\( \beta_t(s) = p(x_{t+1}, \ldots, x_N | s_t = s) \)

- We can compute likelihood directly from the forward/backward probabilities:

\[
p(x_1, \ldots, x_N) = \sum_s p(x_1, \ldots, x_N, s_t = s) \quad \text{(for any } t \in \{1, \ldots, N\} \text{ )}
\]

\[
= \sum_s p(x_1, \ldots, x_t, s_t = s)p(x_{t+1}, \ldots, x_N | s_t = s, x_1, \ldots, x_t)
\]

\[
= \sum_s p(x_1, \ldots, x_t, s_t = s)p(x_{t+1}, \ldots, x_N | s_t = s)
\]
Computing likelihood of observations

\[
\alpha_t(s) = p(x_1, \ldots, x_t, s_t = s) \\
\beta_t(s) = p(x_{t+1}, \ldots, x_N | s_t = s)
\]

- We can compute likelihood directly from the forward/backward probabilities:

\[
p(x_1, \ldots, x_N) = \sum_s p(x_1, \ldots, x_N, s_t = s) \quad (\text{for any } t \in \{1, \ldots, N\})
\]

\[
= \sum_s p(x_1, \ldots, x_t, s_t = s)p(x_{t+1}, \ldots, x_N | s_t = s, x_1, \ldots, x_t)
\]

\[
= \sum_s p(x_1, \ldots, x_t, s_t = s)p(x_{t+1}, \ldots, x_N | s_t = s)
\]

\[
= \sum_s \alpha_t(s) \beta_t(s)
\]
Likelihood of observations

\[ p(x_1, \ldots, x_N) = \sum_{s} \alpha_t(s) \beta_t(s) \quad \text{(for any } t) \]

\[ = \sum_{s} \alpha_N(s) \]
State posterior probability

\[ \gamma_t(s) \triangleq p(s_t = s \mid x_1, \ldots, x_N) \]
\[ = \frac{p(x_1, \ldots, x_N, s_t = s)}{p(x_1, \ldots, x_N)} \]
State posterior probability

\[ \gamma_t(s) \triangleq p(s_t = s \mid x_1, \ldots, x_N) \]

\[ = \frac{p(x_1, \ldots, x_{t+1}, \ldots, x_N \mid s_t = s)}{p(x_1, \ldots, x_N)} \]

\[ = \frac{\alpha_t(s)}{p(x_1, \ldots, x_t, s_t = s)} p(x_{t+1}, \ldots, x_N \mid s_t = s) \]

\[ = \frac{\beta_t(s)}{p(x_1, \ldots, x_N)} \]
State posterior probability

\[ \gamma_t(s) \triangleq p(s_t = s \mid x_1, \ldots, x_N) = \frac{p(x_1, \ldots, x_N, s_t = s)}{p(x_1, \ldots, x_N)} \]

\[
= \frac{\alpha_t(s)}{p(x_1, \ldots, x_t, s_t = s)} \cdot \frac{\beta_t(s)}{p(x_{t+1}, \ldots, x_N \mid s_t = s)} \cdot \frac{p(x_1, \ldots, x_N)}{p(x_1, \ldots, x_N)} \]

\[ = \frac{\alpha_t(s) \beta_t(s)}{\sum' s' \alpha_t(s') \beta_t(s')} \]
Posterior probability of a transition

\[ \xi_t(i, j) \triangleq p(s_t = i, s_{t+1} = j \mid x_1, \ldots, x_N) \]

\[ = \frac{p(x_1, \ldots, x_N, s_t = i, s_{t+1} = j)}{p(x_1, \ldots, x_N)} \]
Posterior probability of a transition

\[ \xi_t(i, j) \triangleq p(s_t = i, s_{t+1} = j \mid x_1, \ldots, x_N) = \frac{p(x_1, \ldots, x_N, s_t = i, s_{t+1} = j)}{p(x_1, \ldots, x_N)} \]

\[ = \frac{\alpha_t(i) p(s_{t+1} = j \mid s_t = i) p(x_{t+1} \mid s_{t+1} = j) \beta_{t+1}(j)}{\sum' \alpha_t(s') \beta_t(s')} \]
HMM: three fundamental problems

• Inference regarding observation sequence \( x_1, \ldots, x_N \)
  - Compute likelihood of a model given observations  \( \Rightarrow \) the forward-backward algorithm
  - Sample from the model

• Inference regarding hidden states
  - Estimate most likely state sequence  \( \Rightarrow \) the Viterbi algorithm

• Estimating the model parameters
  \( \Rightarrow \) EM (Baum-Welch algorithm)
The EM (Baum-Welch) algorithm for HMM

- Assume we have $L$ sequences with $n_1, \ldots, n_L$ observations;

\[ X^{(l)} = x^{(l)}_1, \ldots, x^{(l)}_{n_l}. \]

- Start with a random (but intelligent) guess for $\theta = \{p(s_1), p(i \to j)p(x \mid s)\}$.

- **E-step**: use current estimates of $\theta$, and compute the “responsibilities”

\[ \gamma_t^{(l)}(s) \quad \text{for all } s = 1, \ldots, M, \ l = 1, \ldots, L, \ \text{and } t = 1, \ldots, n_l; \]
\[ \xi_t^{(l)}(i, j) \quad \text{for all } i, j = 1, \ldots, M, \ l = 1, \ldots, L, \ \text{and } t = 1, \ldots, n_l. \]
EM: the M-step

- First, update

\[ p^{new}(s_1 = s) = \frac{1}{L} \sum_{l=1}^{L} \gamma_{1}^{(l)}(s) \]

Second, compute weighted “transition counts”

\[ \hat{n}_{i,j} = \sum_{l=1}^{L} \sum_{t=1}^{n_{l}-1} \xi_{t}^{(l)}(i, j) \]

\[ \Rightarrow p^{new}(i \to j) = \frac{\hat{n}_{i,j}}{\sum_{j'} \hat{n}_{i,j'}}. \]
M-step for emission probabilities

- Parametric form $p(x_t | s_t = s) = p(x_t; \theta_s)$
  - M. of Gaussians HMM with $k$ components:
    $$\theta_s = \{\mu_{s1}, \Sigma_{s1}, \ldots, \mu_{sk}, \Sigma_{sk}, p_1, \ldots, p_k\}$$
  - Multinomial HMM, e.g., $k$ symbols: $\theta_s = [\theta_{s1}, \ldots, \theta_{sk}]^T$

- We solve separately for each state $s$ the weighted ML problem:
  $$\theta_s^{new} = \arg\min_{\theta_s} \sum_{l=1}^{L} \sum_{t=1}^{n_l-1} \gamma^{(l)}(s) \log p(x_t^{(l)}; \theta_s)$$
Next time

HMM decoding;
Variants of HMM.