

COMS W4701: Artificial Intelligence

Lecture 7b: Smoothing and Learning HMMs

Tony Dear, Ph.D.

Department of Computer Science

School of Engineering and Applied Sciences

Today

- Smoothing: Forward-backward algorithm
- Supervised learning: Maximum likelihood
- Unsupervised learning: Baum-Welch algorithm

Inference

- Inference tasks compute belief states or hidden states given evidence
- **Filtering (state estimation):** Find $P(X_t | e_{1:t})$
 - Estimate the belief state, given a sequence of past observations
- **Smoothing:** Find $P(X_k | e_{1:N})$, for $1 \leq k < N$
 - Use both past and future evidence to *smooth* a belief state
- **Learning:** Learn the parameters $P(X_0)$, $P(X_t | X_{t+1})$, $P(E_t | X_t)$ from data

Smoothing

- Suppose we have evidence $e_{1:N}$ and we want to go “back” in time to compute “more informed” belief states $P(X_k|e_{1:N})$, $1 \leq k < N$

- We can derive $P(X_k|e_{1:N}) = P(X_k, e_{1:N})$ using the product rule:

$$P(X_k, e_{1:k}, e_{k+1:N}) = P(X_k, e_{1:k})P(e_{k+1:N}|X_k, e_{1:k}) = \alpha_k * \beta_k$$

Conditional
independence

- We have α_k from the forward algorithm
- β_k is the joint probability of “future” evidence, conditioned on state at k
- We take the elementwise product of α_k and β_k arrays

Backward Algorithm

- As with the forward algorithm, we *iteratively* compute β_k but going *backward*
- Let $\beta_{k+1} = P(e_{k+2:N} | X_{k+1})$, $\beta'_{k+1} = P(e_{k+1:N} | X_{k+1})$, and $\beta_k = P(e_{k+1:N} | X_k)$
- Let $\beta_N = (1, \dots, 1)$ (algorithm base case)
- Given X_{k+1} , E_{k+1} is conditionally independent of E_{k+2}, \dots, E_t

$$P(e_{k+1:N} | X_{k+1}) = P(e_{k+1} | X_{k+1}) P(e_{k+2:N} | X_{k+1}) \quad \boxed{\beta'_{k+1} = \beta_{k+1} * O_{k+1}}$$

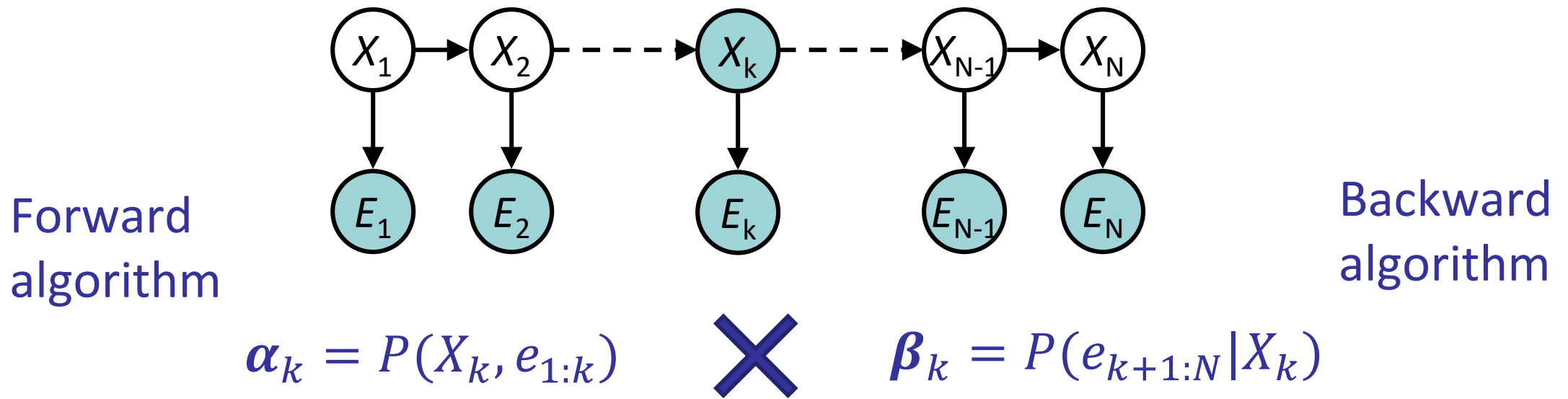
- Elapse time *backward*:

$$P(e_{k+1:N} | X_k) = \sum_{x_{k+1}} P(e_{k+1:N} | x_{k+1}, \cancel{X_k}) P(x_{k+1} | X_k) \quad \boxed{\beta_k = \beta'_{k+1} T^\top}$$

Conditional independence

Forward-Backward Algorithm

- The **forward-backward algorithm** computes $P(X_k|e_{1:N})$ for all k
 - Use forward algorithm to compute and store α_k for all k
 - Use backward algorithm to compute β_k for all k
 - $P(X_k|e_{1:N}) \propto \alpha_k * \beta_k$: take elementwise product and normalize



Example: Weather HMM

$$T = \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix} \begin{matrix} +r \\ -r \\ +r \\ -r \end{matrix}$$

$$\begin{aligned} \beta'_{k+1} &= \beta_{k+1} * O_{k+1} \\ \beta_k &= \beta'_{k+1} T^\top \end{aligned}$$

$$\begin{aligned} O_1 = O_3 &= \begin{pmatrix} 0.1 \\ 0.8 \end{pmatrix}^\top \\ O_2 &= \begin{pmatrix} 0.9 \\ 0.2 \end{pmatrix}^\top \end{aligned}$$

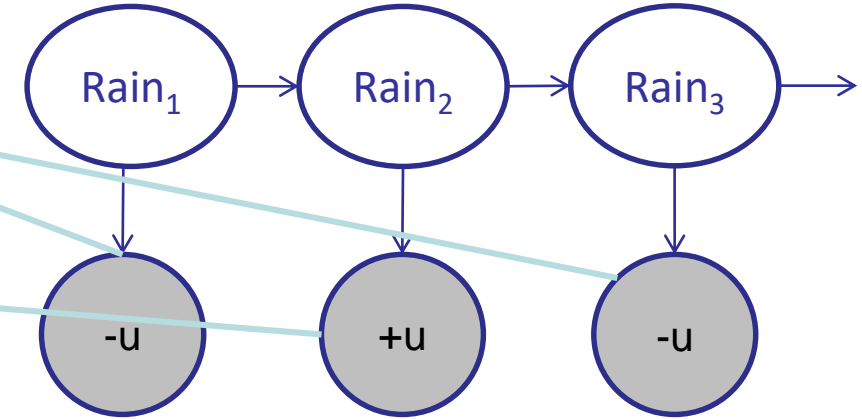
$$\text{Initialize } \beta_3 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}^\top$$

$$\beta'_3 = \beta_3 * O_3 = \begin{pmatrix} 0.1 \\ 0.8 \end{pmatrix}^\top$$

$$\beta'_2 = \beta_2 * O_2 = \begin{pmatrix} .279 \\ .118 \end{pmatrix}^\top$$

$$\beta_2 = \beta'_3 T^\top = P(e_3 | X_2) = \begin{pmatrix} 0.31 \\ 0.59 \end{pmatrix}^\top$$

$$\beta_1 = \beta'_2 T^\top = P(e_{2:3} | X_1) = \begin{pmatrix} .2307 \\ .1663 \end{pmatrix}^\top$$



Example: Weather HMM

$$\alpha_1 = \begin{pmatrix} 0.05 \\ 0.4 \end{pmatrix}$$

$$\alpha_2 = \begin{pmatrix} .1395 \\ .059 \end{pmatrix}$$

$$\alpha_3 = \begin{pmatrix} .0115 \\ .0665 \end{pmatrix}$$

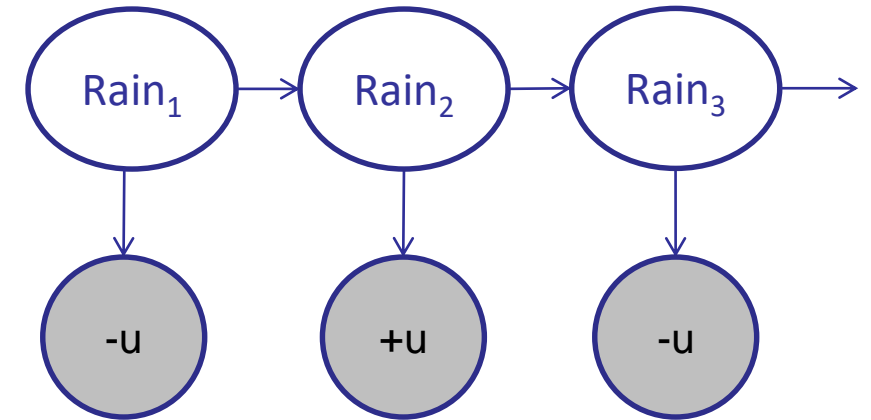
normalize



$$P(X_1|e_1) = \begin{pmatrix} 0.11 \\ 0.89 \end{pmatrix}$$

$$P(X_2|e_{1:2}) = \begin{pmatrix} .703 \\ .297 \end{pmatrix}$$

$$P(X_3|e_{1:3}) = \begin{pmatrix} .148 \\ .852 \end{pmatrix}$$



$$\beta_1 = \begin{pmatrix} .2307 \\ .1663 \end{pmatrix}$$

$$\beta_2 = \begin{pmatrix} 0.31 \\ 0.59 \end{pmatrix}$$

$$\beta_3 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

multiply



$$\alpha_1 * \beta_1 = \begin{pmatrix} .012 \\ .067 \end{pmatrix}$$

$$\alpha_2 * \beta_2 = \begin{pmatrix} .043 \\ .035 \end{pmatrix}$$

$$\alpha_3 * \beta_3 = \begin{pmatrix} .0115 \\ .0664 \end{pmatrix}$$

normalize



$$P(X_1|e_{1:3}) = \begin{pmatrix} .148 \\ .852 \end{pmatrix}$$

$$P(X_2|e_{1:3}) = \begin{pmatrix} .554 \\ .446 \end{pmatrix}$$

$$P(X_3|e_{1:3}) = \begin{pmatrix} .148 \\ .852 \end{pmatrix}$$

Supervised Learning

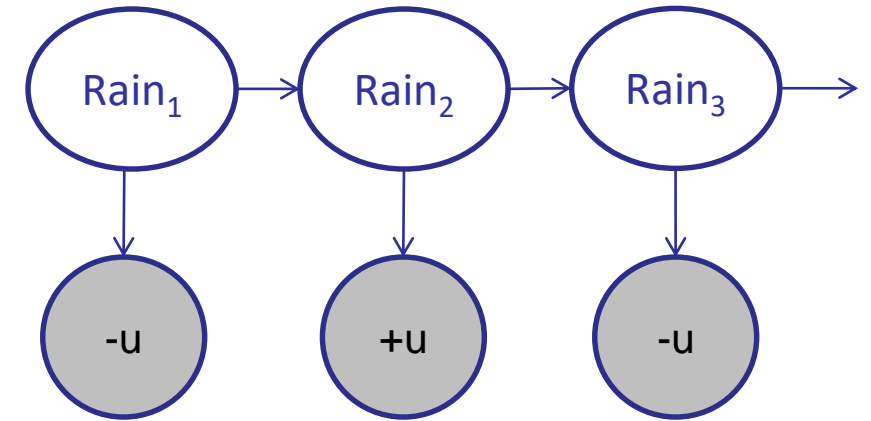
- We now want to *learn* the *parameters* θ (probabilities) of a HMM from data
- Suppose we have a sequence of states and observations: $\mathbf{d} = ((x_1, e_1), \dots, (x_N, e_N))$
- **Maximum-likelihood learning:** Find parameters that maximize likelihood of data
- Likelihood is joint probability of observed data using transition/observation models:

$$\max_{\theta} \Pr(\mathbf{d}|\theta) = \max_{P(X_0), P(X_{i+1}|X_i), P(E_i|X_i)} \left\{ P(x_0) \prod_{i=1}^N P(x_{i+1}|x_i) P(e_i | x_i) \right\}$$

- Intuitive solution: We can simply count the *proportions* of transitions $x_i \rightarrow x_{i+1}$ and *proportions* of observations (x_i, e_i) in our data

Example: Weather HMM

- Given (state, observation) sequences:
- $(+r, +u), (+r, -u), (-r, -u), (+r, +u)$
- $(-r, -u), (-r, +u), (+r, +u)$
- $(+r, -u), (+r, +u), (-r, -u), (-r, +u), (+r, +u)$
- Number of initial states: 2 $+r$, 1 $-r$
- Transitions: 2 $+r$ to $+r$, 2 $+r$ to $-r$, 3 $-r$ to $+r$, 2 $-r$ to $-r$
- Observations: 5 $+r$ to $+u$, 2 $+r$ to $-u$, 2 $-r$ to $+u$, 3 $-r$ to $-u$



$$P(\widehat{U} | +r) = \begin{pmatrix} 5/7 \\ 2/7 \end{pmatrix}$$

$$P(\widehat{X_0}) = \begin{pmatrix} 2/3 \\ 1/3 \end{pmatrix} \quad \hat{T} = \begin{pmatrix} 2/4 & 2/4 \\ 3/5 & 2/5 \end{pmatrix}$$

$$P(\widehat{U} | -r) = \begin{pmatrix} 2/5 \\ 3/5 \end{pmatrix}$$

Unsupervised Learning

- Problem: State information is generally hidden in HMM problems
- We may only see the observations in a data set
- Suppose we have an *initial guess* about the HMM parameters
- We can compute $\gamma_k = P(X_k | e_{1:N})$ from forward-backward algorithm
- γ_k is the *expected* number of occurrences of each state value x at time k
- We can use γ_k in place of “true counts” to update HMM parameters!
- New initial distribution: $\widehat{P}(X_0) = \gamma_0 = P(X_0 | e_{1:N})$

Expected Observations

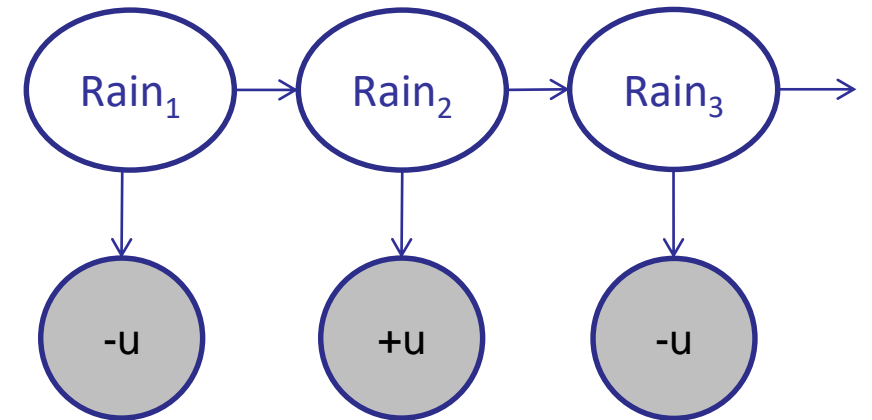
- Our “data set” now looks like $((\gamma_1, e_1), (\gamma_2, e_2), \dots, (\gamma_N, e_N))$
- New estimate of $P(e|x)$: number of (x, e) / total number of x

$$\widehat{P}(e|x) = \frac{\sum_{e_k=e} \gamma_k(x)}{\sum_{k=1}^N \gamma_k(x)}$$

$$o_1 = o_3 = \begin{pmatrix} 0.1 \\ 0.8 \end{pmatrix} \quad o_2 = \begin{pmatrix} 0.9 \\ 0.2 \end{pmatrix}$$

$$\gamma_0 = \begin{pmatrix} 0.36 \\ 0.64 \end{pmatrix} \quad \gamma_1 = \begin{pmatrix} .148 \\ .852 \end{pmatrix} \quad \gamma_2 = \begin{pmatrix} .554 \\ .446 \end{pmatrix} \quad \gamma_3 = \begin{pmatrix} .148 \\ .852 \end{pmatrix}$$

$$\widehat{P}(X_0) = \gamma_0 = \begin{pmatrix} 0.36 \\ 0.64 \end{pmatrix}$$
$$\widehat{P}(+u|X) = \frac{\gamma_2}{\gamma_1 + \gamma_2 + \gamma_3} = \begin{pmatrix} .652 \\ .207 \end{pmatrix}$$
$$\widehat{P}(-u|X) = \frac{\gamma_1 + \gamma_3}{\gamma_1 + \gamma_2 + \gamma_3} = \begin{pmatrix} .348 \\ .793 \end{pmatrix}$$



Expected Transitions

- In the new “data set” $((\boldsymbol{\gamma}_1, e_1), (\boldsymbol{\gamma}_2, e_2), \dots, (\boldsymbol{\gamma}_N, e_N))$, there are now potential transitions between every pair of states in every timestep
- We can compute the *expected* transitions from a state at step k to a state at step $k + 1$

$$\begin{aligned} P(x_k, x_{k+1} | e_{1:N}) &\propto P(x_k, x_{k+1}, e_{1:N}) \\ &= P(x_k, e_{1:k}) P(x_{k+1} | x_k, e_{1:k}) P(e_{k+1} | x_{k+1}, x_k, e_{1:k}) P(e_{k+2:N} | x_{k+1}, x_k, e_{1:k+1}) \\ &= \boldsymbol{\alpha}_k(x_k) P(x_{k+1} | x_k) P(e_{k+1} | x_{k+1}) \boldsymbol{\beta}_{k+1}(x_{k+1}) \end{aligned}$$

- *Expected* transition matrix: $\xi_k = \text{diag}(\boldsymbol{\alpha}_k) \cdot \boldsymbol{T} \cdot \text{diag}(\boldsymbol{O}_{k+1}) \cdot \text{diag}(\boldsymbol{\beta}_{k+1})$

Expected Transition Matrix

- *Total number of expected transitions between states is $\sum \xi_k$*
- *We can normalize each row to obtain new transition probabilities!*

$$P(\widehat{X_{t+1}} | \widehat{X_t} = i) \propto \sum_{k=0}^{N-1} (\xi_k)_i$$

$$\alpha_0 = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} \quad \alpha_1 = \begin{pmatrix} 0.05 \\ 0.4 \end{pmatrix} \quad \alpha_2 = \begin{pmatrix} .1395 \\ .059 \end{pmatrix}$$

$$\beta_1 = \begin{pmatrix} .2307 \\ .1663 \end{pmatrix} \quad \beta_2 = \begin{pmatrix} 0.31 \\ 0.59 \end{pmatrix} \quad \beta_3 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$T = \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix} \quad o_1 = o_3 = \begin{pmatrix} 0.1 \\ 0.8 \end{pmatrix} \quad o_2 = \begin{pmatrix} 0.9 \\ 0.2 \end{pmatrix}$$

$$\xi_0 = \text{diag}(\alpha_0) \cdot T \cdot \text{diag}(o_1) \cdot \text{diag}(\beta_1) = \begin{pmatrix} .008 & .020 \\ .003 & .047 \end{pmatrix}$$

$$\xi_1 = \text{diag}(\alpha_1) \cdot T \cdot \text{diag}(o_2) \cdot \text{diag}(\beta_2) = \begin{pmatrix} .010 & .002 \\ .033 & .033 \end{pmatrix}$$

$$\xi_2 = \text{diag}(\alpha_2) \cdot T \cdot \text{diag}(o_3) \cdot \text{diag}(\beta_3) = \begin{pmatrix} .010 & .033 \\ .002 & .033 \end{pmatrix}$$

$$\xi_0 + \xi_1 + \xi_2 = \begin{pmatrix} .028 & .056 \\ .039 & .113 \end{pmatrix}$$

\propto

$$P(\widehat{X_{k+1}} | X_k) = \begin{pmatrix} .333 & .667 \\ .256 & .744 \end{pmatrix}$$

Baum-Welch Algorithm

- **Baum-Welch** is an *expectation-maximization* algorithm: Learn HMM parameters by alternating between computing expected parameters and maximizing likelihoods
- Initialize HMM parameters $\widehat{P}(X_0)$, $\widehat{P}(X_{k+1}|X_k)$, $\widehat{P}(E|X)$
- Repeat until convergence:
 - **Expectation:** Compute $\alpha_k, \beta_k, \gamma_k, \xi_k \forall k$ using current parameters
 - **Maximization:** Compute new parameters $\widehat{P}(X_0)$, $\widehat{P}(X_{k+1}|X_k)$, $\widehat{P}(E|X)$
- The data likelihood $P(e_{1:N})$ will monotonically increase in each iteration
- Will typically reach a *local maximum*, dependent on parameter initialization

Summary

- Smoothing updates belief distributions using past and future observations
- Forward-backward algorithm runs linear in time and space
- Learning HMM parameters: If both hidden states and observations are given, perform supervised learning by counting occurrences
- If states are not given, perform unsupervised learning using Baum-Welch
- Expectation-maximization: Alternate between the two procedures until likelihood converges at local maximum