

COMP 424 - Artificial Intelligence

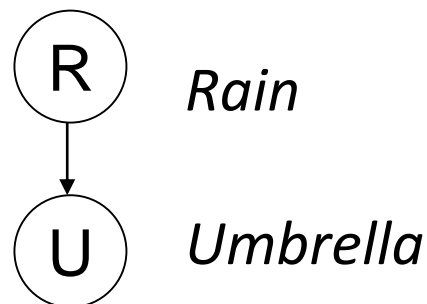
Lecture 18: Temporal Inference

Instructor: Jackie CK Cheung (jcheung@cs.mcgill.ca)

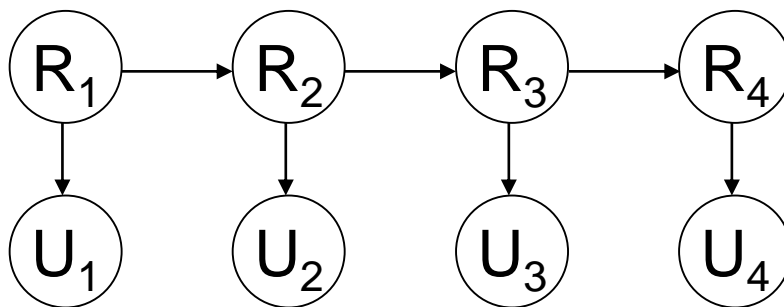
Readings: R&N Ch 15-15.3

Probabilistic graphical models

- Remember Bayes Nets?



- Today:** modelling changes in the world over time



Time and uncertainty

- Types of random variables:

X_t = set of **unobservable** (a.k.a., latent, hidden) state variables at time t

E_t = set of **observable** evidence variables at time t

- Assumptions:
 - Discrete time
 - Same structure at each timestep

What can this model?

Domain	Unobserved Variables	Observed Variables
Health care	State of health	Symptom or diagnostic
Language	Linguistic structure (e.g. verbs, nouns, adjectives)	Words
Speech recognition	Words	Acoustic signal
Robot navigation	Location	Sensory data
Financial markets	Financial health of company	Reports about the company

Notation

X_t = set of **unobservable** state variables at time t

E_t = set of **observable** evidence variables at time t

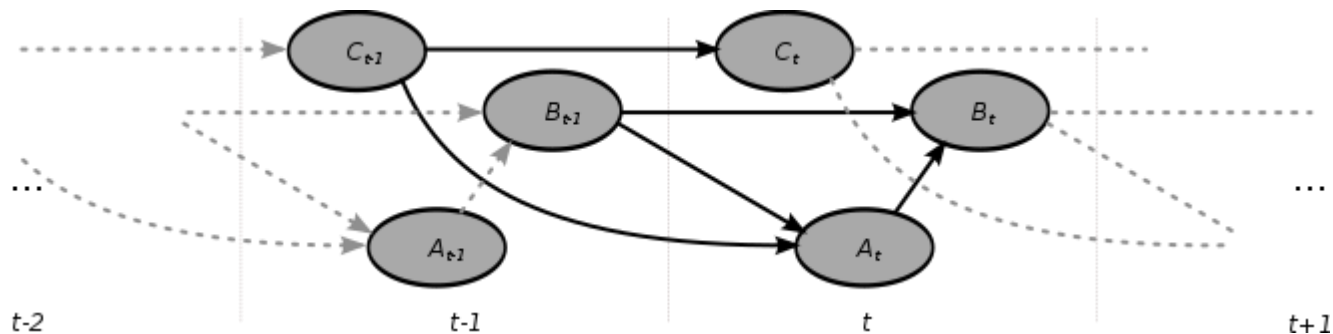
- Notation for time series sequence: $X_{t:t+k} = X_t, X_{t+1}, \dots, X_{t+k-1}, X_{t+k}$
- Changes in the state variable over time are determined by the **transition model**, $P(X_t \mid X_{0:t-1})$ (= “the probability of being in state $X=x$ at time t , conditioned on the previous states visited at times 0 to $t-1$ ”).
- In some cases, the state variable is (partially) observed through a **sensor model**, $P(E_t \mid X_{0:t}, E_{0:t-1})$
- These are defined over the full domains of X and E .

Markov processes (Markov chains)

- **Markov assumption:** X_t depends on bounded subset of $X_{0:t-1}$.
 - First-order Markov process: $P(X_t \mid X_{0:t-1}) = P(X_t \mid X_{t-1})$
 - Second-order Markov process: $P(X_t \mid X_{0:t-1}) = P(X_t \mid X_{t-2}, X_{t-1})$
- **Stationary process assumption:** transition and sensor models are fixed for all time steps
 - e.g., same conditional probability distributions for $P(X_t \mid X_{t-1})$ as for $P(X_{t+1} \mid X_t)$
- These assumptions are often false in the real world! But they are useful assumptions to make learning possible.

Dynamic Bayesian Network

- A Bayes Net whose structure is “copied” over time.
- Dependencies between random variables in neighbouring time steps.
- e.g., three-variable DBN



Wikipedia

Inference tasks in temporal models

1. Filtering: $P(X_t / e_{1:t})$

E.g. In text analysis, infer topic probability, based on observed words.

2. Prediction: $P(X_{t+k} / e_{1:t})$ for $k > 0$

E.g. In financial modeling, predict prices based on previous trends.

3. Smoothing: $P(X_k / e_{1:t})$ for $0 \leq k < t$

E.g. In genome analysis, produce plausible sequence alignments.

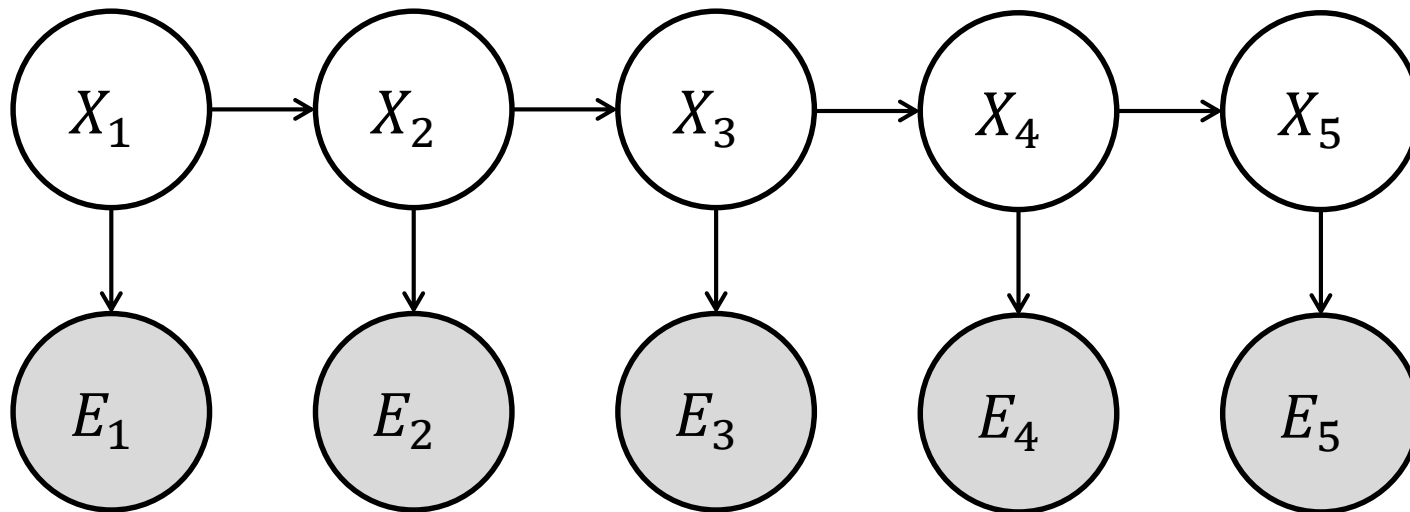
4. Most likely explanation: $\operatorname{argmax}_{X_{1:t}} P(X_{1:t} / e_{1:t})$

E.g. In speech recognition, infer most likely words, based on observed phonemes.

It is commonly assumed that X is the latent state and E is the evidence. In that case we treat X as *missing data*.

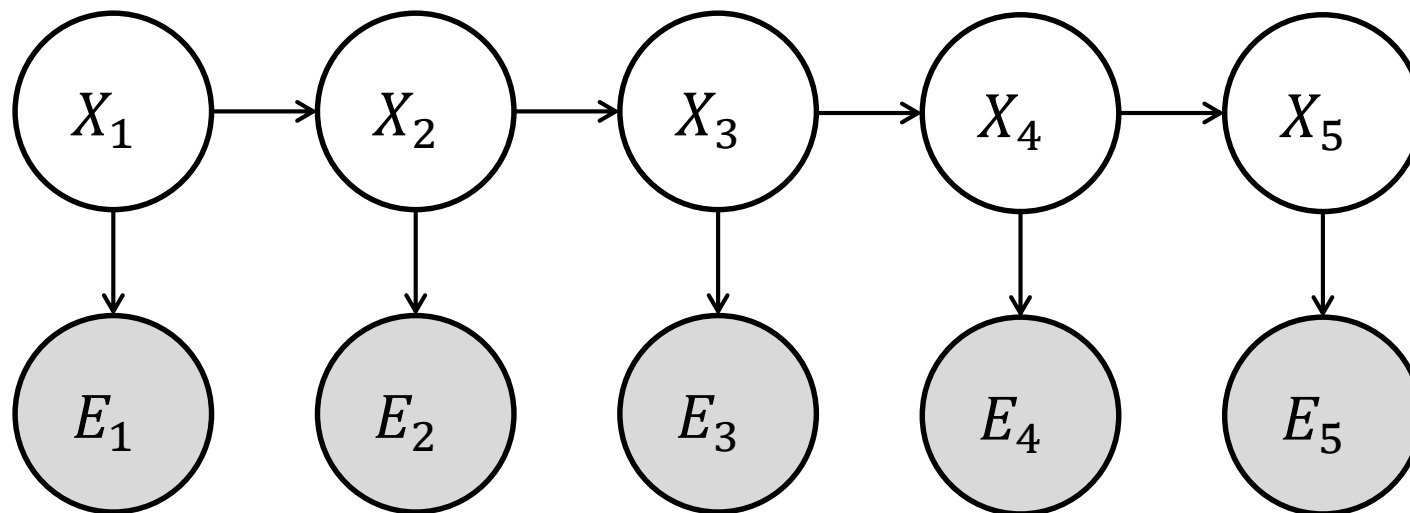
Simplest Case: Hidden Markov Model

- Let's assume that there is only one state random variable and one sensor variable per time step: simplest DBN!
- This is called a **Hidden Markov model**.
- e.g., with 5 time steps:



Decomposing the Joint Probability

- Recall from our discussion of Bayes Nets:



- $$P(\mathbf{X}, \mathbf{E}) = P(X_1) \prod_{t=1}^{T-1} P(X_{t+1}|X_t) \prod_{t=1}^T P(E_t|X_t)$$

Initial state probability

State transition probabilities

Emission probabilities

Model Parameters

- Let there be N possible states, W possible observations
- Parameters θ has three components:
 1. Initial probabilities for X_1 :
 - $\Pi = \{\pi_1, \pi_2, \dots, \pi_N\}$ (categorical)
 2. Transition probabilities for X_t to X_{t+1} :
 - $A = \{a_{ij}\} \ i, j \in [1, N]$ (categorical)
 3. Emission probabilities for X_t to E_t :
 - $B = \{b_i(w_k)\} \ i \in [1, N], k \in [1, W]$ (categorical)
- How many distributions and values of each type are there?

Review: Model Parameters' MLE

- Recall categorical distributions' MLE:
- $P(\text{outcome } i) = \frac{\#(\text{outcome } i)}{\#(\text{all events})}$
- For our parameters:
 - $\pi_i = P(X_1 = i) = \frac{\#(X_1=i)}{\#(\text{instances})}$
 - $a_{ij} = P(X_{t+1} = j | X_t = i) = \#(i, j) / \#(i)$
 - $b_{ik} = P(E_t = k | X_t = i) = \#(\text{outcome } k, \text{state } i) / \#(i)$

Inference tasks in temporal models

Forward algorithm, backward algorithm

1. Filtering: $P(X_t | e_{1:t})$

E.g. In text analysis, infer topic probability, based on observed words.

2. Prediction: $P(X_{t+k} | e_{1:t})$ for $k > 0$

E.g. In financial modeling, predict prices based on previous trends.

3. Smoothing: $P(X_k | e_{1:t})$ for $0 \leq k < t$

E.g. In genome analysis, produce plausible sequence alignments.

4. Most likely explanation: $\operatorname{argmax}_{X_{1:t}} P(X_{1:t} | e_{1:t})$

E.g. In speech recognition, infer most likely words, based on observed phonemes.

Viterbi algorithm

It is commonly assumed that X is the latent state and E is the evidence. In that case we treat X as *missing data*.

Compute data likelihood

- Marginalize over all possible state sequences
- $P(E | \theta) = \sum_X P(E, X | \theta)$
 - Assume domain of each X_i is $x_1, x_2, x_3, x_4 \dots$
- *Problem:* Exponentially many paths (N^T)
- *Solution:* **Forward algorithm**
 - *Dynamic programming* to avoid unnecessary recalculations
 - Create a table of all the possible state sequences, annotated with probabilities

Forward Algorithm

- Trellis of possible state sequences

States		E_1	E_2	E_3	E_4	E_5
	x1	$\alpha_1(1)$	$\alpha_1(2)$	$\alpha_1(3)$	$\alpha_1(4)$	$\alpha_1(5)$
	x2	$\alpha_2(1)$	$\alpha_2(2)$	$\alpha_2(3)$	$\alpha_2(4)$	$\alpha_2(5)$
	x3	$\alpha_3(1)$	$\alpha_3(2)$	$\alpha_3(3)$	$\alpha_3(4)$	$\alpha_3(5)$
	x4	$\alpha_4(1)$	$\alpha_4(2)$	$\alpha_4(3)$	$\alpha_4(4)$	$\alpha_4(5)$
	x5	$\alpha_5(1)$	$\alpha_5(2)$	$\alpha_5(3)$	$\alpha_5(4)$	$\alpha_5(5)$
		Time				

- $\alpha_i(t)$ is $P(\mathbf{E}_{1:t}, X_t = i | \theta)$
- Probability of current state and observations up to now

First Column

- $\alpha_i(t)$ is $P(E_{1:t}, X_t = i | \theta)$

	E_1	E_2	E_3	E_4	E_5
x1	$\alpha_1(1)$	$\alpha_1(2)$	$\alpha_1(3)$	$\alpha_1(4)$	$\alpha_1(5)$
x2	$\alpha_2(1)$	$\alpha_2(2)$	$\alpha_2(3)$	$\alpha_2(4)$	$\alpha_2(5)$
x3	$\alpha_3(1)$	$\alpha_3(2)$	$\alpha_3(3)$	$\alpha_3(4)$	$\alpha_3(5)$
x4	$\alpha_4(1)$	$\alpha_4(2)$	$\alpha_4(3)$	$\alpha_4(4)$	$\alpha_4(5)$
x5	$\alpha_5(1)$	$\alpha_5(2)$	$\alpha_5(3)$	$\alpha_5(4)$	$\alpha_5(5)$

States

Time

$\alpha_j(1) = \pi_j b_j(E_1)$

- Consider:
 - Initial state probability
 - First emission

Middle Cells

- $\alpha_i(t)$ is $P(\mathbf{E}_{1:t}, X_t = i | \theta)$

	E_1	E_2	E_3	E_4	E_5
x1	$\alpha_1(1)$	$\alpha_1(2)$	$\alpha_1(3)$	$\alpha_1(4)$	$\alpha_1(5)$
x2	$\alpha_2(1)$	$\alpha_2(2)$	$\alpha_2(3)$	$\alpha_2(4)$	$\alpha_2(5)$
x3	$\alpha_3(1)$	$\alpha_3(2)$	$\alpha_3(3)$	$\alpha_3(4)$	$\alpha_3(5)$
x4	$\alpha_4(1)$	$\alpha_4(2)$	$\alpha_4(3)$	$\alpha_4(4)$	$\alpha_4(5)$
x5	$\alpha_5(1)$	$\alpha_5(2)$	$\alpha_5(3)$	$\alpha_5(4)$	$\alpha_5(5)$

States

Time

$$\alpha_j(t) = \sum_{i=1}^N \alpha_i(t-1) a_{ij} b_j(E_t)$$

- Consider:
 - All possible ways to get to current state
 - Emission from current state

After Last Column

- $\alpha_i(t)$ is $P(\mathbf{E}_{1:t}, X_t = i | \theta)$

States

	E_1	E_2	E_3	E_4	E_5
x1	$\alpha_1(1)$	$\alpha_1(2)$	$\alpha_1(3)$	$\alpha_1(4)$	$\alpha_1(5)$
x2	$\alpha_2(1)$	$\alpha_2(2)$	$\alpha_2(3)$	$\alpha_2(4)$	$\alpha_2(5)$
x3	$\alpha_3(1)$	$\alpha_3(2)$	$\alpha_3(3)$	$\alpha_3(4)$	$\alpha_3(5)$
x4	$\alpha_4(1)$	$\alpha_4(2)$	$\alpha_4(3)$	$\alpha_4(4)$	$\alpha_4(5)$
x5	$\alpha_5(1)$	$\alpha_5(2)$	$\alpha_5(3)$	$\alpha_5(4)$	$\alpha_5(5)$

$$P(\mathbf{E}|\theta) = \sum_{j=1}^N \alpha_j(T)$$

- Sum over last column for overall likelihood

Forward Algorithm Summary

- Create trellis $\alpha_i(t)$ for $i = 1 \dots N, t = 1 \dots T$
- $\alpha_j(1) = \pi_j b_j(E_1)$ for $j = 1 \dots N$
- for $t = 2 \dots T$:
 - for $j = 1 \dots N$:
 - $\alpha_j(t) = \sum_{i=1}^N \alpha_i(t-1) a_{ij} b_j(E_t)$
- $P(\mathbf{E}|\theta) = \sum_{j=1}^N \alpha_j(T)$
- Runtime: $O(N^2 T)$

Filtering

- **Goal:** $P(X_t = i | \mathbf{E}_{1:t})$
- Rewrite:
$$P(X_t = i | \mathbf{E}_{1:t}, \theta) = \frac{P(X_t=i, \mathbf{E}_{1:t} | \theta)}{P(\mathbf{E}_{1:t} | \theta)}$$
 - Numerator is exactly cell $\alpha_i(t)$
 - Denominator is what we get from the Forward algorithm

Prediction

- **Goal:** $P(X_{t+k} | \mathbf{E}_{1:t})$, for $k > 0$
- First, do filtering from time steps 1 to t , so that we have $P(X_t | \mathbf{E}_{1:t})$
- Then, use the following:
for $m = 0 : k-1$, compute
$$P(X_{t+m+1} | \mathbf{E}_{1:t}) = \sum_j P(X_{t+m+1} | X_{t+m} = j) P(X_{t+m} = j | \mathbf{E}_{1:t})$$
Save the computations $P(X_{t+m+1} | \mathbf{E}_{1:t})$ for the next iteration.

Backward Algorithm

- Nothing stops us from going backwards too!
 - This is not just for fun, as we'll see later.
- Define new trellis with cells $\beta_i(t)$
 - $\beta_i(t) = P(\mathbf{E}_{t+1:T} | X_t = i, \theta)$
 - Probability of all subsequent emissions, given current state is i .
 - Note that unlike $\alpha_i(t)$, it *excludes* the current emission

Backward Algorithm Summary

- Create trellis $\beta_i(t)$ for $i = 1 \dots N, t = 1 \dots T$
- $\beta_i(T) = 1$ for $i = 1 \dots N$
- for $t = T-1 \dots 1$:
 - for $i = 1 \dots N$:
 - $\beta_i(t) = \sum_{j=1}^N a_{ij} b_j(E_{t+1}) \beta_j(t+1)$
- $P(E|\theta) = \sum_{i=1}^N \pi_i b_i(E_1) \beta_i(1)$
- Runtime: $O(N^2 T)$

Remember $\beta_j(t+1)$ does not include $b_j(O_{t+1})$,
so we need to add this factor in!



Smoothing

- **Goal:** $P(X_k | \mathbf{E}_{1:t})$, for $0 \leq k < t$

- Notice that:

$$\alpha_i(k) = P(\mathbf{E}_{1:k}, X_k = i | \theta)$$

$$\beta_i(k) = P(\mathbf{E}_{k+1:t} | X_k = i, \theta)$$

- That means

- $\alpha_i(k)\beta_i(k) = P(\mathbf{E}_{1:t}, X_k = i | \theta)$
- Probability of the *entire* sequence of observations, and we are in state i at timestep k .
- From this, we can compute $P(X_k | \mathbf{E}_{1:t})$ using the definition of conditional probability, and $P(\mathbf{E}_{1:t})$ from before.

Working in the Log Domain

- Practical note: need to avoid underflow—work in log domain

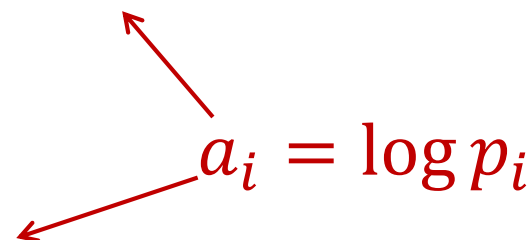
$$\log(\prod p_i) = \sum \log p_i = \sum a_i$$

- Log sum trick

$$\log(\sum p_i) = \log(\sum \exp a_i)$$

$$\text{Let } b = \max a_i$$

$$\begin{aligned}\log(\sum \exp a_i) &= \log \exp(b) \sum \exp(a_i - b) \\ &= b + \log \sum \exp(a_i - b)\end{aligned}$$


$$a_i = \log p_i$$

Sequence Labelling

- Find most likely explanation for a sample
 - $\mathbf{X}^* = \operatorname{argmax}_{\mathbf{X}} P(\mathbf{X}, \mathbf{E} | \theta) = \operatorname{argmax}_{\mathbf{X}} P(\mathbf{X} | \mathbf{E}, \theta)$
- *Intuition*: use forward algorithm, but replace summation with max
 - This is now called the **Viterbi algorithm**.
 - Trellis cells:
 - $\delta_i(t) = \max_{X_{1:t-1}} P(X_{1:t-1}, E_{1:t}, X_t = i | \theta)$

Viterbi Algorithm Summary

- Create trellis $\delta_i(t)$ for $i = 1 \dots N, t = 1 \dots T$
- $\delta_j(1) = \pi_j b_j(E_1)$ for $j = 1 \dots N$
- for $t = 2 \dots T$:
 - for $j = 1 \dots N$:
 - $\delta_j(t) = \max_i \delta_i(t-1) a_{ij} b_j(E_t)$
- Take $\max_i \delta_i(T)$
- Runtime: $O(N^2 T)$

Backpointers

	E_1	E_2	E_3	E_4	E_5	
States	x1	$\delta_1(1)$	$\delta_1(2)$	$\delta_1(3)$	$\delta_1(4)$	$\delta_1(5)$
	x2	$\delta_2(1)$	$\delta_2(2)$	$\delta_2(3)$	$\delta_2(4)$	$\delta_2(5)$
	x3	$\delta_3(1)$	$\delta_3(2)$	$\delta_3(3)$	$\delta_3(4)$	$\delta_3(5)$
	x4	$\delta_4(1)$	$\delta_4(2)$	$\delta_4(3)$	$\delta_4(4)$	$\delta_4(5)$
	x5	$\delta_5(1)$	$\delta_5(2)$	$\delta_5(3)$	$\delta_5(4)$	$\delta_5(5)$
	Time					

- Keep track of where the max entry to each cell came from
- Work backwards to recover best label sequence

Exercise

- 3 states (X, Y, Z), 2 possible emissions (!,@)
 - $\pi = \langle 0.2 \quad 0.5 \quad 0.3 \rangle$
 - $A = \begin{bmatrix} 0.5 & 0.4 & 0.1 \\ 0.2 & 0.3 & 0.5 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$
 - $B = \begin{bmatrix} 0.1 & 0.9 \\ 0.5 & 0.5 \\ 0.7 & 0.3 \end{bmatrix}$
 - Run the forward, backward, and Viterbi algorithms on the emission sequence "!"

Unsupervised learning

- *Problem*: No state sequences to compute above
- *Solution*: Guess the state sequences
- Initialize parameters randomly
- Repeat for a while:
 1. Predict the current state sequences using the current model
 2. Update the current parameters based on current predictions

"Hard EM" or Viterbi EM

- Initialize parameters randomly
- Repeat for a while:
 1. Predict the current state sequences using the current model with the Viterbi algorithm
 2. Update the current parameters using the current predictions as in the supervised learning case
- Can also use "soft" predictions; i.e., the probabilities of all the possible state sequences

Baum-Welch Algorithm

- a.k.a., **Forward-backward** algorithm
- EM algorithm applied to HMMs:
 - **Expectation** Get *expected* counts for hidden structures using current θ^k .
 - **Maximization** Find θ^{k+1} to maximize the likelihood of the training data given the expected counts from the E-step.

Responsibilities Needed

- Supervised/Hard EM: A single tag
- Baum-Welch: *Distribution* over tags
- Call such probabilities **responsibilities**.
 - $\gamma_i(t) = P(X_t = i | E, \theta^k)$
 - Probability of being in state i at time t given the observed sequence under the current model.
 - $\xi_{ij}(t) = P(X_t = i, X_{t+1} = j | E, \theta^k)$
 - Probability of transitioning from i at time t to j at time $t + 1$.

E-Step γ

$$\begin{aligned}\gamma_i(t) &= P(X_t = i | \mathbf{E}, \theta^k) \\ &= \frac{P(X_t = i, \mathbf{E} | \theta^k)}{P(\mathbf{E} | \theta^k)} \\ &= \frac{\alpha_i(t) \beta_i(t)}{P(\mathbf{E} | \theta^k)}\end{aligned}$$

E-Step ξ

$$\begin{aligned}\xi_{ij}(t) &= P(X_t = i, X_{t+1} = j | \mathbf{E}, \theta^k) \\ &= \frac{P(X_t = i, X_{t+1} = j, \mathbf{E} | \theta^k)}{P(\mathbf{E} | \theta^k)} \\ &= \frac{\alpha_i(t) a_{ij} b_j(E_{t+1}) \beta_j(t+1)}{P(\mathbf{E} | \theta^k)}\end{aligned}$$

Beginning to state i at time t

Transition from i to j

Emit E_{t+1}

Rest of the sequence

M-Step

- "Soft" versions of the MLE updates:

- $\pi_i^{k+1} = \gamma_i(1)$

- $a_{ij}^{k+1} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$

- $b_i^{k+1}(e_k) = \frac{\sum_{t=1}^T \gamma_i(t) |_{E_t=e_k}}{\sum_{t=1}^T \gamma_i(t)}$

- With multiple sentences, sum up expected counts over all sentences

Compare:

$$\frac{\#(i, j)}{\#(i)}$$

$$\frac{\#(\text{emission } k, \text{ state } i)}{\#(i)}$$

When to Stop EM?

- Multiple options
 - When training set likelihood stops improving
 - When prediction performance on held-out **development** or **validation** set stops improving

Proof of Baum-Welch Correctness

- **Part 1:** Show that in our procedure, one iteration corresponds to this update:
 - $\theta^{k+1} = \underset{\theta}{\operatorname{argmax}} \sum_X \log[P(\mathbf{E}, \mathbf{X}|\theta)]P(\mathbf{X}|\mathbf{E}, \theta^k)$
 - <http://www.cs.berkeley.edu/~stephentu/writeups/hmm-baum-welch-derivation.pdf>
- **Part 2:** Show that improving the quantity
 - $\sum_X \log[P(\mathbf{E}, \mathbf{X}|\theta)]P(\mathbf{X}|\mathbf{E}, \theta^k)$
 - corresponds to improving $P(\mathbf{E}|\theta)$
 - https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm#Proof_of_correctness

Summary

- Temporal models replicate probabilistic model over t time slides.
- Four important tasks:
 - Filtering
 - Prediction
 - Smoothing
 - Most likely explanation
- Plus: unsupervised learning with EM
- Each poses a specific probabilistic question. Know the similarities / differences, and how to answer (compute) each.