Hidden Markov Models

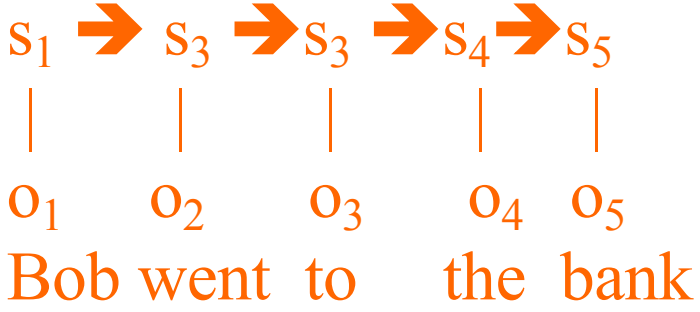
Slides mostly from Mitch Marcus and Eric Fosler
(with lots of modifications).

Have you seen HMMs?
Have you seen dynamic programming?
Have you seen Kalman filters or linear dynamical systems?
Have you see LSTMs?

Penn
University of Pennsylvania

HMMs are dynamic latent variable models

- Given a sequence of *sounds*, find the sequence of *words* most likely to have produced them
- Given a sequence of *images* find the sequence of *locations* most likely to have produced them.
- Given a sequence of *words*, find the sequence of "*meanings*" most likely to have generated them
  - Or parts of speech
    - —Noun, verb, adverb, …
  - Or entity type
    - —Person, place, company, date, movie
      - E.g. *river bank* vs. *money bank*



Penn
University of Pennsylvania

Conditional Independence

- If we want the joint probability of an entire sequence, the *Markov* assumption lets us treat it as a product of "bigram" conditional probabilities:

$$p(w1,w2,w3,w4) =$$

$$p(w1)\ p(w2|w1)\ p(w3|w2,w1)\ p(w4|w3,w2,w1) \sim$$

$$p(w1)\ p(w2|w1)\ p(w3|w2) \qquad p(w4|w3)$$

A Markovian weather model:

- **Tomorrow is like today, except when it isn't.**

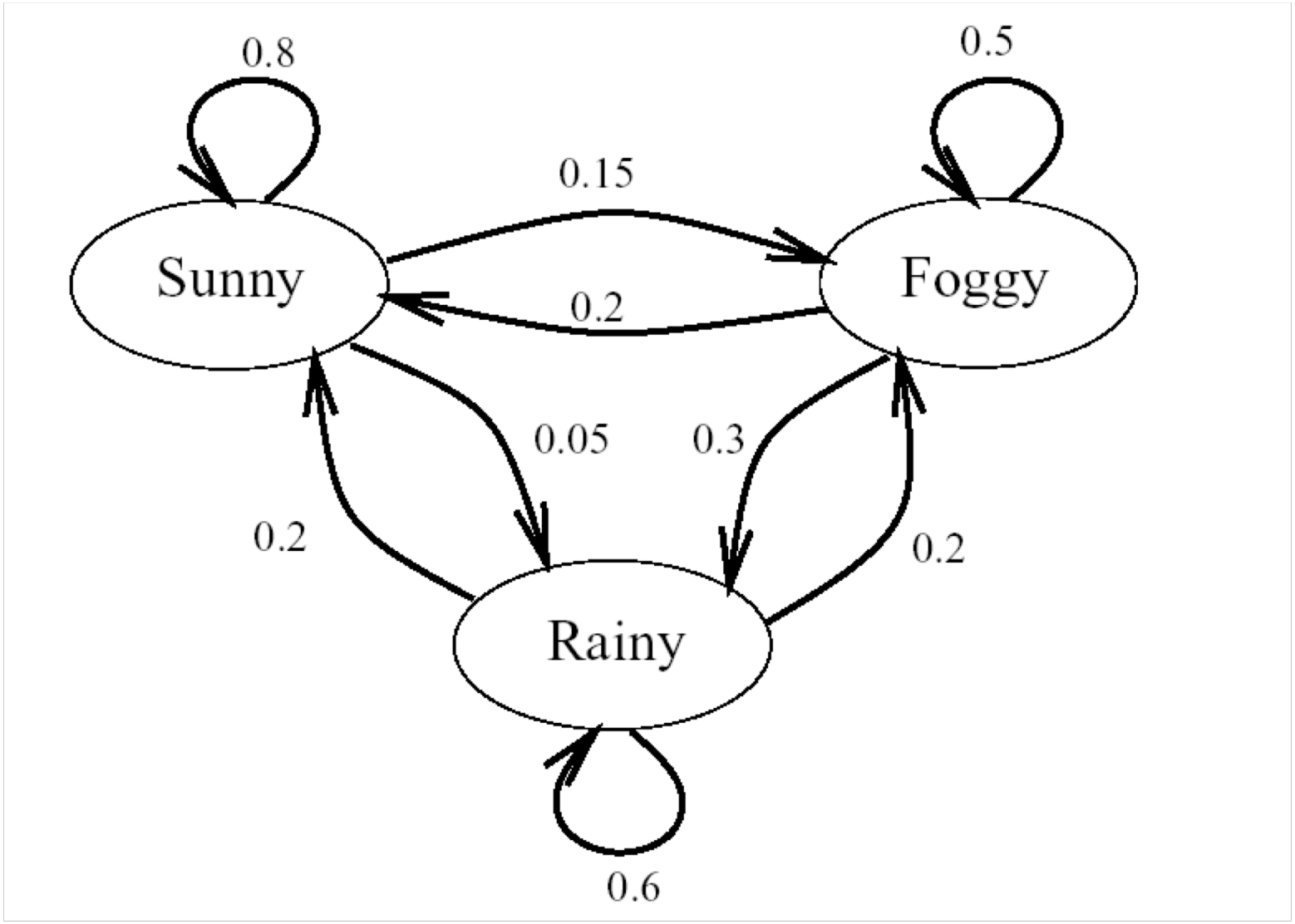
		Tomorrow's Weather		
		Sunny	Rainy	Foggy
Today's Weather	Sunny	0.8	0.05	0.15
	Rainy	0.2	0.6	0.2
	Foggy	0.2	0.3	0.5

- **Markov matrix gives**

  p(tomorrow's weather | today's weather)

If you start from any prior distribution over states (weathers) and run long enough you converge to a stationary distribution.

The same model expressed as a graph



Imperfect Knowledge

But sometimes we can only observe a process "as through a glass darkly." We don't get to see what is really happening, but only some clues that are more or less strongly indicative of what might be happening.

So you're bucking for partner in a windowless law office and you don't see the weather for days at a time … But you do see whether your office mate (who has an actual life) brings in an umbrella or not:

	Probability of Umbrella
Sunny	0.1
Rainy	0.8
Foggy	0.3

How to make predictions?

Now you're bored with researching briefs,
and you want to guess the weather
from a sequence of umbrella (non)sightings:

$P(w_1, w_2, \ldots w_n \mid u_1, \ldots, u_n)$

You observe u, but not w.

w is the "hidden" part of the "Hidden Markov Model"

How to do it?

In speech recognition, we will observe the sounds, but not the intended words

Bayes rule rules!

Bayes' Rule!

$$P(w_1, \ldots, w_n \mid u_1, \ldots, u_n) = \frac{P(u_1, \ldots, u_n \mid w_1, \ldots, w_n) P(w_1, \ldots, w_n)}{P(u_1, \ldots, u_n)}$$

A Rainy-Day Example

- **You go into the office Sunday morning and it's sunny.**
  - w_1 = Sunny
- **You work through the night on Sunday, and on Monday morning, your officemate comes in with an umbrella.**
  - u_2 = T
- **What's the probability that Monday is rainy?**
  - $P(w_2=\text{Rainy} \mid w_1=\text{Sunny}, u_2=T) =$

    $P(u_2=T|w_2=\text{Rainy})/P(u_2=T| w_1=\text{Sunny}) \times P(w_2=\text{Rainy}| w_1=\text{Sunny})$

    (likelihood of umbrella)/normalization x prior

Bayes rule for speech

- **To find the most likely word**
  - Start with a prior of how likely each word is
  - And the likelihood of each set of sounds given the word
- **The most likely word is the one most likely to have generated the sounds heard**

The "fundamental equation of speech recognition":

$$argmax_w\ P(w|u) = argmax_w\ P(u|w)\ P(w) / P(u)$$

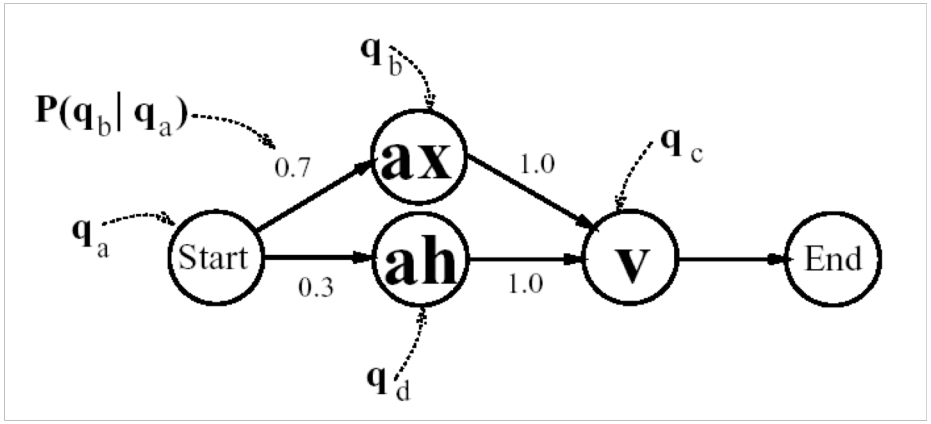
w = word, u = sound ("utterance")

Speech Recognition

- **Markov model for words in a sentence**

  P(I like cabbages) = P(I|START)P(like|I)P(cabbages|like)

- **Markov model for sounds in a word**
  - Model the relation of words to sounds by breaking words down into pieces



HMMs can also extract meaning

- **Natural language is ambiguous**
  - —"Banks banks in banks on banks."
- **Sequence of hidden states are the "meanings" (what the word refers to) and words are the percepts**
- **The (Markov) Language Model says how likely each meaning is to follow other meanings**
  - All meanings of "banks" may produce the same percept

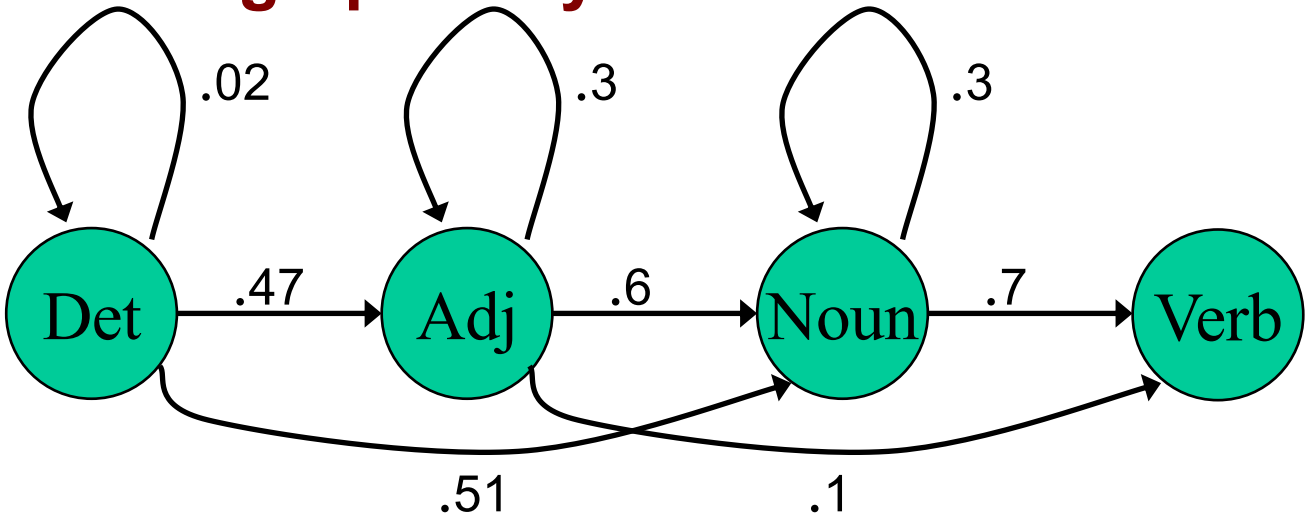
HMMs: Midpoint Summary

- **Language can be modeled by HMMs**
  - Predict words from sounds
  - Captures priors on words
  - Hierarchical
    - Phonemes to morphemes to words to phrases to sentences
  - Was used in all commercial speech recognition software
    - Now replaced by deep networks
- **Markov assumption, HMM definition**
  - “Fundamental equation of speech recognition”

Hidden Markov Models (HMMs)

Part of Speech tagging is often done using HMMs

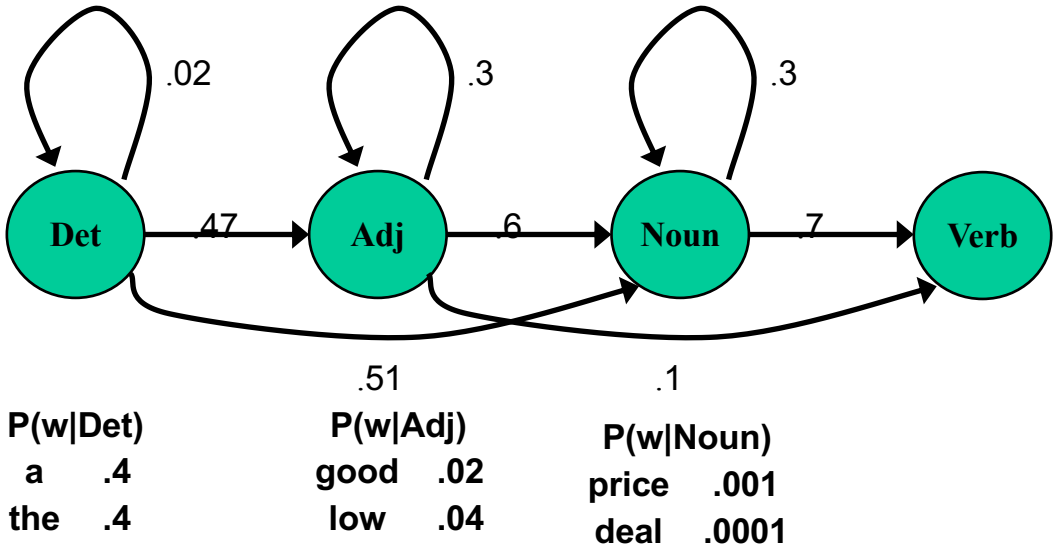
Viewed graphically:



P(w|Det)		P(w|Adj)		P(w|Noun)	
a	.4	good	.02	price	.001
the	.4	low	.04	deal	.0001

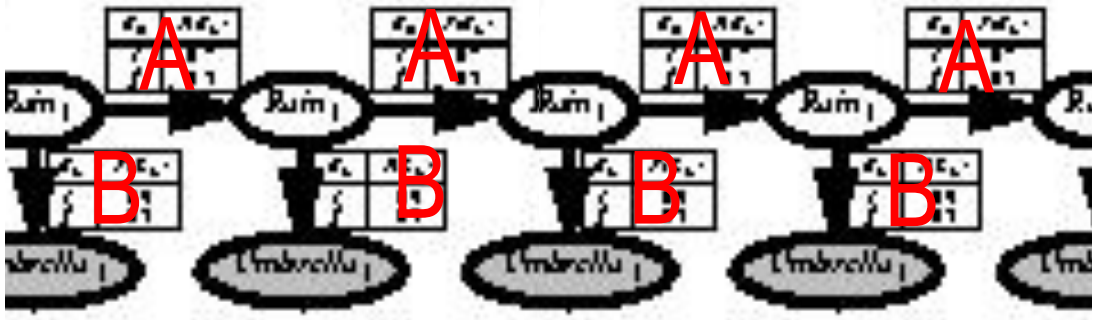
HMM: The Model

- **Starts in some initial state s_i**
- **Moves to a new state s_j with probability $p(s_j | s_i) = a_{ij}$**
  - The matrix *A* with elements a_{ij} is called the *Markov transition matrix*
- **Emits an observation o_v (e.g. a word, w) with probability $p(o_v | s_i) = b_{iv}$**



A HMM is a dynamic Bayesian Network

There is a node for the hidden state and for the emission (observed state) at each time step, but the probability tables are the same at all times.



A: Markov transition matrix

B: Emission probabilities

Recognition using an HMM

To find the tag sequence T which maximizes $P(T|W)$, we note that:

Note that the 'tags' *t* are the hidden states (***s***) and the words *w* are the emissions (***o***)

$$P(T|W) \propto \pi(t_1) * \prod_{i=1}^{n-1} a(t_i, t_{i+1}) * \prod_{i=1}^{n} b(t_i, w_i)$$

Transitions Emissions

So we need to find

$$\hat{t}_{1,n} = \arg\max_{t_{1,n}} \pi(t_1) * \prod_{i=1}^{n-1} a(t_i, t_{i+1}) * \prod_{i=1}^{n} b(t_i, w_i)$$

$$P(T) * P(W|T) = P(t_1) * P(t_2|t_1) * P(t_3|t_2) * \ldots * P(t_n|t_{n-1}) * P(w_1|t_1) * P(w_2|t_2) * \ldots * P(w_n|t_n)$$

Parameters of an HMM

- *States*: A set of states $S = s_1,\ldots,s_k$
- *Markov transition probabilities*: $A = a_{1,1},a_{1,2},\ldots,a_{k,k}$ Each $a_{i,j} = p(s_j \mid s_i)$ represents the probability of transitioning from state s_i to s_j.
- *Emission probabilities*: A set B of functions of the form $b_i(o_t) = p(o|s_i)$ giving the probability of observation o_t being emitted by s_i
- *Initial state distribution*: π_i is the probability that s_i is a start state

The Three Basic HMM Problems

- ***Problem 1 (Evaluation):*** **Given the observation sequence $O=o_1,\ldots,o_T$ and an HMM model $\lambda = (A,B,\pi)$, compute the probability of O given the model.**
- ***Problem 2 (Decoding):*** **Given the observation sequence $O=o_1,\ldots,o_T$ and an HMM model $\lambda = (A,B,\pi)$, find the state sequence that best explains the observations**

(This and following slides follow classic formulation by Rabiner and Juang, as adapted by Manning and Schutze. Slides adapted from Dorr.)

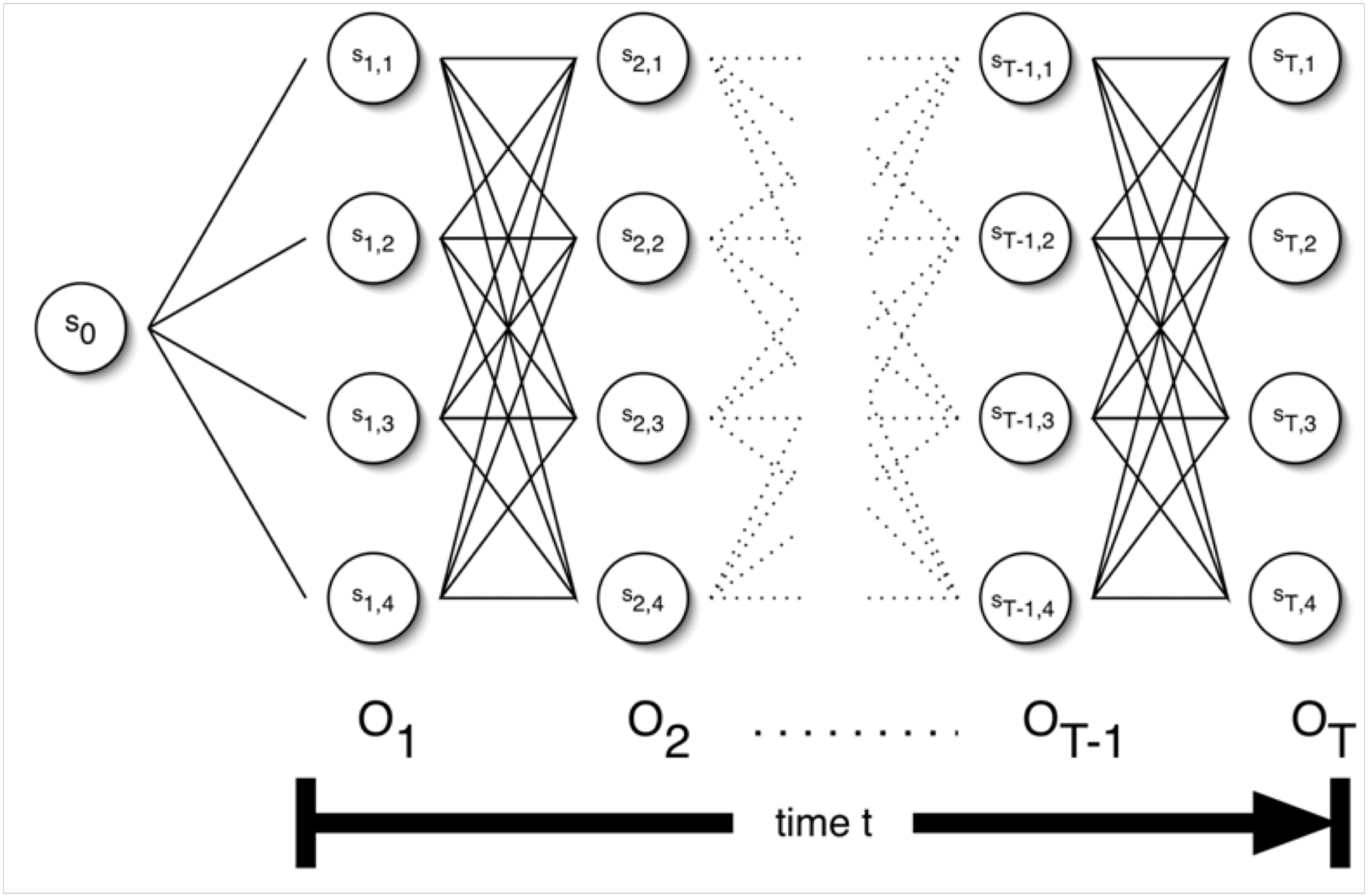
The Three Basic HMM Problems

- ***Problem 3 (Learning):*** **Pick the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O \mid \lambda)$**

Problem 1: Probability of an Observation Sequence

- **What is** $P(O \mid \lambda)$**?**
- The probability of a observation sequence is the sum of the probabilities of all possible state sequences in the HMM.
- Naïve computation is very expensive. *Given T observations and k states,* there are k^T possible state sequences.
- Even small HMMs, e.g. T=10 and k=10, contain 10 billion different paths
- The solution: use *dynamic programming*
  - *Once you are in a state it doesn't matter how you got there*

The Trellis (for a 4 state model)

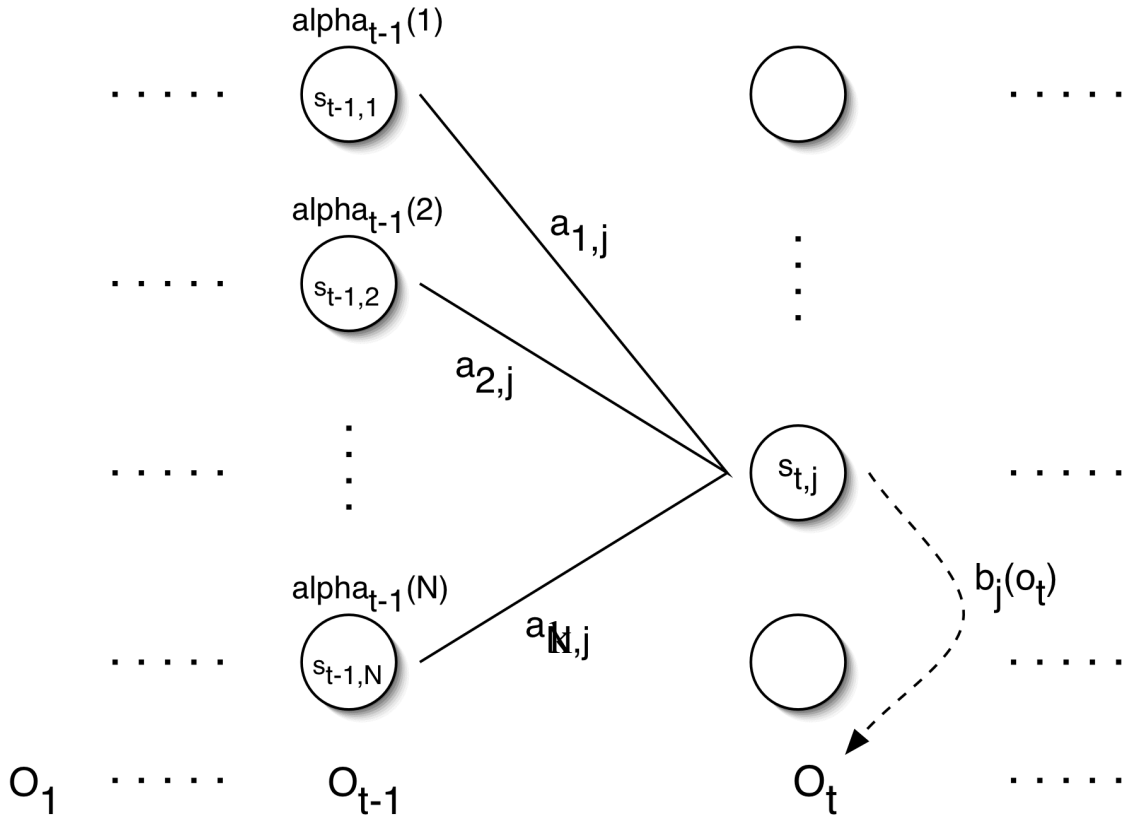


Forward Probabilities

- **What is the probability that, given an HMM λ, at time t the state is s_i and the partial observation $o_1 \ldots o_t$ has been generated?**

$$\alpha_t(i) = P(o_1 \ldots o_t, q_t = s_i \mid \lambda)$$

Forward Probabilities



N should be k; sorry!

$$\alpha_t(j) = \left[\sum_{i=1}^{k} \alpha_{t-1}(i) a_{ij} \right] b_j(o_t)$$

Note: summation is over the k hidden states

Forward Algorithm

- **Initialization:** $\alpha_1(i) = \pi_i b_i(o_1) \quad 1 \le i \le k$

- **Induction:**

$$\alpha_t(j) = \left[\sum_{i=1}^{k} \alpha_{t-1}(i) a_{ij} \right] b_j(o_t) \quad 2 \le t \le T, 1 \le j \le k$$

- **Termination:**

$$P(O \mid \lambda) = \sum_{i=1}^{k} \alpha_T(i)$$

Forward Algorithm Complexity

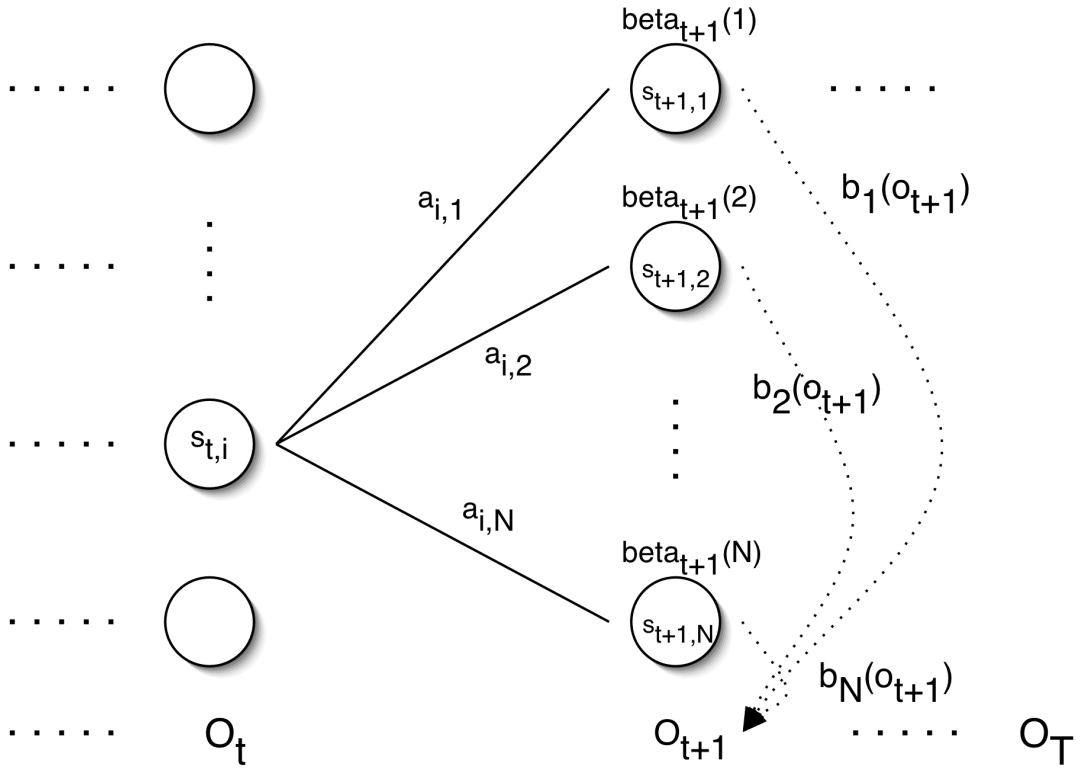
- Naïve approach takes $O(2T*k^T)$ computation
- Forward algorithm using dynamic programming takes $O(k^2T)$ computations

Backward Probabilities

- **What is the probability that**
  - given an HMM $\lambda = (A, B, \pi)$ and
  - given the state at time t is s_i,
  - the partial observation $o_{t+1} \ldots o_T$ is generated?
- **Analogous to forward probability, just in the other direction**

$$\beta_t(i) = P(o_{t+1} \ldots o_T \mid q_t = s_i, \lambda)$$

Backward Probabilities



N should be k; sorry!

$$\beta_t(i) = \left[\sum_{j=1}^{k} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \right]$$

Backward Algorithm

- **Initialization:** $\beta_T(i) = 1, \quad 1 \le i \le k$

- **Induction:**

$$\beta_t(i) = \left[\sum_{j=1}^{k} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \right] \quad t = T-1 \ldots 1, 1 \le i \le k$$

- **Termination:**

$$P(O \mid \lambda) = \sum_{i=1}^{k} \pi_i \beta_1(i)$$

Problem 2: Decoding

- The forward algorithm gives the *sum of all paths* through an HMM efficiently.
- Here, we want to find the *highest probability path*.
- We want to find the state sequence $Q=q_1 \ldots q_T$, such that

$$Q = \arg\max_{Q'} P(Q' \mid O, \lambda)$$

Viterbi Algorithm

- Similar to computing the forward probabilities, but instead of summing over transitions from incoming states, compute the maximum

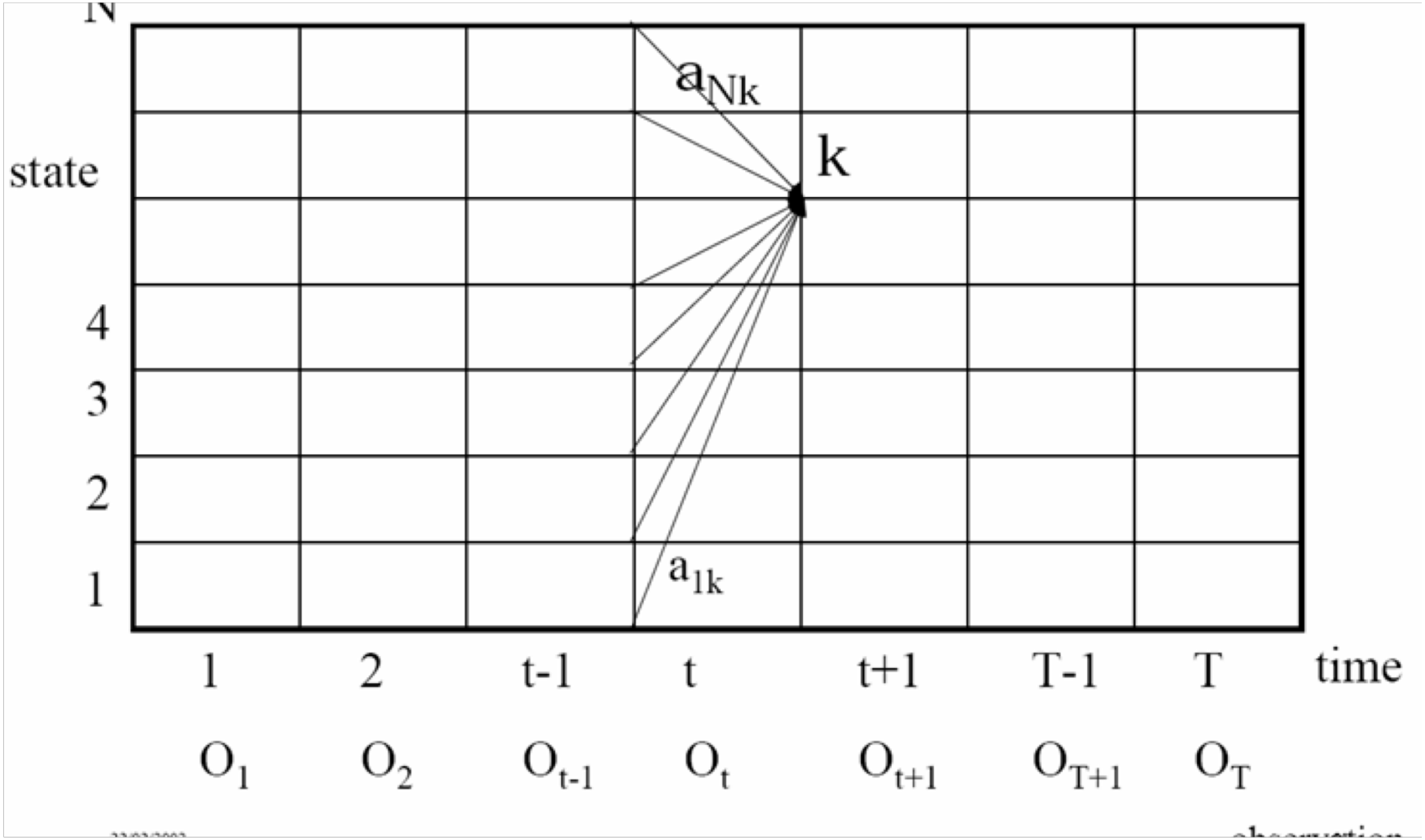
- **Forward:**

$$\alpha_t(j) = \left[\sum_{i=1}^{k} \alpha_{t-1}(i) a_{ij} \right] b_j(o_t)$$

- **Viterbi Recursion:**

$$\delta_t(j) = \left[\max_{1 \le i \le k} \delta_{t-1}(i) a_{ij} \right] b_j(o_t)$$

Core Idea of Viterbi Algorithm



Viterbi Algorithm

- **Initialization:** $\delta_1(i) = \pi_i b_i(o_1) \quad 1 \le i \le k$
- **Induction:**

$$\delta_t(j) = \left[\max_{1 \le i \le k} \delta_{t-1}(i) a_{ij} \right] b_j(o_t)$$

$$\psi_t(j) = \left[\arg\max_{1 \le i \le k} \delta_{t-1}(i) a_{ij} \right] \quad 2 \le t \le T, 1 \le j \le k$$

- **Termination:** $p^* = \max_{1 \le i \le k} \delta_T(i) \qquad q_T^* = \arg\max_{1 \le i \le k} \delta_T(i)$

- **Read out path:** $q_t^* = \psi_{t+1}(q_{t+1}^*) \quad t = T-1, \ldots, 1$

Problem 3: Learning

- **Up to now we've assumed that we *know* the underlying model** $\lambda = (A, B, \pi)$

- **Often these parameters are estimated on annotated training data (i.e. with known 'hidden state')**
  - But of course, such labels are often lacking

- **We want to maximize the parameters with respect to the current data, i.e., find a model** λ' **, such that**

$$\lambda' = \arg\max_{\lambda} P(O \mid \lambda)$$

Problem 3: Learning

- **Unfortunately, there is no known way to analytically find a *global* maximum, i.e., a model λ', such that** $\lambda' = \arg\max_{\lambda} P(O \mid \lambda)$
- **But it is possible to find a *local* maximum**
- **Given an initial model λ, we can always find a model λ', such that** $P(O \mid \lambda') \geq P(O \mid \lambda)$

Forward-Backward (Baum-Welch) algorithm

- **EM algorithm**
  - Find the forward and backward probabilities for the current parameters
  - Re-estimate the model parameters given the estimated probabilities
  - Repeat

Parameter Re-estimation

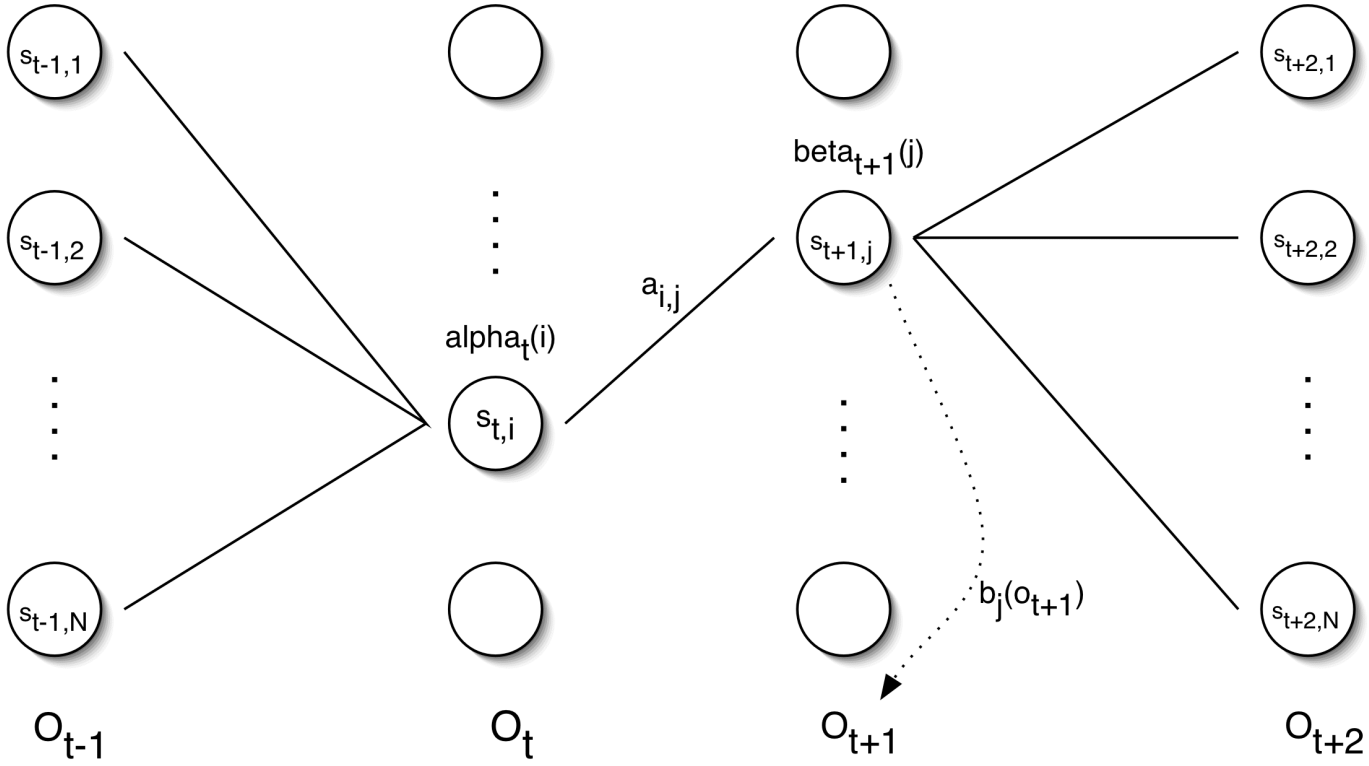
- **Three parameters need to be re-estimated:**
  - Initial state distribution: π_i
  - Transition probabilities: $a_{i,j}$
  - Emission probabilities: $b_i(o_t)$

Re-estimating Transition Probabilities

- **What's the probability of being in state s_i at time t and going to state s_j, given the current model and parameters?**

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j \mid O, \lambda)$$

Re-estimating *Transition* Probabilities



$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{i,j}\, b_j(o_{t+1})\, \beta_{t+1}(j)}{\sum_{i=1}^{k}\sum_{j=1}^{k} \alpha_t(i)\, a_{i,j}\, b_j(o_{t+1})\, \beta_{t+1}(j)}$$

Re-estimating *Transition* Probabilities

- **The intuition behind the re-estimation equation for transition probabilities is**

$$\hat{a}_{i,j} = \frac{expected\ number\ of\ transitions\ from\ state\ s_i\ to\ state\ s_j}{expected\ number\ of\ transitions\ from\ state\ s_i}$$

- **Formally:**

$$\hat{a}_{i,j} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{j'=1}^{k} \xi_t(i,j')}$$

Re-estimating *Transition* Probabilities

- **Defining** $\gamma_t(i) = \sum_{j=1}^{k} \xi_t(i,j)$

  As the probability of being in state s_i, given the complete observation O

- **We can say:**

$$\hat{a}_{i,j} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

Re-estimating *Initial State* Probabilities

- **Initial state distribution: π_i is the probability that s_i is a start state**

- **Re-estimation is easy:**

$$\hat{\pi}_i = \textit{expected number of times in state } s_i \textit{ at time } 1$$

- **Formally:**

$$\hat{\pi}_i = \gamma_1(i)$$

Re-estimation of *Emission* Probabilities

- **Emission probabilities are re-estimated as**

$$\hat{b}_i(k) = \frac{expected\ number\ of\ times\ in\ state\ s_i\ and\ observe\ symbol\ v_k}{expected\ number\ of\ times\ in\ state\ s_i}$$

- **Formally:**

$$\hat{b}_i(k) = \frac{\sum_{t=1}^{T} \delta(o_t, v_k)\,\gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)}$$

where $\delta(o_t, v_k) = 1$, if $o_t = v_k$, and 0 otherwise

- **Note that δ here is the Kronecker delta function and is not related to the δ in the discussion of the Viterbi algorithm!!**

The Updated Model

- **Coming from** $\lambda = (A, B, \pi)$ **we get to** $\lambda' = (\hat{A}, \hat{B}, \hat{\pi})$ **by the following update rules:**

$$\hat{a}_{i,j} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \qquad \hat{b}_i(k) = \frac{\sum_{t=1}^{T} \delta(o_t, v_k)\, \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)} \qquad \hat{\pi}_i = \gamma_1(i)$$

HMM vs CRF

- **HMMs are *generative* models**
  - They model the full probability distribution
- **Conditional Random Fields are *discriminative***
  - Similar to HMMs, but only model the probability of labels given the data
  - Generally give more accurate predictions
  - A bit harder to code
    - —But good open source versions are available
  - Very popular in the research world (until deep nets arrived)

What you should know

- **HMMs**
  - Markov assumption
  - Markov transition matrix, Emission probabilities
  - Viterbi decoding (just well enough to do the homework)
- **Many other models generalize HMM**
  - Emission can be a real valued (e.g. Gaussian) function of the hidden state
  - The hidden state can be a real valued vector instead of a "one hot" discrete state
    - —Instead of moving with a Markov Transition Matrix between states, one moves with Gaussian noise between real states
  - Nonlinear versions give dynamical neural nets