

# Lecture 4

## Hidden Markov Models

Michael Picheny, Bhuvana Ramabhadran, Stanley F. Chen,  
Markus Nussbaum-Thom

Watson Group  
IBM T.J. Watson Research Center  
Yorktown Heights, New York, USA  
{picheny,bhuvana,stanchen,nussbaum}@us.ibm.com

10 February 2016

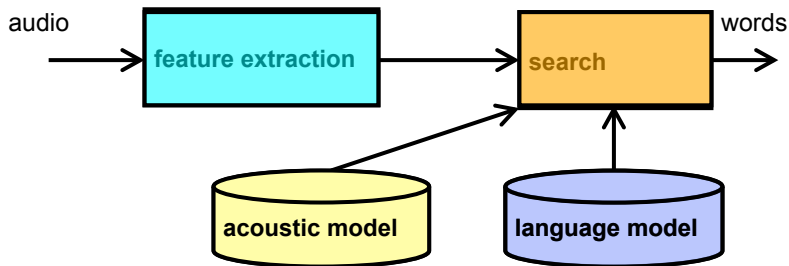
# Administrivia

- Lab 1 is due Friday at 6pm!
  - Use Piazza for questions/discussion.
  - Thanks to everyone for answering questions!
- Late policy:
  - Can be late on one lab up to two days for free.
  - After that, penalized 0.5 for every two days (4d max).
- Lab 2 posted on web site by Friday.

# Administrivia

- Clear (11); mostly clear (6).
- Pace: OK (7), slow (1).
- Muddiest: EM (5); hidden variables (2); GMM formulae/params (2); MLE (1).
- Comments (2+ votes):
  - good jokes/enjoyable (3)
  - lots of good examples (2)
- Quote: "Great class, loved it. Left me *speech*-less."

# Components of a speech recognition system



$$W^* = \underset{W}{\operatorname{arg\,max}} \quad P(\textcolor{cyan}{X} | W, \Theta) \quad P(W | \Theta)$$

Today's Subject →

# Recap: Probabilistic Modeling for ASR

- Old paradigm: DTW.

$$w^* = \arg \min_{w \in \text{vocab}} \text{distance}(A'_{\text{test}}, A'_w)$$

- New paradigm: Probabilities.

$$w^* = \arg \max_{w \in \text{vocab}} P(A'_{\text{test}} | w)$$

- $P(A' | w)$  is (relative) frequency with which  $w \dots$
  - Is realized as feature vector  $A'$ .
- The more “accurate”  $P(A' | w)$  is ...
  - The more accurate classification is.

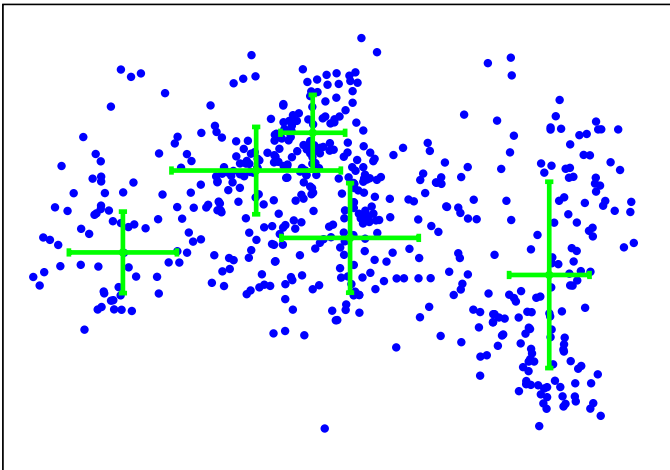
# Recap: Gaussian Mixture Models

- Probability distribution over ...
  - Individual (*e.g.*, 40d) feature vectors.

$$P(\mathbf{x}) = \sum_j p_j \frac{1}{(2\pi)^{d/2} |\Sigma_j|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu_j)^T \Sigma_j^{-1} (\mathbf{x}-\mu_j)}$$

- Can model arbitrary (non-Gaussian) data pretty well.
- Can use EM algorithm to do ML estimation of parameters of the Gaussian distributions
  - Finds local optimum in likelihood, iteratively.

# Example: Modeling Acoustic Data With GMM



# What We Have And What We Want

- What we have:  $P(\mathbf{x})$ .
  - GMM is distribution over indiv feature vectors  $\mathbf{x}$ .
- What we want:  $P(A' = \mathbf{x}_1, \dots, \mathbf{x}_T)$ .
  - Distribution over *sequences* of feature vectors.
  - Build separate model  $P(A'|w)$  for each word  $w$ .
  - There you go.

$$w^* = \arg \max_{w \in \text{vocab}} P(A'_{\text{test}}|w)$$

# Today's Lecture

- Introduce a general probabilistic framework for speech recognition
- Explain how Hidden Markov Models fit in this overall framework
- Review some of the concepts of ML estimation in the context of an HMM framework
- Describe how the three basic HMM operations are computed

# Probabilistic Model for Speech Recognition

$$\begin{aligned}w^* &= \arg \max_{w \in \text{vocab}} P(w|x, \theta) \\&= \arg \max_{w \in \text{vocab}} \frac{P(x|w, \theta)P(w|\theta)}{P(x)} \\&= \arg \max_{w \in \text{vocab}} P(x|w, \theta)P(w|\theta)\end{aligned}$$

- $w^*$  Best sequence of words
- $x$  Sequence of acoustic vectors
- $\theta$  Model Parameters

# Sequence Modeling

- Hidden Markov Models . . .
  - To model sequences of feature vectors (compute probabilities)
  - *i.e.*, how feature vectors evolve over time.
  - Probabilistic counterpart to DTW.
- How things fit together.
  - GMM's: for each sound, what are likely feature vectors?
  - *e.g.*, why the sound “*b*” is different from “*d*”.
  - HMM's: what “sounds” are likely to follow each other?
  - *e.g.*, why *rat* is different from *tar*.

# Acoustic Modeling

- Assume a word is made up from a sequence of speech sounds
  - Cat: K AE T
  - Dog: D AO G
  - Fish: F IH SH
- When a speech sound is uttered, a sequence of feature vectors is produced according to a GMM associated with each sound
- However, the distributions of speech sounds overlap! So you cannot identify which speech sound produced the feature vectors
- If you did, you could just use the techniques we discussed last week
- Solution is the **HMM**

# Simplification: Discrete Sequences

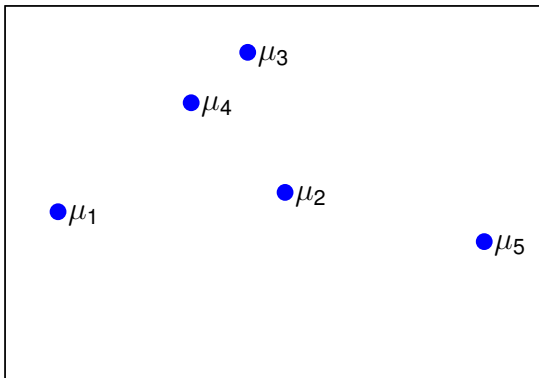
- Goal: continuous data.
  - *e.g.*,  $P(\mathbf{x}_1, \dots, \mathbf{x}_T)$  for  $\mathbf{x} \in \mathcal{R}^{40}$ .
- Most of today: discrete data.
  - $P(x_1, \dots, x_T)$  for  $x \in$  finite alphabet.
- *Discrete* HMM's vs. *continuous* HMM's.

# Vector Quantization

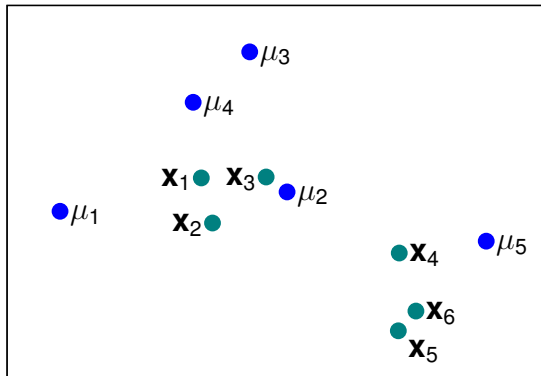
- Before continuous HMM's and GMM's ( $\sim 1990$ ) ...
  - People used discrete HMM's and VQ (1980's).
- Convert multidimensional feature vector ...
  - To discrete symbol  $\{1, \dots, V\}$  using *codebook*.
- Each symbol has representative feature vector  $\mu_j$ .
- Convert each feature vector ...
  - To symbol  $j$  with nearest  $\mu_j$ .

# The Basic Idea

- How to pick the  $\mu_j$ ?



# The Basic Idea



$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6 \dots \Rightarrow 4, 2, 2, 5, 5, 5, \dots$

# Recap

- Need probabilistic sequence modeling for ASR.
- Let's start with discrete sequences.
  - Simpler than continuous.
  - What was used first in ASR.
- Let's go!

# Part I

## Nonhidden Sequence Models

# Case Study: Coin Flipping

- Let's flip (unfair) coin 10 times:  $x_1, \dots, x_{10} \in \{T, H\}$ , e.g.,  
T, T, H, H, H, H, T, H, H, H
- Design  $P(x_1, \dots, x_T)$  matching actual frequencies ...
  - Of sequences  $(x_1, \dots, x_T)$ .
- What should form of distribution be?
- How to estimate its parameters?

# Where Are We?

1 Models Without State

2 Models With State

# Independence

- Coin flips are *independent*!
  - Outcome of previous flips doesn't influence ...
  - Outcome of future flips (given parameters).

$$P(x_1, \dots, x_{10}) = \prod_{i=1}^{10} P(x_i)$$

- System has no *memory* or *state*.
- Example of dependence: draws from deck of cards.
  - e.g., if last card was A♠, next card isn't.
  - State: all cards seen.

# Modeling a Single Coin Flip $P(x_i)$

- *Multinomial* distribution.
- One parameter for each outcome:  $p_H, p_T \geq 0 \dots$ 
  - Modeling frequency of that outcome, *i.e.*,  $P(x) = p_x$ .
- Parameters must sum to 1:  $p_H + p_T = 1$ .
- Where have we seen this before?

# Computing the Likelihood of Data

- Some parameters:  $p_H = 0.6$ ,  $p_T = 0.4$ .
- Some data:

T, T, H, H, H, H, T, H, H, H

- The likelihood:

$$\begin{aligned}P(x_1, \dots, x_{10}) &= \prod_{i=1}^{10} P(x_i) = \prod_{i=1}^{10} p_{x_i} \\&= p_T \times p_T \times p_H \times p_H \times p_H \times \dots \\&= 0.6^7 \times 0.4^3 = 0.00179\end{aligned}$$

- How many such sequences are possible?
- What is the likelihood of

T, H, H, T, H, T, H, H, H, H

# Computing the Likelihood of Data

- More generally:

$$\begin{aligned}P(x_1, \dots, x_N) &= \prod_{i=1}^N p_{x_i} \\&= \prod_x p_x^{c(x)} \\ \log P(x_1, \dots, x_N) &= \sum_x c(x) \log p(x)\end{aligned}$$

where  $c(x)$  is *count* of outcome  $x$ .

- Likelihood only depends on counts of outcomes ...
  - Not on order of outcomes.

# Estimating Parameters

- Choose parameters that maximize likelihood of data ...
  - Because ML estimation is awesome!
- If  $H$  heads and  $T$  tails in  $N = H + T$  flips, log likelihood is:

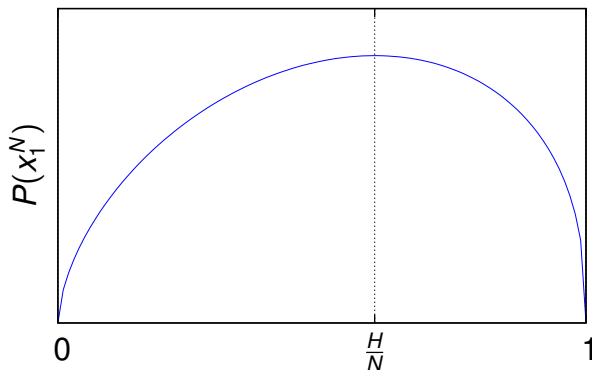
$$L(x_1^N) = \log(p_H)^H (p_T)^T = H \log p_H + T \log(1 - p_H)$$

- Taking derivative w.r.t.  $p_H$  and setting to 0.

$$\begin{aligned} \frac{H}{p_H} - \frac{T}{1 - p_H} &= 0 & p_H &= \frac{H}{H + T} = \frac{H}{N} \\ H - H \times p_H &= T \times p_H & p_T &= 1 - p_H = \frac{T}{N} \end{aligned}$$

# Maximum Likelihood Estimation

- MLE of multinomial parameters is an intuitive estimate!
  - Just relative frequencies:  $p_H = \frac{H}{N}$ ,  $p_T = \frac{T}{N}$ .
  - Count and normalize, baby!
- MLE is the probability that maximizes the likelihood of the sequence



# Example: Maximum Likelihood Estimation

- Training data: 50 samples.

T, T, H, H, H, H, T, H, H, H, T, H, H, T, H, H, T, T, T, T, H,  
T, T, H, H, H, H, H, T, T, H, T, H, T, H, H, T, H, T, H, H, H,  
T, H, H, T, H, H, H, T

- Counts: 30 heads, 20 tails.

$$p_H^{\text{MLE}} = \frac{30}{50} = 0.6 \qquad p_T^{\text{MLE}} = \frac{20}{50} = 0.4$$

- Sample from MLE distribution:

H, H, T, T, H, H, H, T, T, T, H, H, H, H, T, H, T, H, T, H, T,  
T, H, T, H, H, T, H, T, T, H, T, H, T, H, H, T, H, H, H, H, T,  
H, T, H, T, T, H, H, H

# Recap: Multinomials, No State

- Log likelihood just depends on counts.

$$L(x_1^N) = \sum_x c(x) \log p_x$$

- MLE: count and normalize.

$$p_x^{\text{MLE}} = \frac{c(x)}{N}$$

- Easy peasy.

# Where Are We?

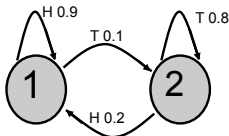
1 Models Without State

2 Models With State

# Case Study: Two coins

- Consider 2 coins:
  - Coin 1:  $p_H = 0.9, p_T = 0.1$
  - Coin 2:  $p_H = 0.2, p_T = 0.8$
- Experiment:  
Flip Coin 1.  
If outcome is H, flip Coin 1 ; else flip Coin 2.
  - H H T T (0.0648)
  - H T H T (0.0018)
- Sequence has memory! Order matters.
- Order matters for speech too (*rat* vs *tar*)

# A Picture: State space representation



- State sequence can be uniquely determined from the observations **given the initial state**
- Output probability is the product of the transition probabilities
  - Example:
  - Obs: H T T T
  - St: **1** 1 2 2
  - P:  $0.9 \times 0.1 \times 0.8 \times 0.8$

# Case Study: Austin Weather

- From National Climate Data Center.
  - R = rainy = precipitation  $> 0.00$  in.
  - W = windy = not rainy; avg. wind  $\geq 10$  mph.
  - C = calm = not rainy and not windy.

- Some data:

W, W, C, C, W, W, C, R, C, R, W, C, C, C, R, R, R, R, C,  
C, R, R, R, R, R, R, R, R, R, C, C, C, C, C, R, R, R, R, R,  
R, R, C, C, C, W, C, C, C, C, C, C, R, C, C, C, C

- Does system have state/memory?
  - Does yesterday's outcome influence today's?

# State and the Markov Property

- How *much* state to remember?
  - How much past information to encode in state?
- Independent events/no memory: remember nothing.

$$P(x_1, \dots, x_N) \stackrel{?}{=} \prod_{i=1}^N P(x_i)$$

- General case: remember everything (always holds).

$$P(x_1, \dots, x_N) = \prod_{i=1}^N P(x_i | x_1, \dots, x_{i-1})$$

- Something in between?

# The Markov Property, Order $n$

- Holds if:

$$\begin{aligned} P(x_1, \dots, x_N) &= \prod_{i=1}^N P(x_i | x_1, \dots, x_{i-1}) \\ &= \prod_{i=1}^N P(x_i | x_{i-n}, x_{i-n+1}, \dots, x_{i-1}) \end{aligned}$$

- e.g., if know weather for past  $n$  days ...
  - Knowing more doesn't help predict future weather.
- i.e., if data satisfies this property ...
  - No loss from just remembering past  $n$  items!

# A Non-Hidden Markov Model, Order 1

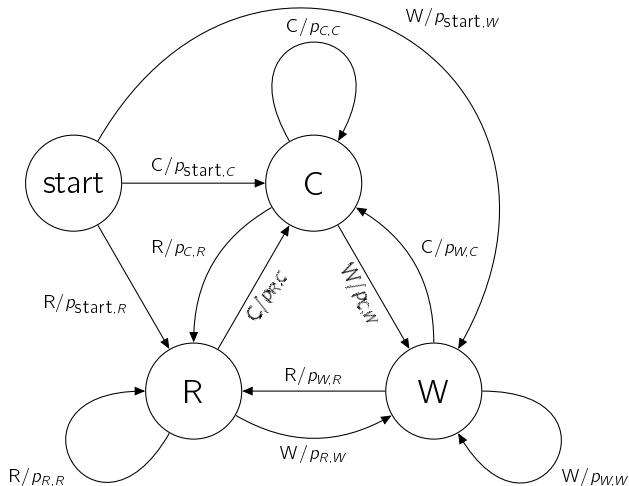
- Let's assume: knowing yesterday's weather is enough.

$$P(x_1, \dots, x_N) = \prod_{i=1}^N P(x_i | x_{i-1})$$

- Before (no state): single multinomial  $P(x_i)$ .
- After (with state): separate multinomial  $P(x_i | x_{i-1}) \dots$ 
  - For each  $x_{i-1} \in \{\text{rainy, windy, calm}\}$ .
  - Model  $P(x_i | x_{i-1})$  with parameter  $p_{x_{i-1}, x_i}$ .
- What about  $P(x_1 | x_0)$ ?
  - Assume  $x_0 = \text{start}$ , a special value.
  - One more multinomial:  $P(x_i | \text{start})$ .
- Constraint:  $\sum_{x_i} p_{x_{i-1}, x_i} = 1$  for all  $x_{i-1}$ .

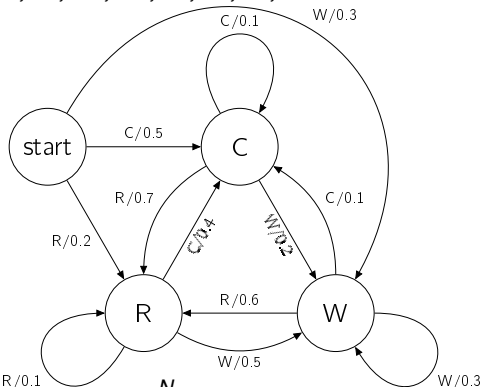
# A Picture

- After observe  $x$ , go to state labeled  $x$ .
- Is state non-hidden?



# Computing the Likelihood of Data

- Some data:  $\mathbf{x} = \text{W, W, C, C, W, W, C, R, C, R}$ .



- The likelihood:

$$\begin{aligned} P(x_1, \dots, x_{10}) &= \prod_{i=1}^N P(x_i | x_{i-1}) = \prod_{i=1}^N p_{x_{i-1}, x_i} \\ &= p_{\text{start}, \text{W}} \times p_{\text{W}, \text{W}} \times p_{\text{W}, \text{C}} \times \dots \\ &= 0.3 \times 0.3 \times 0.1 \times 0.1 \times \dots = 1.06 \times 10^{-6} \end{aligned}$$

# Computing the Likelihood of Data

- More generally:

$$\begin{aligned}P(x_1, \dots, x_N) &= \prod_{i=1}^N P(x_i | x_{i-1}) = \prod_{i=1}^N p_{x_{i-1}, x_i} \\&= \prod_{x_{i-1}, x_i} p_{x_{i-1}, x_i}^{c(x_{i-1}, x_i)}\end{aligned}$$

$$\log P(x_1, \dots, x_N) = \sum_{x_{i-1}, x_i} c(x_{i-1}, x_i) \log p_{x_{i-1}, x_i}$$

- $x_0 = \textit{start}$ .
- $c(x_{i-1}, x_i)$  is count of  $x_i$  following  $x_{i-1}$ .
- Likelihood only depends on counts of pairs (bigrams).

# Maximum Likelihood Estimation

- Choose  $p_{x_{i-1}, x_i}$  to optimize log likelihood:

$$\begin{aligned} L(x_1^N) &= \sum_{x_{i-1}, x_i} c(x_{i-1}, x_i) \log p_{x_{i-1}, x_i} \\ &= \sum_{x_i} c(\text{start}, x_i) \log p_{\text{start}, x_i} + \sum_{x_i} c(R, x_i) \log p_{R, x_i} + \\ &\quad \sum_{x_i} c(W, x_i) \log p_{W, x_i} + \sum_{x_i} c(C, x_i) \log p_{C, x_i} \end{aligned}$$

- Each sum is log likelihood of multinomial.
  - Each multinomial has nonoverlapping parameter set.
- Can optimize each sum independently!

$$p_{x_{i-1}, x_i}^{\text{MLE}} = \frac{c(x_{i-1}, x_i)}{\sum_x c(x_{i-1}, x)}$$

# Example: Maximum Likelihood Estimation

- Some raw data:

W, W, C, C, W, W, C, R, C, R, W, C, C, C, R, R, R, R, C,  
C, R, R, R, R, R, R, R, R, C, C, C, C, C, R, R, R, R, R,  
R, R, C, C, C, W, C, C, C, C, C, C, R, C, C, C, C

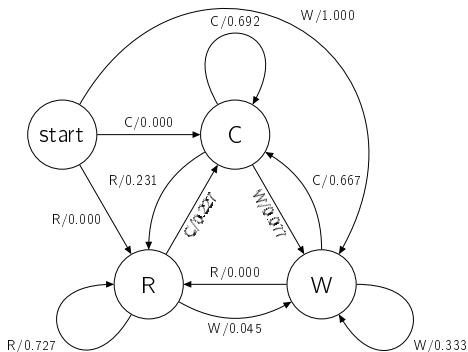
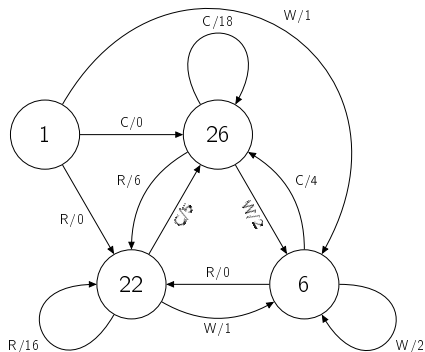
- Counts and ML estimates:

$c(\cdot, \cdot)$	R	W	C	sum	$p^{\text{MLE}}$	R	W	C
start	0	1	0	1	start	0.000	1.000	0.000
R	16	1	5	22	R	0.727	0.045	0.227
W	0	2	4	6	W	0.000	0.333	0.667
C	6	2	18	26	C	0.231	0.077	0.692

$$p_{x_{i-1}, x_i}^{\text{MLE}} = \frac{c(x_{i-1}, x_i)}{\sum_x c(x_{i-1}, x)}$$

$$p_{R,C}^{\text{MLE}} = \frac{5}{16 + 1 + 5} = 0.227$$

# Example: Maximum Likelihood Estimation



# Example: Orders

- Some raw data:

W, W, C, C, W, W, C, R, C, R, W, C, C, C, R, R, R, R, C,  
C, R, R, R, R, R, R, R, R, R, C, C, C, C, C, R, R, R, R, R,  
R, R, C, C, C, W, C, C, C, C, C, C, R, C, C, C, C

- Data sampled from MLE Markov model, order 1:

W, W, C, R, R, R, R, R, R, R, R, R, R, R, R, R, R, R, R,  
C, C, C, C, C, C, W, W, C, C, C, R, R, R, C, C, W, C, C,  
C, C, C, R, R, R, R, R, C, R, R, C, R, R, R, R, R

- Data sampled from MLE Markov model, order 0:

C, R, C, R, R, R, R, C, R, R, C, C, R, C, C, R, R, R, R, C,  
C, C, R, C, R, W, R, C, C, C, W, C, R, C, C, W, C, C, C,  
C, R, R, C, C, C, R, C, R, R, C, R, C, R, W, R

# Recap: Non-Hidden Markov Models

- Use *states* to encode limited amount of information ...
  - About the past.
- Current state is known .
- Log likelihood just depends on pair counts.

$$L(x_1^N) = \sum_{x_{i-1}, x_i} c(x_{i-1}, x_i) \log p_{x_{i-1}, x_i}$$

- MLE: count and normalize.

$$p_{x_{i-1}, x_i}^{\text{MLE}} = \frac{c(x_{i-1}, x_i)}{\sum_x c(x_{i-1}, x)}$$

- Easy beezy.

## Part II

# Discrete Hidden Markov Models

# Case Study: Austin Weather 2.0

- Ignore rain; one sample every two weeks:

C, W, C, C, C, C, W, C, C, C, C, C, C, W, W, C, W, C, W,  
W, C, W, C, W, C, C, C, C, C, C, C, C, C, C, C, C, C, C,  
W, C, C, C, W, W, C, C, W, W, C, W, C, W, C, C, C, C, C,  
C, C, C, C, C, C, C, C, W, C, W, C, C, W, W, C, W, W, W,  
C, W, C, C, C, C, C, C, C, C, C, C, W, C, W, W, W, C, C,  
C, C, C, W, C, C, W, C, C, C, C, C, C, C, C, C, C, C, W

- Does system have state/memory?

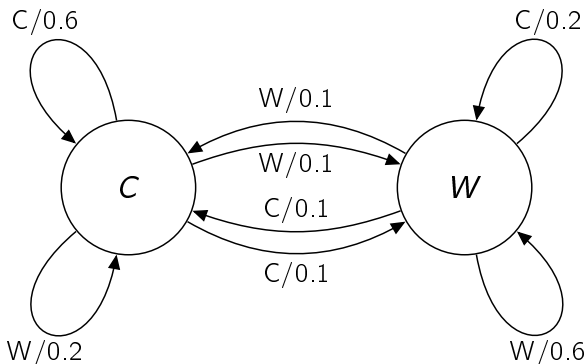
# Another View

C	W	C	C	C	C	W	C	C	C	C	C	C	W	W	C	W	C	W	W	C	W	C	W	C	C	
C	C	C	C	C	C	C	C	C	C	C	C	C	W	C	C	C	W	W	C	C	W	W	C	W	C	W
C	C	C	C	C	C	C	C	C	C	C	C	C	W	C	W	C	C	W	W	C	W	W	W	C	W	
C	C	C	C	C	C	C	C	C	C	C	W	C	W	W	W	C	C	C	C	C	W	C	C	W	C	C
C	C	C	C	C	C	C	C	C	C	W	C	C	W	W	C	W	C	C	C	W	C	W	C	W	C	C
C	C	C	C	C	C	W	C	C	C	C	C	W	C	C	C	W	C	W	C	W	C	C	W	C	W	
C	C	C	C	C	C	C	C	C	C	C	C	C	W	C	C	C	W	W	C	C	C	W	C	W	C	

- Does system have memory?
- How many states?

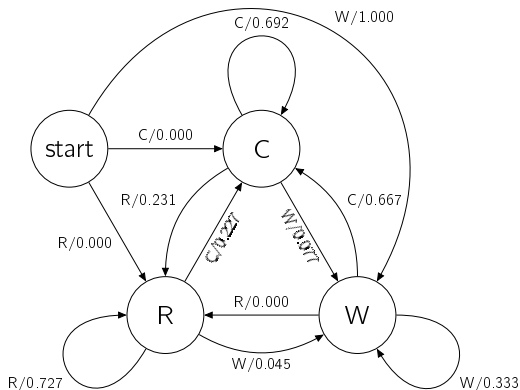
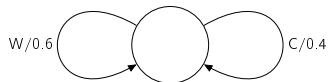
# A Hidden Markov Model

- For simplicity, no separate *start* state.
  - Always start in calm state *c*.



- Why is state “hidden”?
  - What are conditions for state to be non-hidden?

# Contrast: Non-Hidden Markov Models



# Back to Coins: Hidden Information

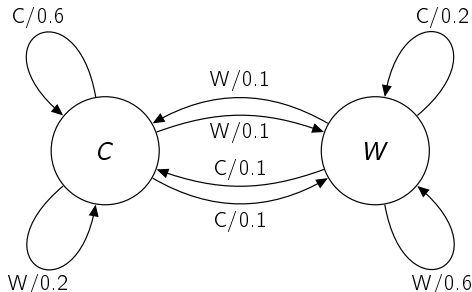
- Memory-less example:
  - Coin 0:  $p_H = 0.7, p_T = 0.3$   
Coin 1:  $p_H = 0.9, p_T = 0.1$   
Coin 2:  $p_H = 0.2, p_T = 0.8$
  - Experiment: Flip Coin 0. If outcome is H, flip Coin 1 and record ; else flip Coin 2 and record.
  - Coin 0 flips outcomes are hidden!
    - What is the probability of the sequence: H T T T ?
    - $p(H) = 0.9 \times 0.7 + 0.2 \times 0.3; p(T) = 0.1 \times 0.7 + 0.8 \times 0.3$
- An example with memory:
  - 2 coins, flip each twice. Record first flip, use second to determine which coin to flip.
  - No way to know the outcome of even flips.
  - Order matters now and . . .
  - Cannot uniquely determine which state sequence produced the observed output sequence

# Why Hidden State?

- No “simple” way to determine state given observed.
  - If see “*W*”, doesn’t mean windy season started.
- Speech recognition: one HMM per word.
  - Each state represents different sound in word.
  - How to tell from observed when state switches?
- Hidden models can model same stuff as non-hidden ...
  - Using much fewer states.
- Pop quiz: name a hidden model with no memory.

# The Problem With Hidden State

- For observed  $\mathbf{x} = x_1, \dots, x_N$ , what is hidden state  $\mathbf{h}$ ?
  - Corresponding state sequence  $\mathbf{h} = h_1, \dots, h_{N+1}$ .
- In non-hidden model, how many  $\mathbf{h}$  possible given  $\mathbf{x}$ ?
- In hidden model, what  $\mathbf{h}$  are possible given  $\mathbf{x}$ ?



- This makes everything difficult.

# Three Key Tasks for HMM's

- 1 Find single best path in HMM given observed  $\mathbf{x}$ .
  - *e.g.*, when did windy season begin?
  - *e.g.*, when did each sound in word begin?
- 2 Find total likelihood  $P(\mathbf{x})$  of observed.
  - *e.g.*, to pick which word assigns highest likelihood.
- 3 Find ML estimates for parameters of HMM.
  - *i.e.*, estimate arc probabilities to match training data.

These problems are easy to solve for a state-observable Markov model. More complicated for a HMM as we have to consider all possible state sequences.

# Where Are We?

- 1 Computing the Best Path
- 2 Computing the Likelihood of Observations
- 3 Estimating Model Parameters
- 4 Discussion

# What We Want to Compute

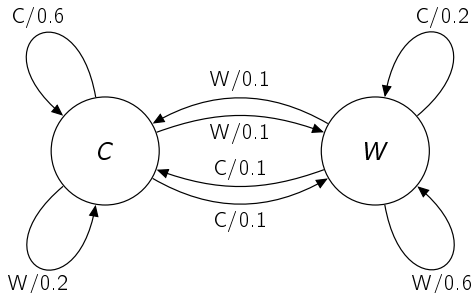
- Given observed, *e.g.*,  $\mathbf{x} = C, W, C, C, W, \dots$ 
  - Find state sequence  $\mathbf{h}^*$  with highest likelihood.

$$\mathbf{h}^* = \arg \max_{\mathbf{h}} P(\mathbf{h}, \mathbf{x})$$

- Why is this easy for non-hidden model?
- Given state sequence  $\mathbf{h}$ , how to compute  $P(\mathbf{h}, \mathbf{x})$ ?
  - Same as for non-hidden model.
  - Multiply all arc probabilities along path.

# Likelihood of Single State Sequence

- Some data:  $\mathbf{x} = W, C, C$ .
- A state sequence:  $\mathbf{h} = c, c, c, w$ .



- Likelihood of path:

$$P(\mathbf{h}, \mathbf{x}) = 0.2 \times 0.6 \times 0.1 = 0.012$$

# What We Want to Compute

- Given observed, *e.g.*,  $\mathbf{x} = C, W, C, C, W, \dots$ 
  - Find state sequence  $\mathbf{h}^*$  with highest likelihood.

$$\mathbf{h}^* = \arg \max_{\mathbf{h}} P(\mathbf{h}, \mathbf{x})$$

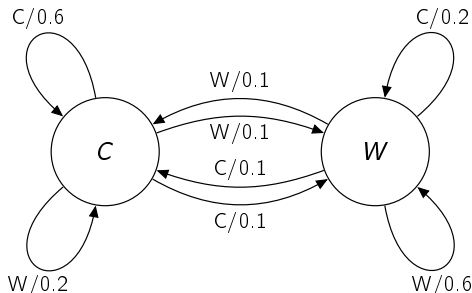
- Let's start with simpler problem:
  - Find likelihood of best state sequence  $P_{\text{best}}(\mathbf{x})$ .
  - Worry about identity of best sequence later.

$$P_{\text{best}}(\mathbf{x}) = \max_{\mathbf{h}} P(\mathbf{h}, \mathbf{x})$$

# What's the Problem?

$$P_{\text{best}}(\mathbf{x}) = \max_{\mathbf{h}} P(\mathbf{h}, \mathbf{x})$$

- For observation sequence of length  $N$  ...
  - How many different possible state sequences  $\mathbf{h}$ ?



- How in blazes can we do max ...
  - Over exponential number of state sequences?

# Dynamic Programming

- Let  $S_0$  be start state; *e.g.*, the calm season  $c$ .
- Let  $\mathcal{P}(S, t)$  be set of paths of length  $t \dots$ 
  - Starting at start state  $S_0$  and ending at  $S \dots$
  - Consistent with observed  $x_1, \dots, x_t$ .
- Any path  $p \in \mathcal{P}(S, t)$  must be composed of ...
  - Path of length  $t - 1$  to predecessor state  $S' \rightarrow S \dots$
  - Followed by arc from  $S'$  to  $S$  labeled with  $x_t$ .
  - This decomposition is unique.

$$\mathcal{P}(S, t) = \bigcup_{S' \xrightarrow{x_t} S} \mathcal{P}(S', t - 1) \cdot (S' \xrightarrow{x_t} S)$$

# Dynamic Programming

$$\mathcal{P}(S, t) = \bigcup_{S' \xrightarrow{x_t} S} \mathcal{P}(S', t-1) \cdot (S' \xrightarrow{x_t} S)$$

- Let  $\hat{\alpha}(S, t)$  = likelihood of best path of length  $t$  ...
  - Starting at start state  $S_0$  and ending at  $S$ .
  - $P(p)$  = prob of path  $p$  = product of arc probs.

$$\begin{aligned}\hat{\alpha}(S, t) &= \max_{p \in \mathcal{P}(S, t)} P(p) \\&= \max_{p' \in \mathcal{P}(S', t-1), S' \xrightarrow{x_t} S} P(p' \cdot (S' \xrightarrow{x_t} S)) \\&= \max_{S' \xrightarrow{x_t} S} P(S' \xrightarrow{x_t} S) \max_{p' \in \mathcal{P}(S', t-1)} P(p') \\&= \max_{S' \xrightarrow{x_t} S} P(S' \xrightarrow{x_t} S) \times \hat{\alpha}(S', t-1)\end{aligned}$$

# What Were We Computing Again?

- Assume observed  $\mathbf{x}$  of length  $T$ .
  - Want likelihood of best path of length  $T \dots$
  - Starting at start state  $S_0$  and ending anywhere.

$$P_{\text{best}}(\mathbf{x}) = \max_{\mathbf{h}} P(\mathbf{h}, \mathbf{x}) = \max_S \hat{\alpha}(S, T)$$

- If can compute  $\hat{\alpha}(S, T)$ , we are done.
- If know  $\hat{\alpha}(S, t - 1)$  for all  $S$ , easy to compute  $\hat{\alpha}(S, t)$ :

$$\hat{\alpha}(S, t) = \max_{S' \xrightarrow{x_t} S} P(S' \xrightarrow{x_t} S) \times \hat{\alpha}(S', t - 1)$$

- This looks promising  $\dots$

# The Viterbi Algorithm

- $\hat{\alpha}(S, 0) = 1$  for  $S = S_0$ , 0 otherwise.
- For  $t = 1, \dots, T$ :
  - For each state  $S$ :

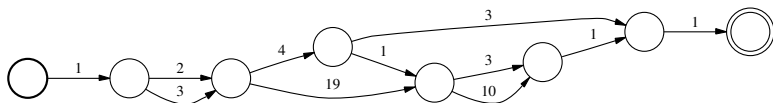
$$\hat{\alpha}(S, t) = \max_{S' \xrightarrow{x_t} S} P(S' \xrightarrow{x_t} S) \times \hat{\alpha}(S', t-1)$$

- The end.

$$P_{\text{best}}(\mathbf{x}) = \max_{\mathbf{h}} P(\mathbf{h}, \mathbf{x}) = \max_S \hat{\alpha}(S, T)$$

# Viterbi and Shortest Path

- Equivalent to shortest path problem.



- One “state” for each state/time pair  $(S, t)$ .
- Iterate through “states” in topological order:
  - All arcs go forward in time.
  - If order “states” by time, valid ordering.

$$d(S) = \min_{S' \rightarrow S} \{d(S') + \text{distance}(S', S)\}$$

$$\hat{\alpha}(S, t) = \max_{S' \xrightarrow{x_t} S} P(S' \xrightarrow{x_t} S) \times \hat{\alpha}(S', t-1)$$

# Identifying the Best Path

- Wait! We can calc likelihood of best path:

$$P_{\text{best}}(\mathbf{x}) = \max_{\mathbf{h}} P(\mathbf{h}, \mathbf{x})$$

- What we really wanted: *identity* of best path.
  - *i.e.*, the best state sequence  $\mathbf{h}$ .
- Basic idea: for each  $S, t \dots$ 
  - Record identity  $S_{\text{prev}}(S, t)$  of previous state  $S'$  ...
  - In best path of length  $t$  ending at state  $S$ .
- Find best final state.
  - Backtrace best previous states until reach start state.

# The Viterbi Algorithm With Backtrace

- $\hat{\alpha}(S, 0) = 1$  for  $S = S_0$ , 0 otherwise.
- For  $t = 1, \dots, T$ :
  - For each state  $S$ :

$$\hat{\alpha}(S, t) = \max_{S' \xrightarrow{x_t} S} P(S' \xrightarrow{x_t} S) \times \hat{\alpha}(S', t-1)$$

$$S_{\text{prev}}(S, t) = \arg \max_{S' \xrightarrow{x_t} S} P(S' \xrightarrow{x_t} S) \times \hat{\alpha}(S', t-1)$$

- The end.

$$P_{\text{best}}(\mathbf{x}) = \max_S \hat{\alpha}(S, T)$$

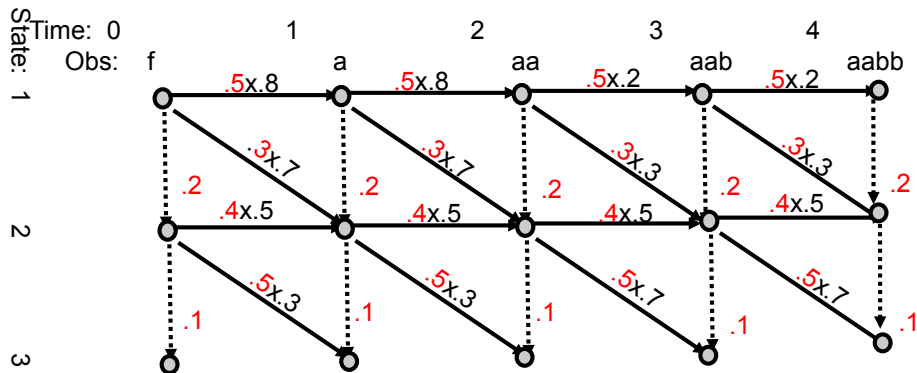
$$S_{\text{final}}(\mathbf{x}) = \arg \max_S \hat{\alpha}(S, T)$$

# The Backtrace

- $S_{\text{cur}} \leftarrow S_{\text{final}}(\mathbf{x})$
- for  $t$  in  $T, \dots, 1$ :
  - $S_{\text{cur}} \leftarrow S_{\text{prev}}(S_{\text{cur}}, t)$
- The best state sequence is ...
  - List of states traversed in reverse order.

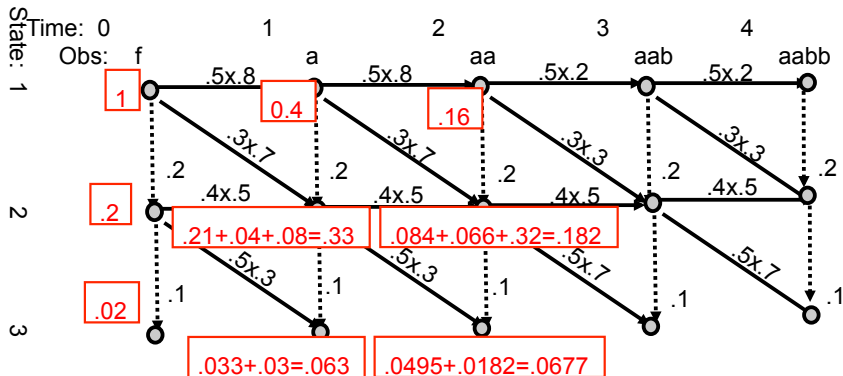
# Illustration with a trellis

State transition diagram in time



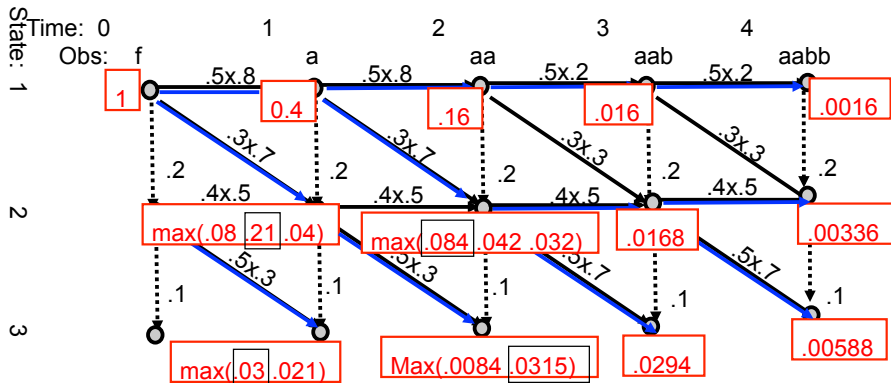
# Illustration with a trellis (contd.)

## Accumulating scores



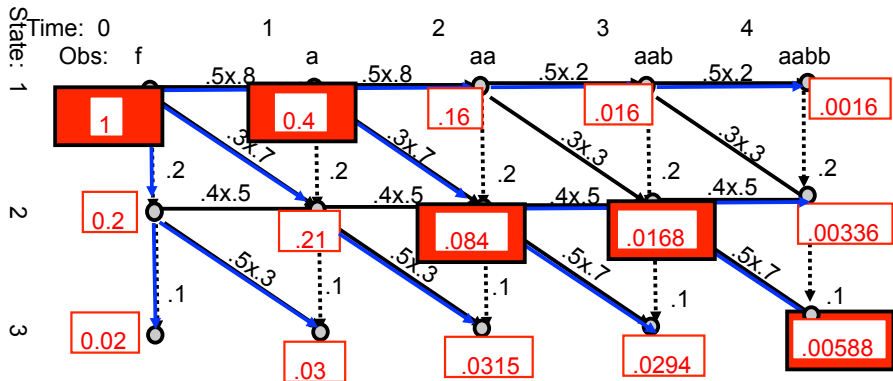
# Viterbi algorithm

## Accumulating scores



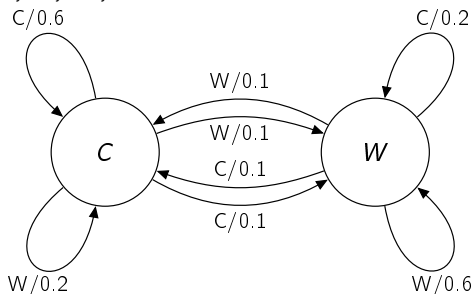
# Best path through the trellis

Accumulating scores



# Example

- Some data: C, C, W, W.



$\hat{\alpha}$	0	1	2	3	4
c	1.000	0.600	0.360	0.072	0.014
w	0.000	0.100	0.060	0.036	<b>0.022</b>

$$\begin{aligned}
 \hat{\alpha}(c, 2) &= \max\{P(c \xrightarrow{C} c) \times \hat{\alpha}(c, 1), P(w \xrightarrow{C} c) \times \hat{\alpha}(w, 1)\} \\
 &= \max\{0.6 \times 0.6, 0.1 \times 0.1\} = 0.36
 \end{aligned}$$

# Example: The Backtrace

$S_{\text{prev}}$	0	1	2	3	4
$c$		<b>c</b>	<b>c</b>	$c$	$c$
$w$		$c$	$c$	<b>c</b>	<b>w</b>

$$\mathbf{h}^* = \arg \max_{\mathbf{h}} P(\mathbf{h}, \mathbf{x}) = (c, c, c, w, w)$$

- The data: C, C, W, W.
- Calm season switching to windy season.

# Recap: The Viterbi Algorithm

- Given observed  $\mathbf{x}$ , ...
  - Exponential number of hidden sequences  $\mathbf{h}$ .
- Can find likelihood and identity of best path ...
  - Efficiently using dynamic programming.
- What is time complexity?

# Where Are We?

- 1 Computing the Best Path
- 2 Computing the Likelihood of Observations
- 3 Estimating Model Parameters
- 4 Discussion

# What We Want to Compute

- Given observed, *e.g.*,  $\mathbf{x} = C, W, C, C, W, \dots$ 
  - Find total likelihood  $P(\mathbf{x})$ .
- Need to sum likelihood over all hidden sequences:

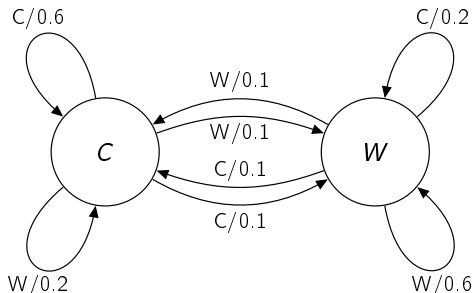
$$P(\mathbf{x}) = \sum_{\mathbf{h}} P(\mathbf{h}, \mathbf{x})$$

- Given state sequence  $\mathbf{h}$ , how to compute  $P(\mathbf{h}, \mathbf{x})$ ?
  - Multiply all arc probabilities along path.
- Why is this sum easy for non-hidden model?

# What's the Problem?

$$P(\mathbf{x}) = \sum_{\mathbf{h}} P(\mathbf{h}, \mathbf{x})$$

- For observation sequence of length  $N$  ...
  - How many different possible state sequences  $\mathbf{h}$ ?



- How in blazes can we do sum ...
  - Over exponential number of state sequences?

# Dynamic Programming

- Let  $\mathcal{P}(S, t)$  be set of paths of length  $t \dots$ 
  - Starting at start state  $S_0$  and ending at  $S \dots$
  - Consistent with observed  $x_1, \dots, x_t$ .
- Any path  $p \in \mathcal{P}(S, t)$  must be composed of ...
  - Path of length  $t - 1$  to predecessor state  $S' \rightarrow S \dots$
  - Followed by arc from  $S'$  to  $S$  labeled with  $x_t$ .

$$\mathcal{P}(S, t) = \bigcup_{S' \xrightarrow{x_t} S} \mathcal{P}(S', t - 1) \cdot (S' \xrightarrow{x_t} S)$$

# Dynamic Programming

$$\mathcal{P}(S, t) = \bigcup_{S' \xrightarrow{x_t} S} \mathcal{P}(S', t-1) \cdot (S' \xrightarrow{x_t} S)$$

- Let  $\alpha(S, t)$  = sum of likelihoods of paths of length  $t$  ...
  - Starting at start state  $S_0$  and ending at  $S$ .

$$\begin{aligned}\alpha(S, t) &= \sum_{p \in \mathcal{P}(S, t)} P(p) \\&= \sum_{p' \in \mathcal{P}(S', t-1), S' \xrightarrow{x_t} S} P(p' \cdot (S' \xrightarrow{x_t} S)) \\&= \sum_{S' \xrightarrow{x_t} S} P(S' \xrightarrow{x_t} S) \sum_{p' \in \mathcal{P}(S', t-1)} P(p') \\&= \sum_{S' \xrightarrow{x_t} S} P(S' \xrightarrow{x_t} S) \times \alpha(S', t-1)\end{aligned}$$

# What Were We Computing Again?

- Assume observed  $\mathbf{x}$  of length  $T$ .
  - Want sum of likelihoods of paths of length  $T \dots$
  - Starting at start state  $S_0$  and ending anywhere.

$$P(\mathbf{x}) = \sum_{\mathbf{h}} P(\mathbf{h}, \mathbf{x}) = \sum_S \alpha(S, T)$$

- If can compute  $\alpha(S, T)$ , we are done.
- If know  $\alpha(S, t - 1)$  for all  $S$ , easy to compute  $\alpha(S, t)$ :

$$\alpha(S, t) = \sum_{S' \xrightarrow{x_t} S} P(S' \xrightarrow{x_t} S) \times \alpha(S', t - 1)$$

- This looks promising ...

# The Forward Algorithm

- $\alpha(S, 0) = 1$  for  $S = S_0$ , 0 otherwise.
- For  $t = 1, \dots, T$ :
  - For each state  $S$ :

$$\alpha(S, t) = \sum_{S' \xrightarrow{x_t} S} P(S' \xrightarrow{x_t} S) \times \alpha(S', t-1)$$

- The end.

$$P(\mathbf{x}) = \sum_{\mathbf{h}} P(\mathbf{h}, \mathbf{x}) = \sum_{\mathbf{S}} \alpha(\mathbf{S}, T)$$

# Viterbi vs. Forward

- The goal:

$$P_{\text{best}}(\mathbf{x}) = \max_{\mathbf{h}} P(\mathbf{h}, \mathbf{x}) = \max_S \hat{\alpha}(S, T)$$

$$P(\mathbf{x}) = \sum_{\mathbf{h}} P(\mathbf{h}, \mathbf{x}) = \sum_S \alpha(S, T)$$

- The invariant.

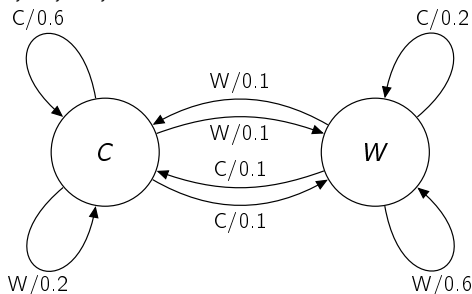
$$\hat{\alpha}(S, t) = \max_{S' \xrightarrow{x_t} S} P(S' \xrightarrow{x_t} S) \times \hat{\alpha}(S', t-1)$$

$$\alpha(S, t) = \sum_{S' \xrightarrow{x_t} S} P(S' \xrightarrow{x_t} S) \times \alpha(S', t-1)$$

- Just replace all max's with sums (any *semiring* will do).

# Example

- Some data: C, C, W, W.



$\alpha$	0	1	2	3	4
$c$	1.000	0.600	0.370	0.082	<b>0.025</b>
$w$	0.000	0.100	0.080	0.085	<b>0.059</b>

$$\begin{aligned}\alpha(c, 2) &= P(c \xrightarrow{c} c) \times \alpha(c, 1) + P(w \xrightarrow{c} c) \times \alpha(w, 1) \\ &= 0.6 \times 0.6 + 0.1 \times 0.1 = 0.37\end{aligned}$$

# Recap: The Forward Algorithm

- Can find total likelihood  $P(\mathbf{x})$  of observed ...
  - Using very similar algorithm to Viterbi algorithm.
- Just replace max's with sums.
- Same time complexity.

# Where Are We?

- 1 Computing the Best Path
- 2 Computing the Likelihood of Observations
- 3 Estimating Model Parameters**
- 4 Discussion

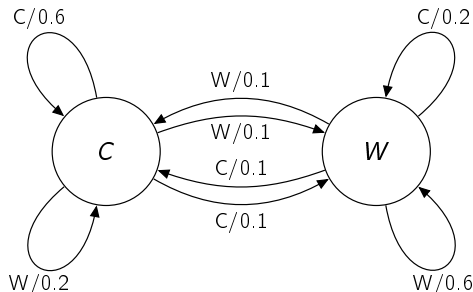
# Training the Parameters of an HMM

- Given training data  $\mathbf{x} \dots$
- Estimate parameters of model  $\dots$ 
  - To maximize likelihood of training data.

$$P(\mathbf{x}) = \sum_{\mathbf{h}} P(\mathbf{h}, \mathbf{x})$$

# What Are The Parameters?

- One parameter for each arc:



- Identify arc by source  $S$ , destination  $S'$ , and label  $x$ :  $p_{S \xrightarrow{x} S'}$ .
- Probs of arcs leaving same state must sum to 1:

$$\sum_{x, S'} p_{S \xrightarrow{x} S'} = 1 \quad \text{for all } S$$

# What Did We Do For Non-Hidden Again?

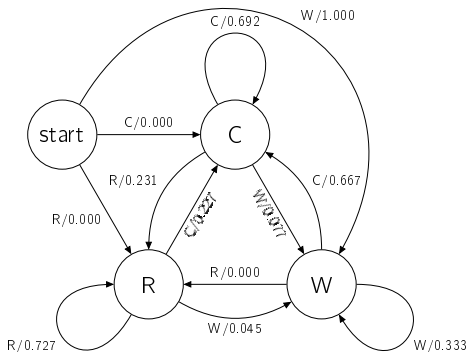
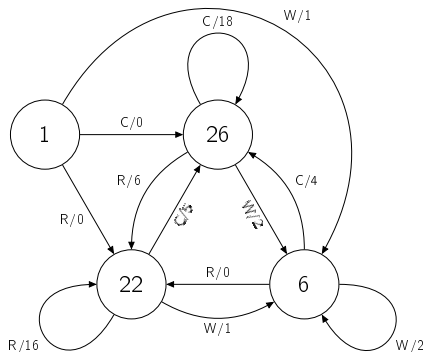
- Likelihood of single path: product of arc probabilities.
- Log likelihood can be written as:

$$L(x_1^N) = \sum_{S \xrightarrow{x} S'} c(S \xrightarrow{x} S') \log p_{S \xrightarrow{x} S'}$$

- Just depends on counts  $c(S \xrightarrow{x} S')$  of each arc.
- Each source state corresponds to multinomial ...
  - With nonoverlapping parameters.
- ML estimation for multinomials: count and normalize!

$$p_{S \xrightarrow{x} S'}^{\text{MLE}} = \frac{c(S \xrightarrow{x} S')}{\sum_{x, S'} c(S \xrightarrow{x} S')}$$

# Example: Non-Hidden Estimation



# How Do We Train Hidden Models?

- Hmmmm, I know this one ...

# Review: The EM Algorithm

- General way to train parameters in hidden models ...
  - To optimize likelihood.
- Guaranteed to improve likelihood in each iteration.
  - Only finds local optimum.
  - Seeding matters.

# The EM Algorithm

- Initialize parameter values somehow.
- For each iteration ...
- Expectation step: compute posterior (count) of each  $\mathbf{h}$ .

$$\tilde{P}(\mathbf{h}|\mathbf{x}) = \frac{P(\mathbf{h}, \mathbf{x})}{\sum_{\mathbf{h}} P(\mathbf{h}, \mathbf{x})}$$

- Maximization step: update parameters.
  - Instead of data  $\mathbf{x}$  with unknown  $\mathbf{h}$ , pretend ...
  - *Non-hidden* data where ...
  - (Fractional) count of each  $(\mathbf{h}, \mathbf{x})$  is  $\tilde{P}(\mathbf{h}|\mathbf{x})$ .

# Applying EM to HMM's: The E Step

- Compute posterior (count) of each  $\mathbf{h}$ .

$$\tilde{P}(\mathbf{h}|\mathbf{x}) = \frac{P(\mathbf{h}, \mathbf{x})}{\sum_{\mathbf{h}} P(\mathbf{h}, \mathbf{x})}$$

- How to compute prob of single path  $P(\mathbf{h}, \mathbf{x})$ ?
  - Multiply arc probabilities along path.
- How to compute denominator?
  - This is just total likelihood of observed  $P(\mathbf{x})$ .

$$P(\mathbf{x}) = \sum_{\mathbf{h}} P(\mathbf{h}, \mathbf{x})$$

- This looks vaguely familiar.

# Applying EM to HMM's: The M Step

- Non-hidden case: single path  $\mathbf{h}$  with count 1.
  - Total count of arc is count of arc in  $\mathbf{h}$ :

$$c(S \xrightarrow{x} S') = c_{\mathbf{h}}(S \xrightarrow{x} S')$$

- Normalize.

$$p_{S \xrightarrow{x} S'}^{\text{MLE}} = \frac{c(S \xrightarrow{x} S')}{\sum_{x, S'} c(S \xrightarrow{x} S')}$$

- Hidden case: every path  $\mathbf{h}$  has count  $\tilde{P}(\mathbf{h}|\mathbf{x})$ .
  - Total count of arc is weighted sum ...
  - Of count of arc in each  $\mathbf{h}$ .

$$c(S \xrightarrow{x} S') = \sum_{\mathbf{h}} \tilde{P}(\mathbf{h}|\mathbf{x}) c_{\mathbf{h}}(S \xrightarrow{x} S')$$

- Normalize as before.

# What's the Problem?

- Need to sum over exponential number of  $h$ :

$$c(S \xrightarrow{x} S') = \sum_{\mathbf{h}} \tilde{P}(\mathbf{h}|\mathbf{x}) c_{\mathbf{h}}(S \xrightarrow{x} S')$$

- If only we had an algorithm for doing this type of thing.

# The Game Plan

- Decompose sum by time (*i.e.*, position in  $\mathbf{x}$ ).
  - Find count of each arc at each “time”  $t$ .

$$c(S \xrightarrow{x} S') = \sum_{t=1}^T c(S \xrightarrow{x} S', t) = \sum_{t=1}^T \sum_{\mathbf{h} \in \mathcal{P}(S \xrightarrow{x} S', t)} \tilde{P}(\mathbf{h}|\mathbf{x})$$

- $\mathcal{P}(S \xrightarrow{x} S', t)$  are paths where arc at time  $t$  is  $S \xrightarrow{x} S'$ .
- $\mathcal{P}(S \xrightarrow{x} S', t)$  is empty if  $x \neq x_t$ .
- Otherwise, use dynamic programming to compute

$$c(S \xrightarrow{x_t} S', t) \equiv \sum_{\mathbf{h} \in \mathcal{P}(S \xrightarrow{x_t} S', t)} \tilde{P}(\mathbf{h}|\mathbf{x})$$

# Let's Rearrange Some

- Recall we can compute  $P(\mathbf{x})$  using Forward algorithm:

$$\tilde{P}(\mathbf{h}|\mathbf{x}) = \frac{P(\mathbf{h}, \mathbf{x})}{P(\mathbf{x})}$$

- Some paraphrasing:

$$\begin{aligned} c(S \xrightarrow{x_t} S', t) &= \sum_{\mathbf{h} \in \mathcal{P}(S \xrightarrow{x_t} S', t)} \tilde{P}(\mathbf{h}|\mathbf{x}) \\ &= \frac{1}{P(\mathbf{x})} \sum_{\mathbf{h} \in \mathcal{P}(S \xrightarrow{x_t} S', t)} P(\mathbf{h}, \mathbf{x}) \\ &= \frac{1}{P(\mathbf{x})} \sum_{p \in \mathcal{P}(S \xrightarrow{x_t} S', t)} P(p) \end{aligned}$$

# What We Need

- Goal: sum over all paths  $p \in \mathcal{P}(S \xrightarrow{x_t} S', t)$ .
  - Arc at time  $t$  is  $S \xrightarrow{x_t} S'$ .
- Let  $\mathcal{P}_i(S, t)$  be set of (initial) paths of length  $t \dots$ 
  - Starting at start state  $S_0$  and ending at  $S \dots$
  - Consistent with observed  $x_1, \dots, x_t$ .
- Let  $\mathcal{P}_f(S, t)$  be set of (final) paths of length  $T - t \dots$ 
  - Starting at state  $S$  and ending at any state  $\dots$
  - Consistent with observed  $x_{t+1}, \dots, x_T$ .
- Then:

$$\mathcal{P}(S \xrightarrow{x_t} S', t) = \mathcal{P}_i(S, t-1) \cdot (S \xrightarrow{x_t} S') \cdot \mathcal{P}_f(S', t)$$

# Translating Path Sets to Probabilities

$$\mathcal{P}(S \xrightarrow{x_t} S', t) = \mathcal{P}_i(S, t-1) \cdot (S \xrightarrow{x_t} S') \cdot \mathcal{P}_f(S', t)$$

- Let  $\alpha(S, t)$  = sum of likelihoods of paths of length  $t \dots$ 
  - Starting at start state  $S_0$  and ending at  $S$ .
- Let  $\beta(S, t) =$  sum of likelihoods of paths of length  $T - t \dots$ 
  - Starting at state  $S$  and ending at any state.

$$\begin{aligned} c(S \xrightarrow{x_t} S', t) &= \frac{1}{P(\mathbf{x})} \sum_{p \in \mathcal{P}(S \xrightarrow{x_t} S', t)} P(p) \\ &= \frac{1}{P(\mathbf{x})} \sum_{p_i \in \mathcal{P}_i(S, t-1), p_f \in \mathcal{P}_f(S', t)} P(p_i \cdot (S \xrightarrow{x_t} S') \cdot p_f) \\ &= \frac{1}{P(\mathbf{x})} \times p_{S \xrightarrow{x_t} S'} \sum_{p_i \in \mathcal{P}_i(S, t-1)} P(p_i) \sum_{p_f \in \mathcal{P}_f(S', t)} P(p_f) \\ &= \frac{1}{P(\mathbf{x})} \times p_{S \xrightarrow{x_t} S'} \times \alpha(S, t-1) \times \beta(S', t) \end{aligned}$$

# Mini-Recap

- To do ML estimation in M step ...
  - Need count of each arc:  $c(S \xrightarrow{x} S')$ .
- Decompose count of arc by time:

$$c(S \xrightarrow{x} S') = \sum_{t=1}^T c(S \xrightarrow{x} S', t)$$

- Can compute count at time efficiently ...
  - If have *forward* probabilities  $\alpha(S, t)$  ...
  - And *backward* probabilities  $\beta(S, T)$ .

$$c(S \xrightarrow{x_t} S', t) = \frac{1}{P(\mathbf{x})} \times p_{S \xrightarrow{x_t} S'} \times \alpha(S, t-1) \times \beta(S', t)$$

# The Forward-Backward Algorithm (1 iter)

- Apply Forward algorithm to compute  $\alpha(S, t)$ ,  $P(\mathbf{x})$ .
- Apply Backward algorithm to compute  $\beta(S, t)$ .
- For each arc  $S \xrightarrow{x_t} S'$  and time  $t \dots$ 
  - Compute posterior count of arc at time  $t$  if  $x = x_t$ .

$$c(S \xrightarrow{x_t} S', t) = \frac{1}{P(\mathbf{x})} \times p_{S \xrightarrow{x_t} S'} \times \alpha(S, t-1) \times \beta(S', t)$$

- Sum to get total counts for each arc.

$$c(S \xrightarrow{x} S') = \sum_{t=1}^T c(S \xrightarrow{x} S', t)$$

- For each arc, find ML estimate of parameter:

$$p_{S \xrightarrow{x} S'}^{\text{MLE}} = \frac{c(S \xrightarrow{x} S')}{\sum_{x, S'} c(S \xrightarrow{x} S')}$$

# The Forward Algorithm

- $\alpha(S, 0) = 1$  for  $S = S_0$ , 0 otherwise.
- For  $t = 1, \dots, T$ :
  - For each state  $S$ :

$$\alpha(S, t) = \sum_{S' \xrightarrow{x_t} S} p_{S' \xrightarrow{x_t} S} \times \alpha(S', t-1)$$

- The end.

$$P(\mathbf{x}) = \sum_{\mathbf{h}} P(\mathbf{h}, \mathbf{x}) = \sum_{\mathbf{S}} \alpha(\mathbf{S}, T)$$

# The Backward Algorithm

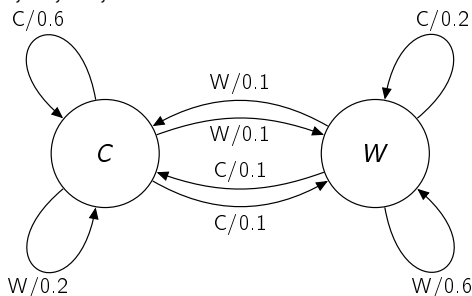
- $\beta(S, T) = 1$  for all  $S$ .
- For  $t = T - 1, \dots, 0$ :
  - For each state  $S$ :

$$\beta(S, t) = \sum_{S \xrightarrow{x_{t+1}} S'} p_{S \xrightarrow{x_{t+1}} S'} \times \beta(S', t + 1)$$

- Pop quiz: how to compute  $P(\mathbf{x})$  from  $\beta$ 's?

# Example: The Forward Pass

- Some data:  $C, C, W, W$ .

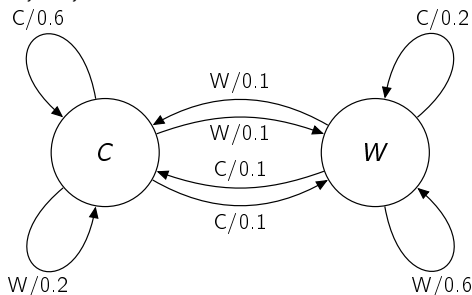


$\alpha$	0	1	2	3	4
$c$	1.000	0.600	0.370	0.082	<b>0.025</b>
$w$	0.000	0.100	0.080	0.085	<b>0.059</b>

$$\begin{aligned}\alpha(c, 2) &= p_{c \rightarrow c} \times \alpha(c, 1) + p_{w \rightarrow c} \times \alpha(w, 1) \\ &= 0.6 \times 0.6 + 0.1 \times 0.1 = 0.37\end{aligned}$$

# The Backward Pass

- The data: C, C, W, W.



$\beta$	0	1	2	3	4
$c$	<b>0.084</b>	0.123	0.130	0.300	1.000
$w$	0.033	0.103	0.450	0.700	1.000

$$\begin{aligned}\beta(c, 2) &= p_{c \rightarrow c}^w \times \beta(c, 3) + p_{c \rightarrow w}^w \times \beta(w, 3) \\ &= 0.2 \times 0.3 + 0.1 \times 0.7 = 0.13\end{aligned}$$

# Computing Arc Posteriors

$\alpha, \beta$	0	1	2	3	4
$c$	1.000	0.600	0.370	0.082	0.025
$w$	0.000	0.100	0.080	0.085	0.059
$c$	<b>0.084</b>	0.123	0.130	0.300	1.000
$w$	0.033	0.103	0.450	0.700	1.000

$c(S \xrightarrow{x} S', t)$	$p_{S \xrightarrow{x} S'}$	1	2	3	4
$c \xrightarrow{C} c$	0.6	0.878	0.556	0.000	0.000
$c \xrightarrow{W} c$	0.2	0.000	0.000	0.264	0.195
$c \xrightarrow{C} w$	0.1	0.122	0.321	0.000	0.000
$c \xrightarrow{W} w$	0.1	0.000	0.000	0.308	0.098
$w \xrightarrow{C} w$	0.2	0.000	0.107	0.000	0.000
$w \xrightarrow{W} w$	0.6	0.000	0.000	0.400	0.606
$w \xrightarrow{C} c$	0.1	0.000	0.015	0.000	0.000
$w \xrightarrow{W} c$	0.1	0.000	0.000	0.029	0.101

# Computing Arc Posteriors

$\alpha, \beta$	0	1	2	3	4
$c$	1.000	0.600	0.370	0.082	0.025
$w$	0.000	0.100	0.080	0.085	0.059
$c$	<b>0.084</b>	0.123	0.130	0.300	1.000
$w$	0.033	0.103	0.450	0.700	1.000

$c(S \xrightarrow{x} S', t)$	$p_{S \xrightarrow{x} S'}$	1	2	3	4
$c \xrightarrow{c} c$	0.6	0.878	0.556	0.000	0.000
$c \xrightarrow{w} c$	0.2	0.000	0.000	0.264	0.195
...	...	...	...	...	...
...	...	...	...	...	...

$$\begin{aligned}
 c(c \xrightarrow{c} c, 2) &= \frac{1}{P(\mathbf{x})} \times p_{c \xrightarrow{c} c} \times \alpha(c, 1) \times \beta(c, 2) \\
 &= \frac{1}{0.084} \times 0.6 \times 0.600 \times 0.130 = 0.0556
 \end{aligned}$$

# Summing Arc Counts and Reestimation

	1	2	3	4	$c(S \xrightarrow{x} S')$	$p_{S \xrightarrow{x} S'}^{\text{MLE}}$
$c \xrightarrow{C} c$	0.878	0.556	0.000	0.000	1.434	0.523
$c \xrightarrow{W} c$	0.000	0.000	0.264	0.195	0.459	0.167
$c \xrightarrow{C} w$	0.122	0.321	0.000	0.000	0.444	0.162
$c \xrightarrow{W} w$	0.000	0.000	0.308	0.098	0.405	0.148
$w \xrightarrow{C} w$	0.000	0.107	0.000	0.000	0.107	0.085
$w \xrightarrow{W} w$	0.000	0.000	0.400	0.606	1.006	0.800
$w \xrightarrow{C} c$	0.000	0.015	0.000	0.000	0.015	0.012
$w \xrightarrow{W} c$	0.000	0.000	0.029	0.101	0.130	0.103

$$\sum_{x, S'} c(c \xrightarrow{x} S') = 2.742$$

$$\sum_{x, S'} c(w \xrightarrow{x} S') = 1.258$$

# Summing Arc Counts and Reestimation

	1	2	3	4	$c(S \xrightarrow{x} S')$	$p_{S \xrightarrow{x} S'}^{\text{MLE}}$
$c \xrightarrow{C} c$	0.878	0.556	0.000	0.000	1.434	0.523
$c \xrightarrow{W} c$	0.000	0.000	0.264	0.195	0.459	0.167
...	...	...	...	...	...	...
...	...	...	...	...	...	...

$$\sum_{x, S'} c(c \xrightarrow{x} S') = 2.742$$

$$\sum_{x, S'} c(w \xrightarrow{x} S') = 1.258$$

$$c(c \xrightarrow{C} c) = \sum_{t=1}^T c(c \xrightarrow{C} c, t)$$

$$= 0.878 + 0.556 + 0.000 + 0.000 = 1.434$$

$$p_{c \xrightarrow{C} c}^{\text{MLE}} = \frac{c(c \xrightarrow{C} c)}{\sum_{x, S'} c(c \xrightarrow{x} S')} = \frac{1.434}{2.742} = 0.523$$

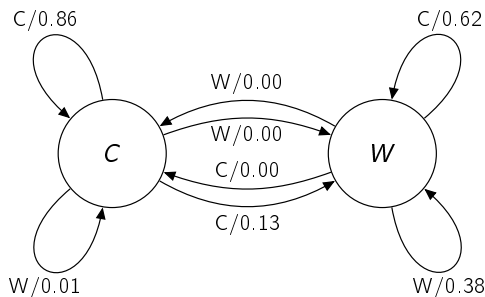
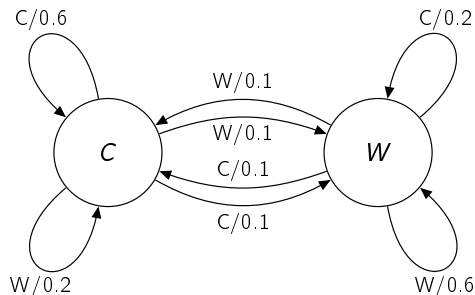
# Slide for Quiet Contemplation

# Another Example

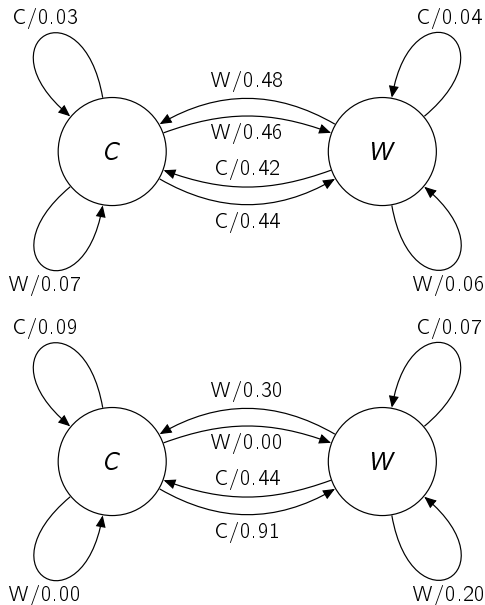
- Same initial HMM.
- Training data: instead of one sequence, many.
  - Each sequence is 26 samples  $\Leftrightarrow$  1 year.

C	W	C	C	C	C	W	C	C	C	C	C	C	W	W	C	W	C	W	W	C	W	C	W	C	C
C	C	C	C	C	C	C	C	C	C	C	C	W	C	C	C	W	W	C	C	W	W	C	W	C	W
C	C	C	C	C	C	C	C	C	C	C	C	C	W	C	W	C	C	W	W	C	W	W	W	C	W
C	C	C	C	C	C	C	C	C	C	C	W	C	W	W	W	C	C	C	C	W	C	C	W	C	C
C	C	C	C	C	C	C	C	C	C	W	C	C	W	W	C	W	C	C	C	W	C	W	C	W	C
C	C	C	C	C	C	W	C	C	C	C	C	W	C	C	C	W	C	W	C	W	C	C	W	C	W
C	C	C	C	C	C	C	C	C	C	C	C	C	W	C	C	C	W	W	C	C	C	W	C	W	C

# Before and After



# Another Starting Point



# Recap: The Forward-Backward Algorithm

- Also called *Baum-Welch* algorithm.
- Instance of EM algorithm.
  - Uses dynamic programming to efficiently sum over ...
  - Exponential number of hidden state sequences.
  - Don't explicitly compute posterior of every  $\mathbf{h}$ .
  - Compute posteriors of counts needed in M step.
- What is time complexity?
- Finds local optimum for parameters in likelihood.
  - Ending point depends on starting point.

# Recap

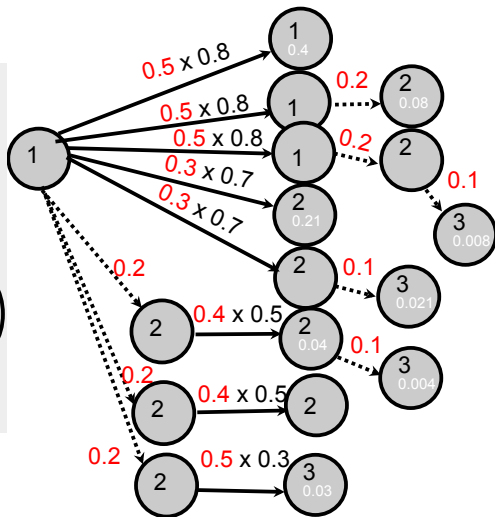
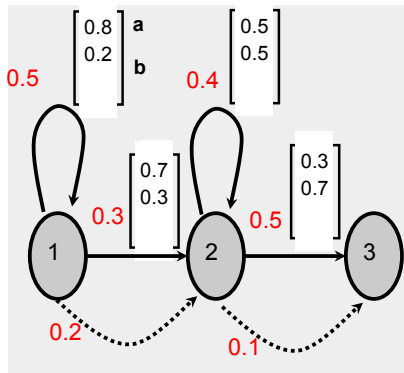
- Given observed, *e.g.*,  $\mathbf{x} = a,a,b,b \dots$ 
  - Find total likelihood  $P(\mathbf{x})$ .
- Need to sum likelihood over all hidden sequences:

$$P(\mathbf{x}) = \sum_{\mathbf{h}} P(\mathbf{h}, \mathbf{x})$$

- The obvious way is to enumerate all state sequences that produce  $\mathbf{x}$
- Computation is exponential in the length of the sequence

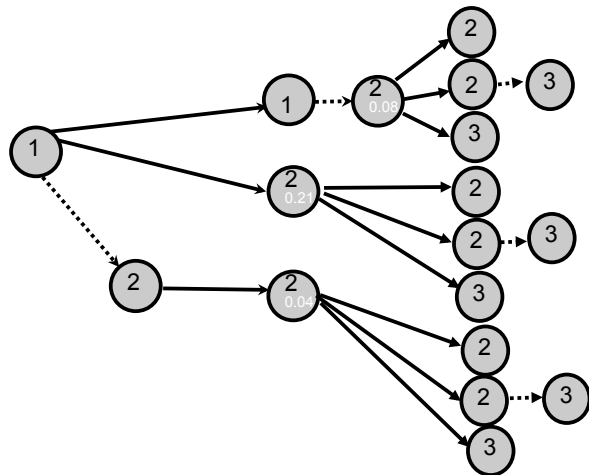
# Examples

Enumerate all possible ways of producing observation **a** starting from state 1



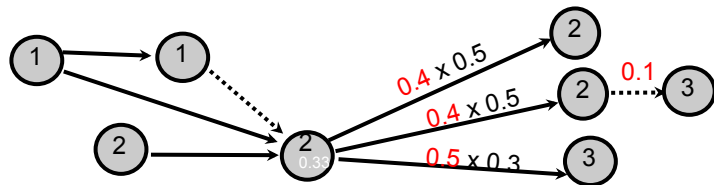
# Examples (contd.)

Enumerate ways of producing observation **aa** for all paths from state **2** after seeing the first observation **a**



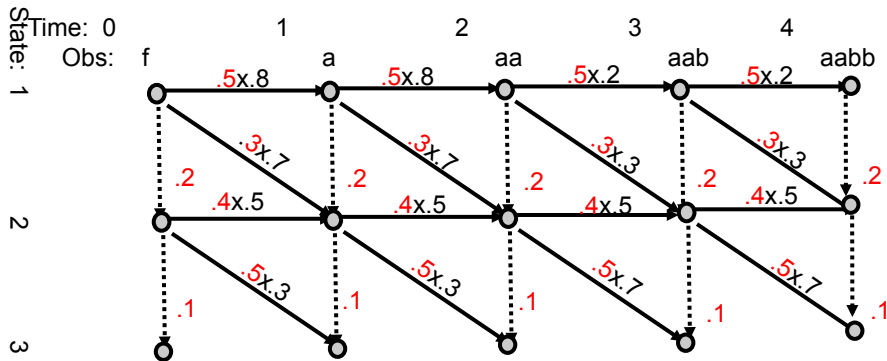
# Examples (contd.)

Save some computation using the Markov property by combining paths



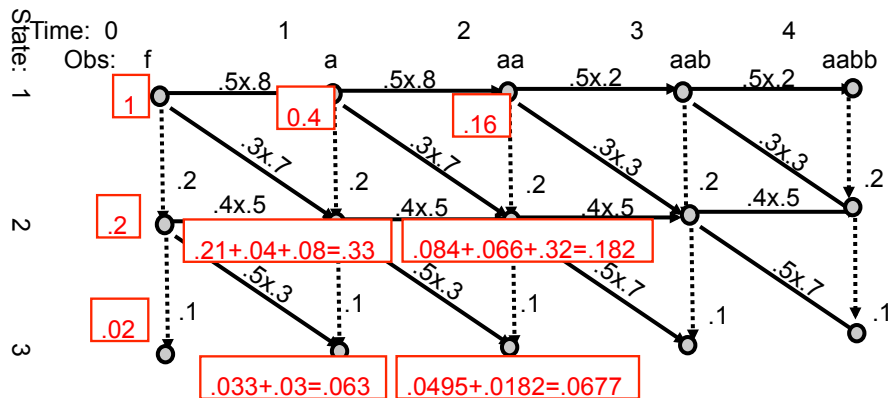
# Examples (contd.)

State transition diagram where each state transition is represented exactly once



# Examples (contd.)

Now let's accumulate the scores ( $\alpha$ )



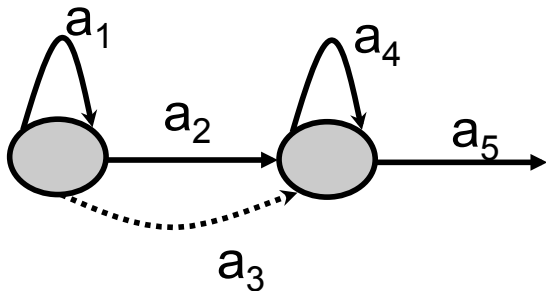
# Parameter Estimation: Examples (contd.)

Estimate the parameters (transition and output probabilities) such that the probability of the output sequence is maximized.

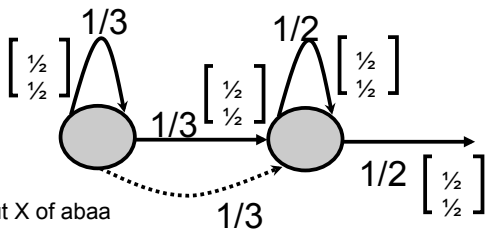
- Start with some initial values for the parameters
- Compute the probability of each path
- Assign fractional path counts to each transition along the paths proportional to these probabilities
- Reestimate parameter values
- Iterate till convergence

## Examples (contd.)



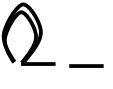

Consider this model, estimate the transition and output probabilities for the sequence: a, b, a, a



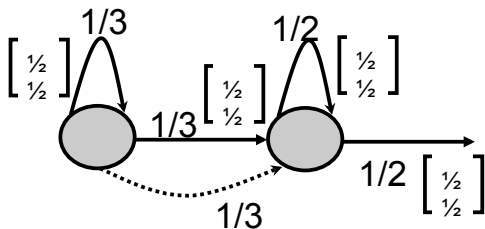
# Examples (contd.)




7 paths corresponding to an output  $X$  of abaa


1.   $p(X, \text{path1}) = 1/3 \times 1/2 \times 1/3 \times 1/2 \times 1/3 \times 1/2 \times 1/3 \times 1/2 \times 1/2 = .000385$
2.   $p(X, \text{path2}) = 1/3 \times 1/2 \times 1/3 \times 1/2 \times 1/3 \times 1/2 \times 1/2 \times 1/2 \times 1/2 = .000578$
3.   $p(X, \text{path3}) = 1/3 \times 1/2 \times 1/3 \times 1/2 \times 1/3 \times 1/2 \times 1/2 \times 1/2 = .001157$
4.   $p(X, \text{path4}) = 1/3 \times 1/2 \times 1/3 \times 1/2 \times 1/2 \times 1/2 \times 1/2 \times 1/2 \times 1/2 = .000868$


# Examples (contd.)



7 paths:

5.   $\text{pr}(X, \text{path5}) = 1/3 \times 1/2 \times 1/3 \times 1/2 \times 1/2 \times 1/2 \times 1/2 \times 1/2 = .001736$

6.   $\text{pr}(X, \text{path6}) = 1/3 \times 1/2 \times 1/2 \times 1/2 \times 1/2 \times 1/2 \times 1/2 \times 1/2 = .001302$

7.   $\text{pr}(X, \text{path7}) = 1/3 \times 1/2 \times 1/2 \times 1/2 \times 1/2 \times 1/2 \times 1/2 \times 1/2 = .002604$

$$P(X) = \sum_i p(X, \text{path}_i) = .008632$$

# Examples (contd.)

## Fractional counts

- Posterior probability of each path:

$$C_i = p(X, \text{path}_i) / P(X)$$

$$C_1 = 0.045, C_2 = 0.067, C_3 = 0.134,$$

$$C_4 = 0.100, C_5 = 0.201, C_6 = 0.150, C_7 = 0.301$$

$$C_{a1} = 3C_1 + 2C_2 + 2C_3 + C_4 + C_5 = 0.838$$

$$C_{a2} = C_3 + C_5 + C_7 = 0.637$$

$$C_{a3} = C_1 + C_2 + C_4 + C_6 = 0.363$$

- Normalize to get new estimates:

$$a_1 = 0.46, a_2 = 0.34, a_3 = 0.20$$

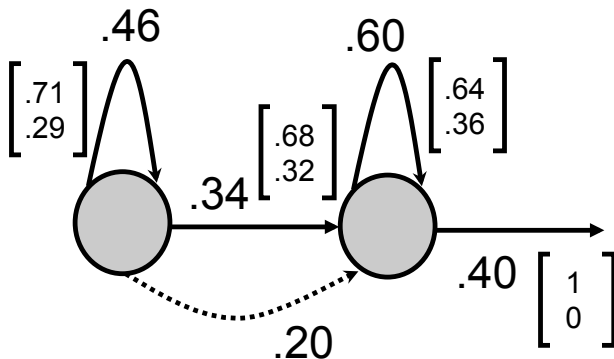
$$C_{a1, 'a'} = 2C_1 + C_2 + C_3 + C_4 + C_5 = 0.592$$

$$C_{a1, 'b'} = C_1 + C_2 + C_3 = 0.246$$

$$p_{a1, 'a'} = 0.71, p_{a1, 'b'} = 0.29$$

# Examples (contd.)

## New Parameters



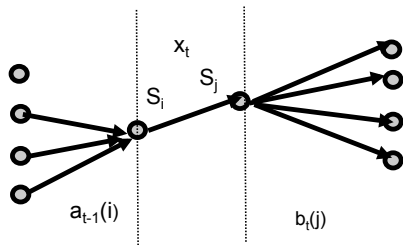
# Examples (contd.)

Iterate till convergence

Step	$P(X)$
1	0.008632
2	0.02438
3	0.02508
100	0.03125004
600	0.037037037 converged

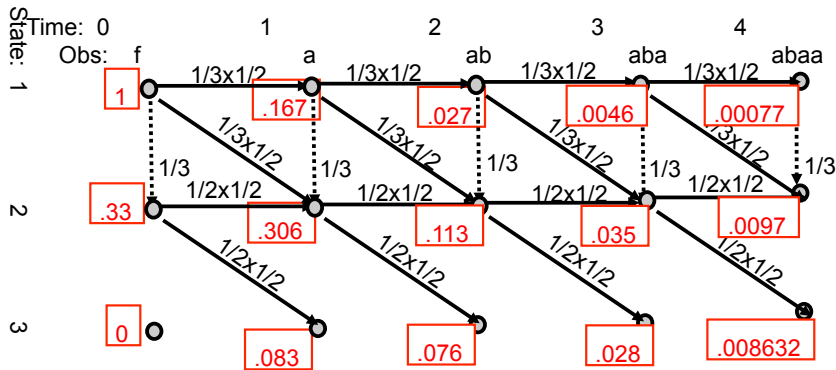
# Examples (contd.)

- Forward-Backward algorithm improves on this enumerative algorithm
- Instead of computing path counts, we compute counts for each transition in the trellis
- Computations are now reduced to linear!



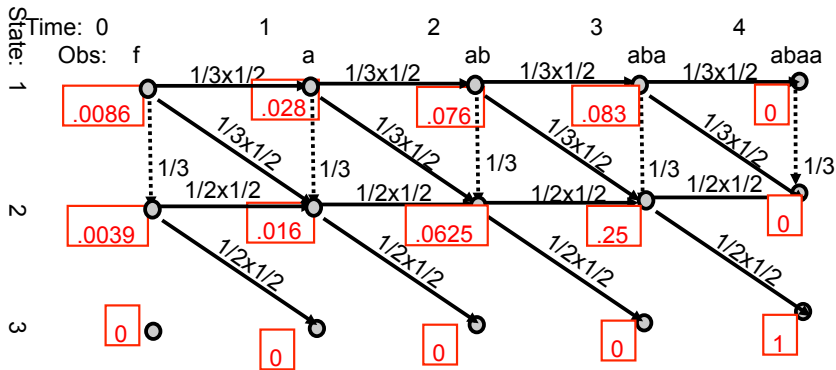
# Examples (contd.)

$\alpha$  computation



## Examples (contd.)

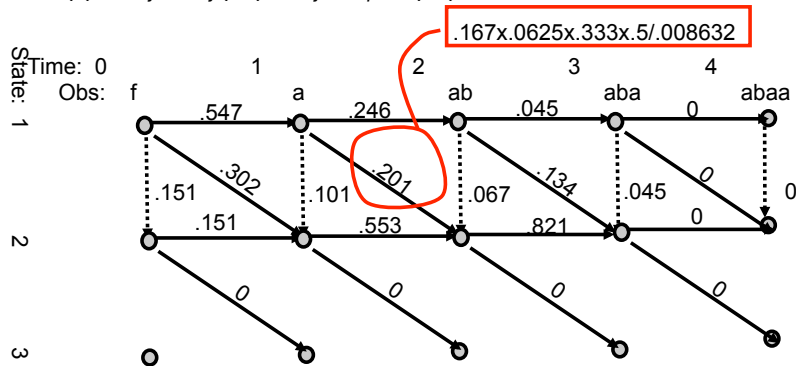
## $\beta$ computation



# Examples (contd.)

How do we use  $\alpha$  and  $\beta$  in computation of fractional counts?

$$\alpha_{t-1}(i) * a_{ij} * b_{ij}(x_t) * \beta_j / p_{\text{path}}(X)$$



$$C_{a1} = 0.547 + 0.246 + 0.045; C_{a2} = 0.302 + 0.201 + 0.134;$$

$$C_{a3} = 0.151 + 0.101 + 0.067 + 0.045$$

# Points to remember

- Re-estimation converges to a local maximum
- Final solution depends on your starting point
- Speed of convergence depends on the starting point

# Where Are We?

- 1 Computing the Best Path
- 2 Computing the Likelihood of Observations
- 3 Estimating Model Parameters
- 4 Discussion**

# HMM's and ASR

- Old paradigm: DTW.

$$w^* = \arg \min_{w \in \text{vocab}} \text{distance}(A'_{\text{test}}, A'_w)$$

- New paradigm: Probabilities.

$$w^* = \arg \max_{w \in \text{vocab}} P(A'_{\text{test}} | w)$$

- Vector quantization:  $A'_{\text{test}} \Rightarrow \mathbf{x}_{\text{test}}$ .
  - Convert from sequence of 40d feature vectors ...
  - To sequence of values from discrete alphabet.

# The Basic Idea

- For each word  $w$ , build HMM modeling  $P(\mathbf{x}|w) = P_w(\mathbf{x})$ .
- Training phase.
  - For each  $w$ , pick HMM topology, initial parameters.
  - Take all instances of  $w$  in training data.
  - Run Forward-Backward on data to update parameters.
- Testing: the Forward algorithm.

$$w^* = \arg \max_{w \in \text{vocab}} P_w(\mathbf{x}_{\text{test}})$$

- Alignment: the Viterbi algorithm.
  - When each sound begins and ends.

# Recap: Discrete HMM's

- HMM's are powerful tool for making probabilistic models ...
  - Of discrete sequences.
- Three key algorithms for HMM's:
  - The Viterbi algorithm.
  - The Forward algorithm.
  - The Forward-Backward algorithm.
- Each algorithm has important role in ASR.
- Can do ASR within probabilistic paradigm ...
  - Using just discrete HMM's and vector quantization.

## Part III

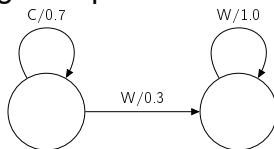
# Continuous Hidden Markov Models

# Going from Discrete to Continuous Outputs

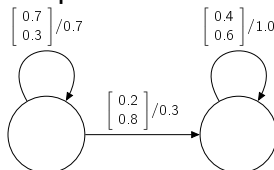
- What we have: a way to assign likelihoods ...
  - To discrete sequences, *e.g.*, C, W, R, C, ...
- What we want: a way to assign likelihoods ...
  - To sequences of 40d (or so) feature vectors.

# Variants of Discrete HMM's

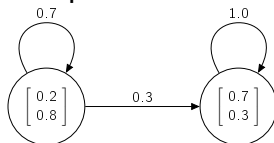
- Our convention: single output on each arc.



- Another convention: output *distribution* on each arc.



- (Another convention: output *distribution* on each state.)



# Moving to Continuous Outputs

- Idea: replace discrete output distribution . . .
  - With continuous output distribution.
- What's our favorite continuous distribution?
  - Gaussian mixture models.

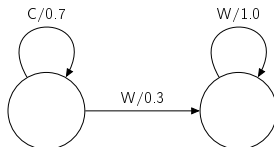
# Where Are We?

1 The Basics

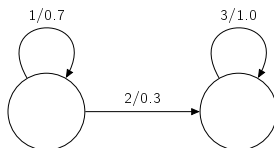
2 Discussion

# Moving to Continuous Outputs

- Discrete HMM's.
  - Finite vocabulary of outputs.
  - Each arc labeled with single output  $x$ .



- Continuous HMM's.
  - Finite number of GMM's:  $g = 1, \dots, G$ .
  - Each arc labeled with single GMM identity  $g$ .



# What Are The Parameters?

- Assume single start state as before.
- Old: one parameter for each arc:  $p_{S \rightarrow S'}^g$ .
  - Identify arc by source  $S$ , destination  $S'$ , and GMM  $g$ .
  - Probs of arcs leaving same state must sum to 1:

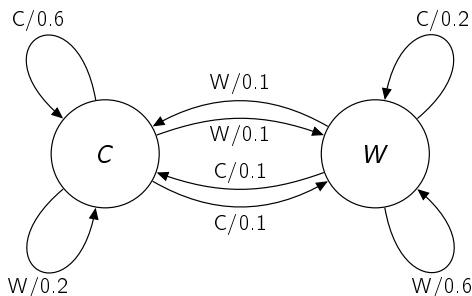
$$\sum_{g, S'} p_{S \rightarrow S'}^g = 1 \quad \text{for all } S$$

- New: GMM parameters for  $g = 1, \dots, G$ :
  - $p_{g,j}$ ,  $\mu_{g,j}$ ,  $\Sigma_{g,j}$ .

$$P_g(x) = \sum_j p_{g,j} \frac{1}{(2\pi)^{d/2} |\Sigma_{g,j}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x} - \mu_{g,j})^T \Sigma_{g,j}^{-1} (\mathbf{x} - \mu_{g,j})}$$

# Computing the Likelihood of a Path

- Multiply arc and output probabilities along path.
- Discrete HMM:
  - Arc probabilities:  $p_{S \rightarrow S'}$ .
  - Output probability 1 if output of arc matches ...
  - And 0 otherwise (*i.e.*, path is disallowed).
- *e.g.*, consider  $\mathbf{x} = C, C, W, W$ .



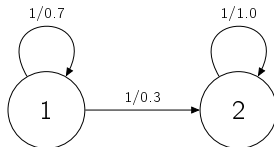
# Computing the Likelihood of a Path

- Multiply arc and output probabilities along path.
- Continuous HMM:
  - Arc probabilities:  $p_{S \xrightarrow{g} S'}$ .
  - Every arc matches any output.
  - Output probability is GMM probability.

$$P_g(x) = \sum_j p_{g,j} \frac{1}{(2\pi)^{d/2} |\Sigma_{g,j}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x} - \mu_{g,j})^T \Sigma_{g,j}^{-1} (\mathbf{x} - \mu_{g,j})}$$

# Example: Computing Path Likelihood

- Single 1d GMM w/ single component:  $\mu_{1,1} = 0$ ,  $\sigma_{1,1}^2 = 1$ .



- Observed:  $\mathbf{x} = 0.3, -0.1$ ; state sequence:  $\mathbf{h} = 1, 1, 2$ .

$$\begin{aligned} P(\mathbf{x}) &= p_{1 \rightarrow 1} \times \frac{1}{\sqrt{2\pi}\sigma_{1,1}} e^{-\frac{(0.3-\mu_{1,1})^2}{2\sigma_{1,1}^2}} \times \\ &\quad p_{1 \rightarrow 2} \times \frac{1}{\sqrt{2\pi}\sigma_{1,1}} e^{-\frac{(-0.1-\mu_{1,1})^2}{2\sigma_{1,1}^2}} \\ &= 0.7 \times 0.381 \times 0.3 \times 0.397 = 0.0318 \end{aligned}$$

# The Three Key Algorithms

- The main change:
  - Whenever see arc probability  $p_{S \xrightarrow{x} S'}$  . . .
  - Replace with arc probability times output probability:

$$p_{S \xrightarrow{g} S'} \times P_g(x)$$

- The other change: Forward-Backward.
  - Need to also reestimate GMM parameters.

# Example: The Forward Algorithm

- $\alpha(S, 0) = 1$  for  $S = S_0$ , 0 otherwise.
- For  $t = 1, \dots, T$ :
  - For each state  $S$ :

$$\alpha(S, t) = \sum_{S' \xrightarrow{g} S} p_{S' \xrightarrow{g} S} \times P_g(x_t) \times \alpha(S', t-1)$$

- The end.

$$P(\mathbf{x}) = \sum_{\mathbf{h}} P(\mathbf{h}, \mathbf{x}) = \sum_S \alpha(S, T)$$

# The Forward-Backward Algorithm

- Compute posterior count of each arc at time  $t$  as before.

$$c(S \xrightarrow{g} S', t) = \frac{1}{P(\mathbf{x})} \times p_{S \xrightarrow{g} S'} \times P_g(x_t) \times \alpha(S, t-1) \times \beta(S', t)$$

- Use to get total counts of each arc as before ...

$$c(S \xrightarrow{x} S') = \sum_{t=1}^T c(S \xrightarrow{x} S', t) \quad p_{S \xrightarrow{x} S'}^{\text{MLE}} = \frac{c(S \xrightarrow{x} S')}{\sum_{x, S'} c(S \xrightarrow{x} S')}$$

- But also use to estimate GMM parameters.
  - Send  $c(S \xrightarrow{g} S', t)$  counts for point  $x_t$  ...
  - To estimate GMM  $g$ .

# Where Are We?

1 The Basics

2 Discussion

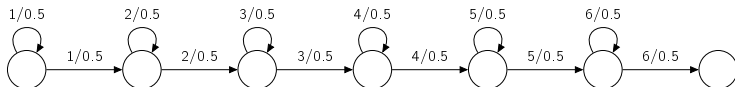
# An HMM/GMM Recognizer

- For each word  $w$ , build HMM modeling  $P(\mathbf{x}|w) = P_w(\mathbf{x})$ .
- Training phase.
  - For each  $w$ , pick HMM topology, initial parameters, ...
  - Number of components in each GMM.
  - Take all instances of  $w$  in training data.
  - Run Forward-Backward on data to update parameters.
- Testing: the Forward algorithm.

$$w^* = \arg \max_{w \in \text{vocab}} P_w(\mathbf{x}_{\text{test}})$$

# What HMM Topology, Initial Parameters?

- A standard topology (three states per phoneme):



- How many Gaussians per mixture?
- Set all means to 0; variances to 1 (*flat start*).
- That's everything!

# HMM/GMM vs. DTW

- Old paradigm: DTW.

$$w^* = \arg \min_{w \in \text{vocab}} \text{distance}(A'_{\text{test}}, A'_w)$$

- New paradigm: Probabilities.

$$w^* = \arg \max_{w \in \text{vocab}} P(A'_{\text{test}} | w)$$

- In fact, can design HMM such that

$$\text{distance}(A'_{\text{test}}, A'_w) \approx -\log P(A'_{\text{test}} | w)$$

- See Holmes, Sec. 9.13, p. 155.

# The More Things Change ...

DTW	HMM
template	HMM
frame in template	state in HMM
DTW alignment	HMM path
local path cost	transition (log)prob
frame distance	output (log)prob
DTW search	Viterbi algorithm

# What Have We Gained?

- Principles!
  - Probability theory; maximum likelihood estimation.
  - Can choose path scores and parameter values ...
  - In non-arbitrary manner.
  - Less ways to screw up!
- Scalability.
  - Can extend HMM/GMM framework to ...
  - Lots of data; continuous speech; large vocab; etc.
- Generalization.
  - HMM can assign high prob to sample ...
  - Even if sample not close to any one training example.

# The Markov Assumption

- Everything need to know about past ...
  - Is encoded in identity of state.
  - *i.e.*, conditional independence of future and past.
- What information do we encode in state?
  - What information don't we encode in state?
  - Issue: the more states, the more parameters.
  - *e.g.*, the weather.
- Solutions.
  - More states.
  - Condition on more stuff, *e.g.*, graphical models.

# Recap: HMM's

- Together with GMM's, good way to model likelihood . . .
  - Of sequences of 40d acoustic feature vectors.
- Use *state* to capture information about past.
  - Lets you model how data evolves over time.
- Not nearly as *ad hoc* as dynamic time warping.
- Need three basic algorithms for ASR.
  - Viterbi, Forward, Forward-Backward.
  - All three are efficient: dynamic programming.
- Know enough to build basic GMM/HMM recognizer.

# Part IV

## Epilogue

# What's Next

- Lab 2: Build simple HMM/GMM system.
  - Training and decoding.
- Lecture 5: Language modeling.
  - Moving from isolated to continuous word ASR.
- Lecture 6: Pronunciation modeling.
  - Moving from small to large vocabulary ASR.