11-751/ 18-781
Speech Recognition and Understanding

# Hidden Markov Models

**Florian Metze**

September 16th & 23rd, 2013

Carnegie Mellon

## Agenda

- Problems with Pattern Matching
- Markov Models
  - Hidden Markov Models
  - Topologies, some properties, terminology
- Three Main Problems of Hidden Markov Models
  - The Evaluation Problem - the **Forward** Algorithm
  - The Decoding Problem - the **Viterbi** Algorithm
  - The Learning/Optimization Problem - the **Forward-Backward** Algorithm
- Hidden Markov Models in Speech Recognition
  - Overview of Hidden Markov Models Training
  - Using (hand-)labeled Data
  - K-Means
  - Training HMMs with Viterbi
- Components of an HMM-Recognizer

## Problems with Pattern Matching

- Need endpoint detection
- Need collection of reference patterns (inconvenient for user)
- Works only well for speaker-dependent recognition (variations)
- High computational effort (esp. for large vocabularies), as proportional to vocabulary size
- **Large vocabulary also means: need huge amount of training data**
  - Difficult to train suitable references (or sets of references)
  - Poor performance when the environment changes
  - Unsuitable where speaker is unknown and no training is feasible
- **Unsuitable for continuous speech (combinatorial explosion of possible patterns, co-articulation)**
- **Impossible to recognize untrained words**
- Difficult to train/ recognize sub-word units

## What we‘re looking for

- We would like to work with speech units shorter than words
  ⇒ each subword unit occurs often, training is easier, less data
- We want to recognize speech from any speaker, without prior training
  ⇒ store "speaker-independent" reference (examples from many speakers)
- We want to recognize **continuous rather than isolated speech**
  ⇒ handle coarticulation effects, handle sequences of words
- We want to recognize words that have not been trained
  ⇒ train subword units and compose any word out of these (vocabulary independence)
- We would prefer a sound mathematical foundation

## Speech Production as Stochastic Process

- We can regard words/ phonemes as **states of a speech production process**
  - The same word/ phoneme/ sound sounds different every time it is uttered
  - In a given state we can observe different acoustic sounds
  - Not all sounds are possible/ likely in every state
  - We say: in a given state the speech process **"emits" sounds** according to some probability distribution/ density
- The production process can make **transitions from one state into another**
  - Not all transitions are possible, transitions have different probabilities
  - When we specify the probabilities for sound-emissions (emission probabilities) and for the state transitions, we call this a **model.**

## Markov Models

- **Observable states:**

$$1, 2, \ldots, N$$

- **Observed sequence:**

$$q_1, q_2, \ldots, q_t, \ldots, q_T$$

- **First order Markov assumption:**

$$P(q_t = j | q_{t-1} = i, q_{t-2} = k, \ldots) = P(q_t = j | q_{t-1} = i)$$

- **Stationarity:**

$$P(q_t = j | q_{t-1} = i) = P(q_{t+l} = j | q_{t+l-1} = i)$$

## Markov Models

interACT

- State transition matrix $A$ :

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2j} & \cdots & a_{2N} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots & a_{iN} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{Nj} & \cdots & a_{NN} \end{bmatrix}$$

where

$$a_{ij} = P(q_t = j | q_{t-1} = i) \qquad 1 \leq i, j, \leq N$$

- Constraints on $a_{ij}$ :

$$\begin{aligned} a_{ij} &\geq 0, \qquad \forall i, j \\ \sum_{j=1}^{N} a_{ij} &= 1, \qquad \forall i \end{aligned}$$

## Example

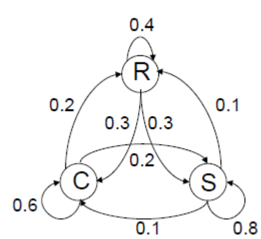
interACT

Formal and graphical representation:

- States:
  1. Rainy ($R$)
  2. Cloudy ($C$)
  3. Sunny ($S$)
- State transition probability matrix:

$$A = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

- Compute the probability of observing $SSRRSCS$ given that today is $S$.



## Example

interACT

Basic conditional probability rule:

$$P(A, B) = P(A|B)P(B)$$

The Markov chain rule:

$$\begin{aligned}
&P(q_1, q_2, \ldots, q_T) \\
&= P(q_T|q_1, q_2, \ldots, q_{T-1})P(q_1, q_2, \ldots, q_{T-1}) \\
&= P(q_T|q_{T-1})P(q_1, q_2, \ldots, q_{T-1}) \\
&= P(q_T|q_{T-1})P(q_{T-1}|q_{T-2})P(q_1, q_2, \ldots, q_{T-2}) \\
&= P(q_T|q_{T-1})P(q_{T-1}|q_{T-2}) \cdots P(q_2|q_1)P(q_1)
\end{aligned}$$

## Example

interACT

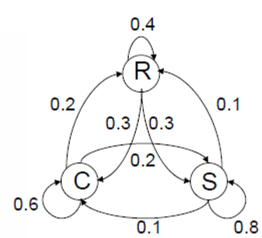
- Observation sequence $O$ :

$$O = (S, S, S, R, R, S, C, S)$$

- Using the chain rule we get:

$$\begin{aligned}
&P(O|model) \\
&= P(S, S, S, R, R, S, C, S|model) \\
&= P(S)P(S|S)P(S|S)P(R|S)P(R|R) \times \\
&\quad P(S|R)P(C|S)P(S|C) \\
&= \pi_3 a_{33} a_{33} a_{31} a_{11} a_{13} a_{32} a_{23} \\
&= (1)(0.8)^2(0.1)(0.4)(0.3)(0.1)(0.2) \\
&= 1.536 \times 10^{-4}
\end{aligned}$$

- The prior probability $\pi_i = P(q_1 = i)$



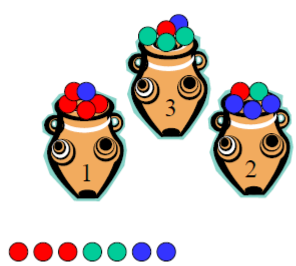

Today is S ⇒ P(S)=1

## Hidden Markov Models

- **Hidden:** States are not observable
- Observations are probabilistic functions of states
- State transitions are also probabilistic

## The „urn and ball“ model

- $n$ urns containing colored balls
- $v$ distinct colors
- Each urn has a (possibly) different distribution of colors




Sequence generation algorithm:

1. (Behind the curtain) Pick initial urn according to some random process
2. (Behind the curtain) Randomly pick ball from the urn. Show it to the audience and put it back
3. (Behind the curtain) Select another urn according to random selection process associated with the urn
4. Repeat step 2 and 3

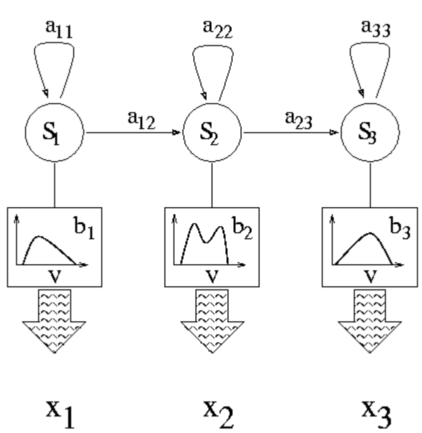
## Formal Definition of HMMs

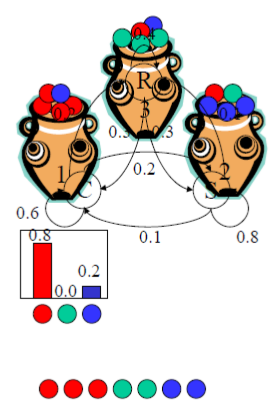
A "Hidden Markov Model" is a 5-tupel consisting of:

- $S$ The set of **States** $S=\{s_1,s_2,\ldots,s_n\}$, n is the number of states
- $\pi$ The **initial probability distribution** , $\pi(s_i) = P(q_1 = s_i)$ probability of $s_i$ being the first state of a sequence
- $A$ *The matrix of **state transition probabilities:*** $1 \le i, j \le n$
  $A=(a_{ij})$ with $a_{ij} = P(q_{t+1}= s_j|q_t = s_i)$ going from state $s_i$ to $s_j$
- $B$ The set of **emission probability distributions/ densities,**
  $B=\{b_1,b_2,\ldots,b_n\}$ where $b_i(x)=P(o_t = x|q_t = s_i)$ is the probability of observing $x$ when the system is in state $s_i$
- $V$ Set of symbols, $v$ is the number of distinct symbols. The observable **feature space** can be discrete: $V=\{x_1,x_2,\ldots,x_v\}$, or continuous $V=\mathbf{R}^d$

## Generating an Observation Sequence

The term "**hidden**" refers to seeing observations and drawing conclusions without knowing the *hidden* sequence of states (urns)

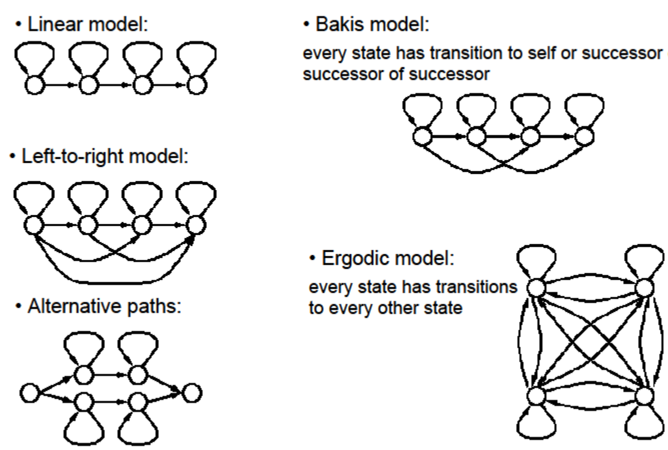


## Some Properties of Hidden Markov Models

- For the initial probabilities we have: $\Sigma_i \, \pi(s_i) = 1$
- Often things are simplified by $\pi(s_1) = 1$, and $\pi(s_{i>1}) = 0$
- Obviously: $\Sigma_j \, a_{ij} = 1$ for all I
- Often: $a_{ij} = 0$ for most $j$ except for a few states
- When $V = \{x_1, x_2, \ldots, x_v\}$ then $b_i$ are discrete probability distributions, the HMMs are called **discrete HMMs**
- When $V = \boldsymbol{R}^d$ then $b_i$ are continuous probability density functions, the HMMs are called **continuous (density) HMMs**

## Some Typical HMM Topologies



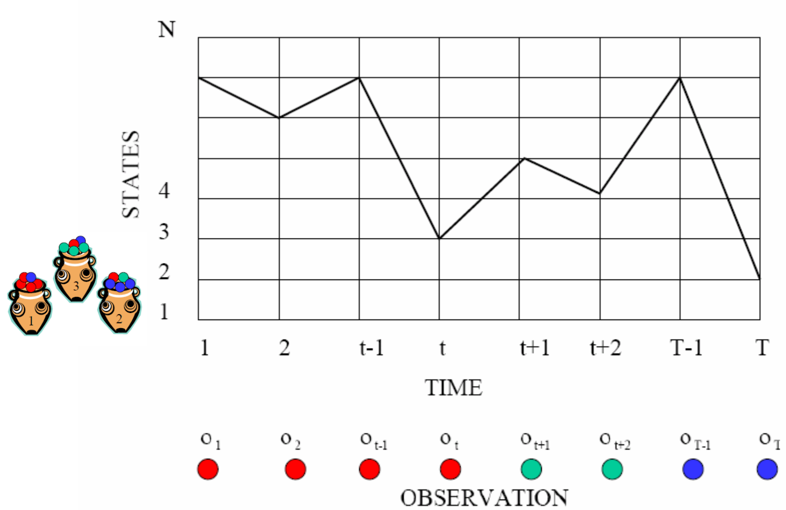

Applications: simulation and analysis of complex stochastic systems (weather, traffic, queues); recognition of dynamic patterns (speech, handwriting, video).

## Some HMM Terminology

The most ambiguously used term is the **"model",** which can be one of:

- The **language model =** combination of all parameters describing probabilities of word sequences
- The ***acoustic model*** **=** combination of all parameters of a recognizer describing all acoustic features
- An ***(acoustic) model*** **=** combination of the parameters that describe acoustic features of a *specific unit of speech*
- A Hidden Markov **Model =** the defined five-tuple
- The **model of a state =** the combination of HMM-parameters that describe the properties of an HMM-state (different states can have the same model)

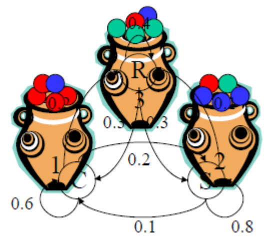
## The „Trellis“



## Typical HMM Questions

interACT

- A magician draws balls from urns behind the curtain, the audience sees the observation sequence $O=(o_1,o_2,\ldots,o_T)$
- Your friend told you about two sets of urns and drawing patterns = models $\lambda_1=(A,B,\pi)$ $\lambda_2=(A,B,\pi)$ the magician usually uses
- Assume you have an efficient algorithm to compute $P(O|\lambda)$




1. Compute $P(O|\lambda)$ for both models, which of the models $\lambda_1$ or $\lambda_2$ was more likely to be used by the magician
2. Given one model, find the “optimal” aka most likely state sequence that would produce the observation
3. Find a new model $\lambda'$ such that $P(O|\lambda') > P(O|\lambda)$

## Three Main Problems of HMMs

interACT

- **The Evaluation Problem:**

  given an HMM $\lambda$ and an observation $O = (x_1, x_2, \ldots, x_T)$

  efficiently compute the probability of the observation: $p(O|\lambda)$

- **The Decoding Problem:**

  given an HMM $\lambda$ and an observation $O = (x_1, x_2, \ldots, x_T)$

  compute the most likely state sequence $s_{q1}, s_{q2}, \ldots, s_{qT}$,

  i.e. $\mathrm{argmax}_{q1,\ldots,qT}\, p(q_1, \ldots, q_T | O, \lambda)$

- **The Learning/ Optimization Problem:**

  given an HMM $\lambda$ and an observation $O = (x_1, x_2, \ldots, x_T)$

  find an HMM $\lambda'$ such that $p(O | \lambda') > p(O | \lambda)$

## The Evaluation Problem

Given an HMM $\lambda$ and an observation $O = (x_1, x_2, \ldots, x_T)$
compute the probability of the observation $p(O|\lambda)$.

Solution to the weaker problem with a fixed state sequence $Q = (s_{q1}, s_{q2}, \ldots, s_{qT})$:

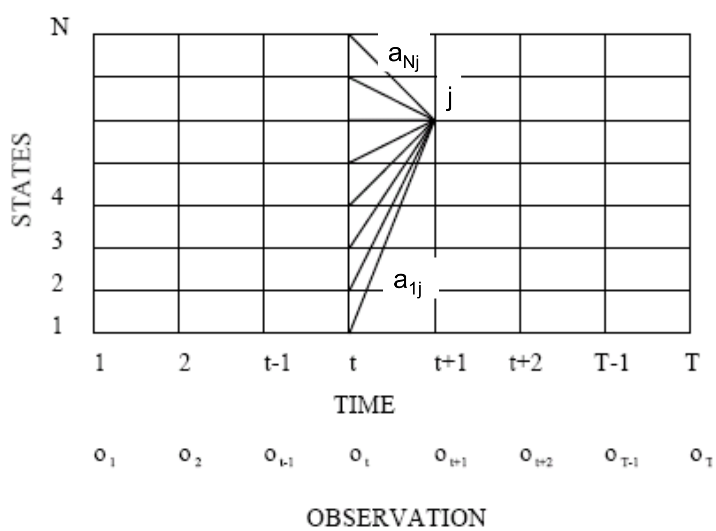
$$
\begin{aligned}
p(O,Q|\lambda) &= p(O|Q,\lambda)\, p(Q|\lambda) \\
&= b_{q1}(x_1)\, b_{q2}(x_2)\, b_{q3}(x_3) \ldots \pi(s_{q1})\, a_{q1q2}\, a_{q2q3} \ldots \\
&= \pi(s_{q1}) b_{q1}(x_1) \prod_{k=1..T-1} a_{qkqk+1}\, b_{qk+1}(x_{k+1})
\end{aligned}
$$

from this we get: enumerate all possible paths (state sequences) and sum probabilities

$$
\begin{aligned}
p(O|\lambda) &= \sum_{all\ Q} p(O,Q|\lambda) \\
&= \sum_{q1,\ldots,qT} \pi(s_{q1}) b_{q1}(x_1) \prod_{k=1..T-1} a_{qkqk+1}\, b_{qk+1}(x_{k+1})
\end{aligned}
$$

**Problem - Complexity:**
$N^T$ possible state sequences and O(T) calculations
⇨ **O($TN^T$)**
⇨ find a more efficient algorithm

## Idea: Forward Algorithm



## The Forward Algorithm

Can we recursively define $p(x_1,x_2,\ldots,x_T \mid \lambda)$?

Obviously: $\quad p(x_1,x_2,\ldots,x_T \mid \lambda)$

$= \sum_{j=1..n} p(x_1,x_2,\ldots,x_T, q_T=j \mid \lambda)$

If we define:

$\alpha_t(j) := p(x_1,x_2,\ldots,x_t, q_t=j \mid \lambda)$ then $p(x_1,x_2,\ldots,x_T \mid \lambda) = \sum_{j=1..n} \alpha_T(j)$

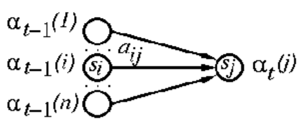
$\alpha_t(j)$ is the probability of observing the partial observation $x_1,x_2,\ldots,x_t$ and then being in state $s_j$

Remember DP: Can we compute $\alpha_t(j)$ out of $\alpha_{t-1}(\ldots)$?

Indeed:

$\boxed{\alpha_t(j) = b_j(x_t) \cdot \sum_{i=1..n} a_{ij}\, \alpha_{t-1}(i)}$

and $\alpha_1(j) = \pi(s_j)\, b_j(x_1)$




## The Forward Procedure

Initialization:

$\alpha_1(j) = \pi(s_j)\, b_j(x_1)$

Induction:

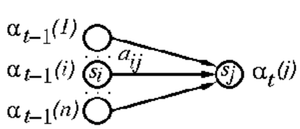
$\alpha_t(j) = b_j(x_t) \cdot \sum_{i=1..n} a_{ij}\, \alpha_{t-1}(i)$

Termination:

$p(x_1,x_2,\ldots,x_T \mid \lambda) = \sum_{j=1..n} \alpha_T(j)$

Complexity:

$O(N^2T)$



## Three Main Problems of HMMs

- **The Evaluation Problem:**
  given an HMM $\lambda$ and an observation $O = (x_1, x_2, ..., x_T)$
  efficiently compute the probability of the observation: $p(O| \lambda)$
- **The Decoding Problem:**
  given an HMM $\lambda$ and an observation $O = (x_1, x_2, ..., x_T)$
  compute the most likely state sequence $s_{q1}, s_{q2}, ..., s_{qT}$,
  i.e. $\mathrm{argmax}_{q1,...,qT}\, p(q_1, .., q_T| O, \lambda)$
- **The Learning/ Optimization Problem:**
  given an HMM $\lambda$ and an observation $O = (x_1, x_2, ..., x_T)$
  find an HMM $\lambda'$ such that $p(O | \lambda') > p(O | \lambda)$

## The Decoding Problem

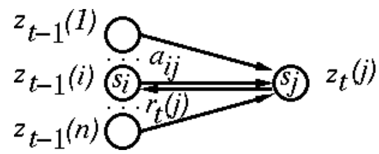
- Given an HMM $\lambda$ and an observation $x_1,x_2,...,x_T$, compute the most likely state sequence $s_{q1},s_{q2},...,s_{qT}$, that has to be traversed to produce the observation. I.e.
  $\mathrm{argmax}_{q1,..,qT} \quad p(q_1,..,q_T|x_1,x_2,...,x_T, \lambda) =$
  $\mathrm{argmax}_{q1,..,qT} \quad p(q_1,..,q_T , x_1,x_2,...,x_T | \lambda) / p(x_1,x_2,...,x_T | \lambda)$
- For finding the max, the denominator of the last term is of no importance, so all we need is:
  $\mathrm{argmax}_{q1,..,qT} \quad p(q_1,..,q_T , x_1,x_2,...,x_T | \lambda)$
- Remember **forward** algorithm. Can we compute
  $\max_{q1,..,qt} p(q_1,..,q_t , x_1,x_2,...,x_t | \lambda)$ out of $p(q_1,..,q_{t-1} , x_1,x_2,...,x_{t-1} | \lambda)$ ?

## The Viterbi Algorithm

Let's define $z_t(j) := \max\{ p(q_1,..,q_t, x_1,x_2,...,x_t \mid \lambda) \mid q_1,.., q_t=j \}$

then $z_t(j) = \max_i(z_{t-1}(i)\, a_{ij})\, b_j(x_t)$

and $z_1(j) = \pi(s_j)\, b_j(x_1)$




To retrieve the optimal state sequence, we have to remember for every state its optimal predecessor $r_t(j) = \text{argmax}_i(z_{t-1}(\text{i})\, a_{ij})$

**Note:** the Viterbi algorithm is very similar to the DTW algorithm

- Viterbi multiplies probabilities, DTW adds up distances.
- Viterbi uses state transition probabilities, DTW weighted transition patterns
- When we logarithmize the probabilities in Viterbi, we get "scores" that can be added like distances

## Some Notes about the Viterbi Algorithm

Compare Viterbi to the Forward algorithm:

Viterbi $z_t(j) = b_j(x_t) * \max_{i=1..n} a_{ij}\, z_{t-1}(i)$

Forward $\alpha_t(j) = b_j(x_t) * \sum_{i=1..n} a_{ij}\, \alpha_{t-1}(i)$

$z_t(j) = b_j(x_t) * \max_i (z_{t-1}(i)\, a_{ij})$

$\alpha_t(j) = b_j(x_t) * \sum_{i=1..n} \alpha_{t-1}(i)\, a_{ij}$

---

Both, **Viterbi** and the **Forward** algorithm are multiplying many (often $T$>1000) small probabilities (density values) ⇒ floating point underflow.

- Solution 1: apply scaling factors:

$$\max_{i=1..n} a_{ij}\, \boldsymbol{f} \cdot z_{t-1}(i) = \boldsymbol{f} \cdot \max_{i=1..n} a_{ij}\, z_{t-1}(i)$$

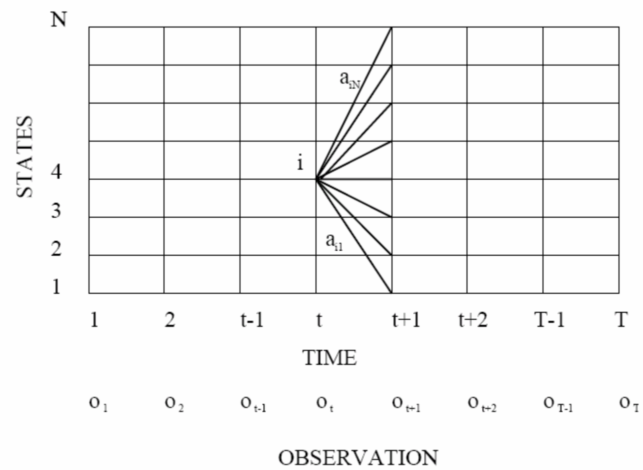
$$\sum_{i=1..n} a_{ij}\, \boldsymbol{f} \cdot \alpha_{t-1}(i) = \boldsymbol{f} \cdot \sum_{i=1..n} a_{ij}\, \alpha_{t-1}(i)$$

- Solution 2: use logarithms: (remember “score” = -log p)

$$\max_{i=1..n} \mathbf{log}\, (a_{ij}\, z_{t-1}(i)) = \mathbf{log}\, \max_{i=1..n} a_{ij}\, z_{t-1}(i)$$

## Idea: the „Backward Algorithm“






## The Backward Algorithm

interACT

Obviously: $p(x_{t+1},x_{t+2},\ldots,x_T \mid \lambda)$
$= \sum_{i=1..n} p(x_{t+1},x_{t+2},\ldots,x_T, q_t=i \mid \lambda)$

Define backward variable $\beta_t(i) := p(x_{t+1},x_{t+2},\ldots,x_T, q_t=i \mid \lambda)$

$\beta_t(i)$ is the probability of observing the partial observation $x_{t+1},x_{t+2},\ldots,x_T$ and being in state $s_i$

Induction

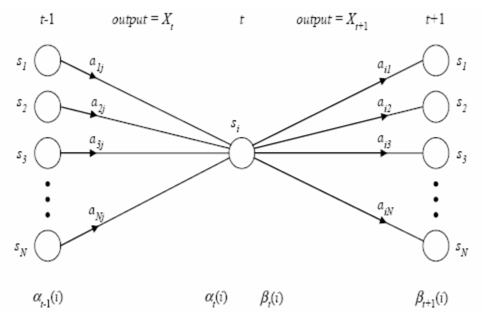
- Initialization: $\beta_T(i) = 1/N$
- Induction: $\beta_t(i) = \sum_{j=1..n} a_{ij}\, b_j(x_{t+1})\, \beta_{t+1}(j),\; t= T-1, \ldots 1$
- Termination: $p(x_{t+1},x_{t+2},\ldots,x_T \mid \lambda) = \sum_{j=1..n} \beta_T(j)$

Complexity: $O(N^2T)$

## Relationship of Adjacient α and β

interACT

Compute α recursively from left to right, and β from right to left




By convention, alpha's contain the *b(t)* (emission probability at *t*)

## Three Main Problems of HMMs

interACT

- **The Evaluation Problem:**
  given an HMM λ and an observation $O = (x_1, x_2, ..., x_T)$
  efficiently compute the probability of the observation: $p(O| \lambda)$
- **The Decoding Problem:**
  given an HMM λ and an observation $O = (x_1, x_2, ..., x_T)$
  compute the most likely state sequence $s_{q1}, s_{q2}, ..., s_{qT}$,
  i.e. $\mathrm{argmax}_{q1,...,qT}\, p(q_1, .., q_T| O, \lambda)$
- **The Learning/ Optimization Problem:**
  given an HMM λ and an observation $O = (x_1, x_2, ..., x_T)$
  find an HMM λ′ such that $p(O | \lambda') > p(O | \lambda)$

## Optimize HMM $\lambda$

In order to optimize $\lambda$ and retrieve $\lambda'$ we can iteratively refine the HMM parameters $\lambda=(A,B,\pi)$ by maximizing the likelihood $P(X|\lambda)$

- Optimize $\pi$, optimize $b_j$, optimize $a_{ij}$
- No analytical method known, therefore solve by Expectation Maximisation (EM)-algorithm (iteratively maximizing the expectation of likelihood)

<u>To optimize $\pi$, $b_j$</u> we need the probability that the system $\lambda$ is in state $s_j$ at time $t$ when producing the observation $X = x_1,x_2,\ldots,x_T$ .

$P(q_t=i \mid x_1,x_2,\ldots,x_T, \lambda)$ tells us the "weight" with which we need to modify the parameters of the model in order to increase the likelihood of observing $x_t$.

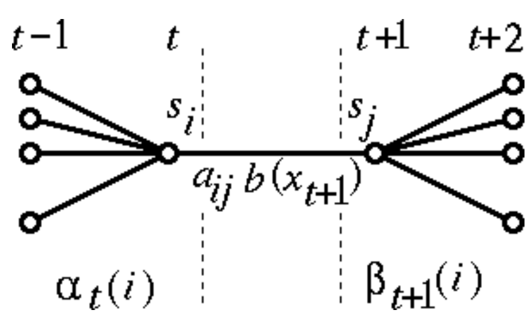
Then we define $\gamma_t(j)$ the probability that system $\lambda$ is in state $s_j$ at time $t$ when producing the observation $X = x_1,x_2,\ldots,x_T$:

$$\gamma_t(j) := \frac{P(q_t=j \mid X, \lambda)}{P(X \mid \lambda)} = \frac{\alpha_t(j) \cdot \beta_t(j)}{\sum_i \alpha_t(i) \cdot \beta_t(i)}$$

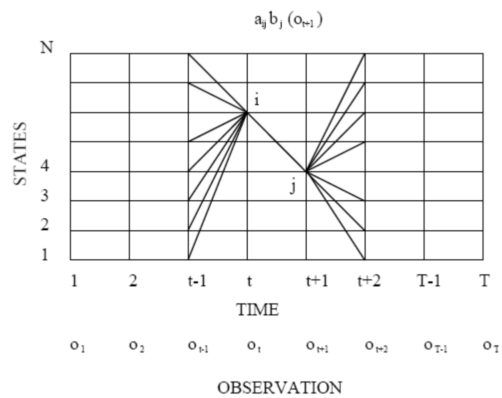
## Optimizing the Transition Probabilities

<u>To optimize $a_{ij}$</u>, we need the probability $\xi_t(i,j)$ of being in state $i$ at time t, transition from state $i$ to $j$, and be in state $j$ at time $t+1$

$$P(q_t=i,q_{t+1}=j|X,\lambda) = \frac{P(q_t=i,q_{t+1}=j, X|\lambda)}{P(X \mid \lambda)} = \frac{\alpha_t(i)\, a_{ij} b_j(x_{t+1})\, \beta_{t+1}(j)}{P(X|\lambda)} =: \xi_t(i,j)$$



## The Baum-Welch Algorithm




## The Baum-Welch Optimization Rules

- Run a forward-backward algorithm, compute the $\alpha_t(i)$, $\beta_t(i)$, $\gamma_t(i)$, and $\xi_t(i,j)$
- $\sum_{t=1..T-1} \gamma_t(i)$ is the expected number of times state *i* is visited
- $\sum_{t=1..T-1} \xi_t(i,j)$ is the expected number of transitions from state *i* to *j*
- $\pi(i)$ is the expected frequency in state *i* at time (t=1) = $\gamma_1(i)$
- $a_{ij}$ is the expected # of transitions from *i* to *j* DIV expected # of transitions from *i*
- $b_i(v_k)$ is the expected # in state *i* and observing symbol k DIV expected # in state *i*

## The Baum-Welch Optimization Rules

interACT

$$\lambda': \begin{cases} \pi'(i) = P(q_1=i \mid X,\lambda) = \dfrac{P(X,q_1=i \mid \lambda)}{P(X \mid \lambda)} = \gamma_1(i) = \dfrac{\alpha_1(i)\,\beta_1(i)}{\sum_{i=1..n} \alpha_1(i)\cdot\beta_1(i)} \\[2ex] a'_{ij} = \dfrac{\sum_{t=1..T-1} P(X, q_t=i, q_{t+1}=j \mid \lambda)}{\sum_{t=1..T-1} P(X, q_t=i \mid \lambda)} = \dfrac{\sum_{t=1..T-1} \xi_t(i,j)}{\sum_{t=1..T-1} \gamma_t(i)} \\[2ex] b'_i(v_k) = \dfrac{\sum_{t=1..T} P(X, q_t = i \mid \lambda)\cdot\delta(x_t,v_k)}{\sum_{t=1..T} P(X, q_t = i \mid \lambda)} = \dfrac{\sum_{t=1..T} \gamma_t(i)\cdot\delta(x_t,v_k)}{\sum_{t=1..T} \gamma_t(i)} \end{cases}$$

For proof that $P(X \mid \lambda') > P(X \mid \lambda)$ see X.D. Huang, Y. Ariki, M.D. Jack: HMMs for Speech Recognition, Edinburgh University Press, 1990.

$$\delta(x,y) = \begin{cases} 1 & \text{if } x=y \\ 0 & \text{else} \end{cases}$$

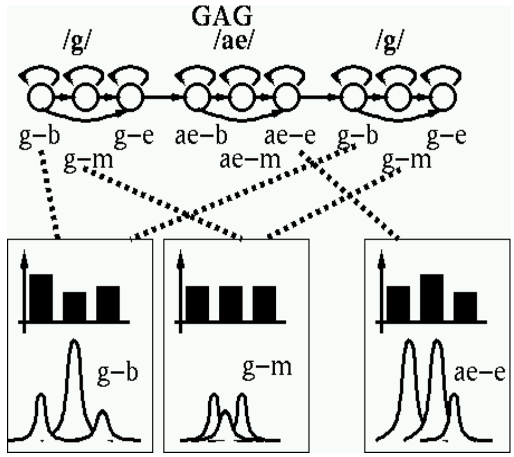
## HMMs and their Recognizers

interACT

- Hidden Markov Models in Speech Recognition
- Overview of Hidden Markov Models Training
    - Using (Hand-)Labeled Data
    - K-Means
    - Training HMMs with Viterbi
- Components of an HMM-Recognizer

## Hidden Markov Models in Speech Recognition

- The states that correspond to the same acoustic phenomenon share the same "acoustic model"
- The training data can be exploited better
- In this example HMM $b_1 = b_7 = b_{g\text{-}b}$
- The parameters of the emission probabilities can be estimated more robustly
- Save computation time (don't evaluate $b(\ldots)$ for every state)

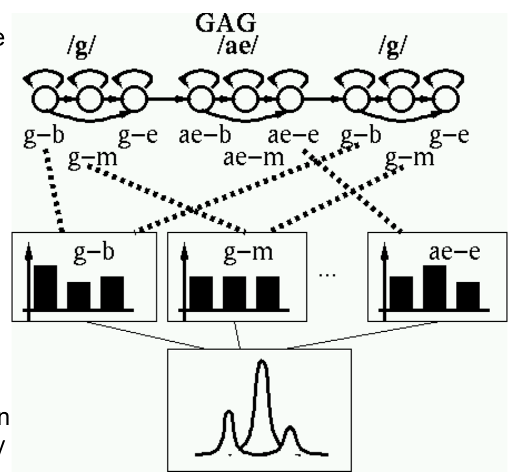



## Hidden Markov Models in Speech Recognition

**Semi-continuous** HMMs do even more

- only one codebook of Gaussians in the system
- every acoustic model shares the same Gaussian codebook
- every acoustic model has its own set of mixture weights
- the Gaussians evaluation can be seen as part of the signal processing, and the HMM works like a discrete HMM

⇒ fewer parameters, but poor resolution of the feature space and thus usually lower recognition accuracy




**Phonetically-tied Semi-continuous** HMMs: one codebook per phone that is shared across polyphones of that phone, i.e. in the example above: one codebook for /g/ and another codebook for /ae/

## Hidden Markov Model Training

A typical HMM training session looks like this:

1. **Initialization**
   - Either initialize all parameters randomly
   - Or use *pre-labeled data* (by hand or other recognizer) and compute initial parameters with e.g. *k-means*
2. **Iterative Optimization/ Training**
   - For all training utterances, run *forward-backward algorithm or Viterbi*
   - Perform *Baum-Welch re-estimation* on HMM parameters (example: *Gaussian Mixtures with the EM-Algorithm)*
   - Continue for a defined number of iterations or until evaluation on development set give no more improvements
3. **Evaluation**
   - Record utterance $X$
   - Run *forward algorithm* with every internal reference HMM
   - Recognize the word $\lambda_w$ with the highest $P(X|\lambda_w)$

## Using (Pre-)Labeled Data

A good initialization of HMM-parameters can be made with labeled data:

**Automatic Labeling (that is probably somehow suboptimal)**

- Use an existing recognizer $\lambda_{\text{old}}$
- Use $\gamma_t(j)$ of its *forward-backward algorithm* and optimize parameters of $\lambda_{\text{new}}$

**Using Hand-Made Labels (that is probably somehow time consuming)**

- For all training utterances, assign an HMM-state $j$ to every speech-frame $t$
- Train the corresponding model as if its $\gamma_t(j)$ was *1.0*

**The Labeling Process**

- Have people (experts) sit down and listen to utterances
- Let them mark boundaries between speech units like words. Phonemes, ...

**Using Partial Labels**

- When labels are not available for HMM-states but for larger speech units ...
  - … then partition the utterances into the labeled segments
- Run forward-backward with randomly/ uniformly initialized parameters on small segments

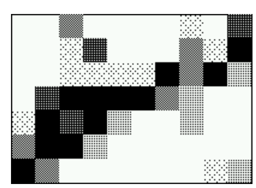
## Training HMMs with Viterbi

- The forward-backward algorithm gives us $P(q_t=j \mid X, \lambda) =: \gamma_t(j)$, the probability that the system $\lambda$ is in state $s_j$ at time $t$ when producing the observation $X = x_1,x_2,\ldots,x_T$.
- In other words: $\gamma_t(j)$ is the weight (0.0 ... 1.0) of the impact of the observation $x_t$ on the training of $b_j$.

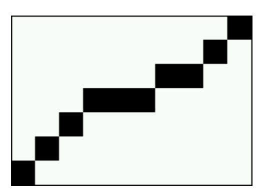
  so we could simply define: $\gamma_t(j) = \begin{cases} 1.0 \text{ if } j = q_t \\ 0.0 \text{ for all other } j \end{cases}$
- Sometimes for a given $t$ the $\gamma_t(j)$ are "spread" over many states, sometimes there are only few (maybe just one) states that have a $\gamma_t(j)$ that is significantly greater than 0.0.
- This leads us to the following idea: the Viterbi-algorithm computes the most likely state sequence $s_{q1},s_{q2},\ldots,s_{qT}$,

## Training HMMs with Viterbi

The forward-backward algorithm produces a $\gamma_t(j)$-Matrix like:

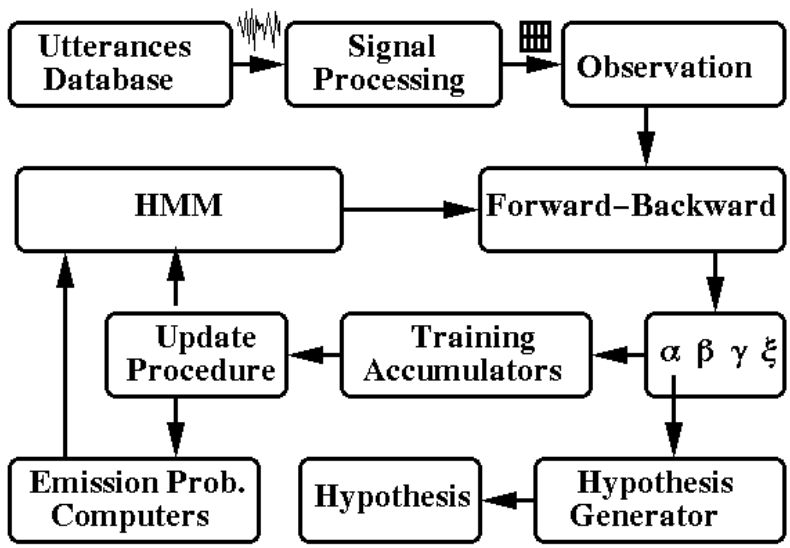


The Viterbi algorithm produces a $\gamma_t(j)$-Matrix like:



Viterbi training is faster than forward backward
BUT: When only little data is available forward-backward performs better

## Components of an HMM-Recognizer




## Components of an HMM Recognizer

- **Signal Processing:** time-representation → frequency-representation, normalization
- **Utterance Database:** training, development, and evaluation data with transcriptions
- **Forward-Backward Algorithm:** forward okay for evaluation, forward-backward for computing $\alpha, \beta, \gamma, \xi$ for training
- **Training Accumulators:** when training more than one utterance replace $\Sigma_{1..T}$ by $\Sigma_u \Sigma_{1..T_u}$
- **Set of HMM Models:** one "five-tuple" for every word/ phoneme/ unit to be recognized
- **Set of Acoustic Models:** different states can share same model, (usually Gaussian mixture densities)
- **Parameter Update Procedure:** model-dependent, e.g. EM algorithm for Gaussians

interACT

# Backup

interACT

## References


- L.R. Rabiner: A Tutorial on HMM and Selected Applications in Speech Recognition, In: [WL], pp 267-296
- X. Huang, A. Acero, H.-W. Hon: Spoken Language Processing, Chapter 8, pp 374-409
- Tapas Kanungo, Uni Maryland, HMM Tutorial Slides (many of his slides have been reused here)
- Andrew Moore slides: similar to the above, see http://www.autonlab.org/tutorials/hmm.html
- J. Bilmes: “What HMMs can do”