Hidden Markov Models for Speech Recognition

Berlin Chen
Department of Computer Science & Information Engineering
National Taiwan Normal University

References:
1. Rabiner and Juang. *Fundamentals of Speech Recognition*. Chapter 6
2. Huang et. al. *Spoken Language Processing*. Chapters 4, 8
Introduction

• Hidden Markov Model (HMM)

**History**
- Published in papers of Baum in late 1960s and early 1970s
- Introduced to speech processing by Baker (CMU) and Jelinek (IBM) in the 1970s

**Assumption**
- Speech signal can be characterized as a parametric random (stochastic) process
- Parameters can be estimated in a precise, well-defined manner

**Three fundamental problems**
- Evaluation of probability (likelihood) of a sequence of observations given a specific HMM
- Adjustment of model parameters so as to best account for observed signals
- Determination of a best sequence of model states
Stochastic Process

• A stochastic process is a mathematical model of a probabilistic experiment that evolves in time and generates a sequence of numeric values
  – Each numeric value in the sequence is modeled by a random variable
  – A stochastic process is just a (finite/infinite) sequence of random variables

• Examples
  (a) the sequence of recorded values of a speech utterance
  (b) the sequence of daily prices of a stock
  (c) the sequence of hourly traffic loads at a node of a communication network
  (d) the sequence of radar measurements of the position of an airplane
Observable Markov Model

- **Observable Markov Model (Markov Chain)**
  - **First-order** Markov chain of $N$ states is a triple $(S, A, \pi)$
    - $S$ is a set of $N$ states
    - $A$ is the $N \times N$ matrix of transition probabilities between states
      \[
      P(s_t=j|s_{t-1}=i, s_{t-2}=k, \ldots) = P(s_t=j|s_{t-1}=i) = A_{ij}
      \]
    - $\pi$ is the vector of initial state probabilities
      \[\pi_j = P(s_1=j)\]
    - The output of the process is the set of states at each instant of time, when each state corresponds to an observable event
      - The output in any given state is not random (**deterministic**)!
    - Too simple to describe the speech signal characteristics

*Fig. 1.* A Markov chain with 5 states (labeled $S_1$ to $S_5$) with selected state transitions.
Observable Markov Model (cont.)

First-order Markov chain of 2 states

Second-order Markov chain of 2 states
Observable Markov Model (cont.)

- Example 1: A 3-state Markov Chain $\lambda$
  
  State 1 generates symbol A **only**,  
  State 2 generates symbol B **only**,  
  and State 3 generates symbol C **only**
  
  \[
  \begin{bmatrix}
  0.6 & 0.3 & 0.1 \\
  0.1 & 0.7 & 0.2 \\
  0.3 & 0.2 & 0.5
  \end{bmatrix}
  \]

  \[
  \pi = \begin{bmatrix}
  0.4 & 0.5 & 0.1
  \end{bmatrix}
  \]

  - Given a sequence of observed symbols $O=$\{CABBCABC\}, the **only** one corresponding state sequence is \{S$_3$S$_1$S$_2$S$_2$S$_3$S$_1$S$_2$S$_3$\}, and the corresponding probability is

  \[
P(O|\lambda) = P(S_3)P(S_1|S_3)P(S_2|S_1)P(S_2|S_2)P(S_3|S_2)P(S_1|S_3)P(S_2|S_1)P(S_3|S_2)
  = 0.1 \times 0.3 \times 0.3 \times 0.7 \times 0.2 \times 0.3 \times 0.3 \times 0.2 = 0.00002268
  \]
 Observable Markov Model (cont.)

- Example 2: A three-state Markov chain for the Dow Jones Industrial average

The probability of 5 consecutive up days

\[ P(5 \text{ consecutive up days}) = P(1,1,1,1,1) = \pi_1 a_{11} a_{11} a_{11} a_{11} = 0.5 \times (0.6)^4 = 0.0648 \]
Observable Markov Model (cont.)

- Example 3: Given a Markov model, what is the mean occupancy duration of each state $i$

\[ p_i(d) = \text{prob. density function of duration } d \text{ in state } i \]
\[ = (a_{ii})^{d-1}(1 - a_{ii}) \]

Expected number of duration in a state

\[ \bar{d}_i = \sum_{d=1}^{\infty} dp_i(d) = \sum_{d=1}^{\infty} d(a_{ii})^{d-1}(1 - a_{ii}) = (1 - a_{ii}) \frac{\partial}{\partial a_{ii}} \sum_{d=1}^{\infty}(a_{ii})^d \]
\[ = (1 - a_{ii}) \frac{\partial}{\partial a_{ii}} \frac{1}{1 - a_{ii}} = \frac{1}{1 - a_{ii}} \]
Hidden Markov Model

(a) Illustration of a two-layered random process. (b) An HMM model of the process in (a).
Hidden Markov Model (cont.)

• HMM, an extended version of Observable Markov Model
  – The observation is turned to be a probabilistic function (discrete or continuous) of a state instead of an one-to-one correspondence of a state
  – The model is a doubly embedded stochastic process with an underlying stochastic process that is not directly observable (hidden)
    • What is hidden? *The State Sequence!*
      According to the observation sequence, we are not sure which state sequence generates it!

• Elements of an HMM (the State-Output HMM) $\lambda = \{S, A, B, \pi\}$
  – $S$ is a set of $N$ states
  – $A$ is the $N \times N$ matrix of transition probabilities between states
  – $B$ is a set of $N$ probability functions, each describing the observation probability with respect to a state
  – $\pi$ is the vector of initial state probabilities
Hidden Markov Model (cont.)

• Two major assumptions
  – First order (Markov) assumption
    • The state transition depends only on the origin and destination
    • Time-invariant

\[
P(s_t = j | s_{t-1} = i) = P(s_{\tau} = j | s_{\tau-1} = i) = P(j | i) = A_{i,j}
\]

– Output-independent assumption
  • All observations are dependent on the state that generated them, not on neighboring observations

\[
P(o_t | s_t, \ldots, o_{t-2}, o_{t-1}, o_{t+1}, o_{t+2} \ldots) = P(o_t | s_t)
\]
Hidden Markov Model (cont.)

- Two major types of HMMs according to the observations
  - **Discrete and finite observations:**
    - The observations that all distinct states generate are finite in number
      \[ V = \{ v_1, v_2, v_3, \ldots, v_M \}, \quad v_k \in \mathbb{R}^L \]
    - In this case, the set of observation probability distributions
      \[ B = \{ b_j(v_k) \}, \text{ is defined as } \quad b_j(v_k) = P(o_t = v_k | s_t = j), \quad 1 \leq k \leq M, \quad 1 \leq j \leq N \]
      \( o_t \): observation at time \( t \), \( s_t \): state at time \( t \)
      \( \Rightarrow \) for state \( j \), \( b_j(v_k) \) consists of only \( M \) probability values
Hidden Markov Model (cont.)

- Two major types of HMMs according to the observations
  - **Continuous and infinite observations:**
    - The observations that all distinct states generate are infinite and continuous, that is, $V = \{v | v \in \mathbb{R}^L\}$
    - In this case, the set of observation probability distributions $B = \{b_j(v)\}$, is defined as $b_j(v) = f_{o|s}(o_t = v | s_t = j)$, $1 \leq j \leq N$
      $\Rightarrow b_j(v)$ is a **continuous probability density function (pdf)** and is often a mixture of Multivariate Gaussian (Normal) Distributions

$$b_j(v) = \sum_{k=1}^{M} w_{jk} \left( \frac{1}{(2\pi)^{L/2} |\Sigma_{jk}|^{1/2}} \exp \left( -\frac{1}{2} (v - \mu_{jk})^t \Sigma_{jk}^{-1} (v - \mu_{jk}) \right) \right)$$

- **Mixture Weight**
- **Covariance Matrix**
- **Observation Vector**
- **Mean Vector**
Hidden Markov Model (cont.)

- Multivariate Gaussian Distributions
  - When \( \mathbf{X} = (X_1, X_2, \ldots, X_L) \) is a \( L \)-dimensional random vector, the multivariate Gaussian pdf has the form:

  \[
  f(\mathbf{X} = \mathbf{x} | \mu, \Sigma) = N(\mathbf{x}; \mu, \Sigma) = \frac{1}{(2\pi)^{L/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^t \Sigma^{-1}(\mathbf{x} - \mu)\right)
  \]

  where \( \mu \) is the \( L \)-dimensional mean vector, \( \mu = E[\mathbf{x}] \)
  \( \Sigma \) is the covariance matrix, \( \Sigma = E[(\mathbf{x} - \mu)(\mathbf{x} - \mu)^t] = E[\mathbf{xx}^t] - \mu\mu^t \)
  and \( |\Sigma| \) is the determinant of \( \Sigma \)

  The \( i-j^{th} \) element \( \sigma_{ij} \) of \( \Sigma \), \( \sigma_{ij} = E[(x_i - \mu_i)(x_j - \mu_j)] = E[x_ix_j] - \mu_i\mu_j \)

  - If \( X_1, X_2, \ldots, X_L \) are independent, the covariance matrix is reduced to diagonal covariance
    - The distribution as \( L \) independent scalar Gaussian distributions
    - Model complexity is significantly reduced
Hidden Markov Model (cont.)

- Multivariate Gaussian Distributions

*Figure 3.12* A two-dimensional multivariate Gaussian distribution with independent random variables $x_1$ and $x_2$ that have the same variance.

*Figure 3.13* Another two-dimensional multivariate Gaussian distribution with independent random variable $x_1$ and $x_2$ which have different variances.
Hidden Markov Model (cont.)

- Covariance matrix of the correlated feature vectors (Mel-Frequency filter bank outputs)

- Covariance matrix of the partially de-correlated feature vectors (MFCC cepstrum without $C_0$)
Hidden Markov Model (cont.)

- Multivariate Mixture Gaussian Distributions (cont.)
  - More complex distributions with multiple local maxima can be approximated by Gaussian (a unimodal distribution) mixture

\[ f(x) = \sum_{k=1}^{M} w_k N_k(x; \mu_k, \Sigma_k), \quad \sum_{k=1}^{M} w_k = 1 \]

- Gaussian mixtures with enough mixture components can approximate any distribution
Hidden Markov Model (cont.)

- **Example 4:** a 3-state discrete HMM $\lambda$

$$A = \begin{bmatrix} 0.6 & 0.3 & 0.1 \\ 0.1 & 0.7 & 0.2 \\ 0.3 & 0.2 & 0.5 \end{bmatrix}$$

$b_1(A) = 0.3, b_1(B) = 0.2, b_1(C) = 0.5$

$b_2(A) = 0.7, b_2(B) = 0.1, b_2(C) = 0.2$

$b_3(A) = 0.3, b_3(B) = 0.6, b_3(C) = 0.1$

$\pi = [0.4 \ 0.5 \ 0.1]$ 

- Given a sequence of observations $O = \{ABC\}$, there are 27 possible corresponding state sequences, and therefore the corresponding probability is

$$P(O|\lambda) = \sum_{i=1}^{27} P(O, S_i | \lambda) = \sum_{i=1}^{27} P(O|S_i, \lambda)P(S_i | \lambda), \quad S_i : \text{state sequence}$$

*E.g.* when $S_i = \{s_2 s_2 s_3\}$, 

$$P(O|S_i, \lambda) = P(A|s_2)P(B|s_2)P(C|s_3) = 0.7 \times 0.1 \times 0.1 = 0.007$$

$$P(S_i | \lambda) = P(s_2)P(s_2|s_2)P(s_3|s_2) = 0.5 \times 0.7 \times 0.2 = 0.07$$
Hidden Markov Model (cont.)

• Notation:
  - $O = \{o_1o_2o_3......o_T\}$: the observation (feature) sequence
  - $S = \{s_1s_2s_3......s_T\}$: the state sequence
  - $\lambda$: model, for HMM, $\lambda = \{A, B, \pi\}$
  - $P(O|\lambda)$: 用 model $\lambda$ 計算 $O$ 的機率值
  - $P(O|S, \lambda)$: 在 $O$ 是 state sequence $S$ 所產生的前提下, 用 model $\lambda$ 計算 $O$ 的機率值
  - $P(O,S|\lambda)$: 用 model $\lambda$ 計算 $[O, S]$ 兩者同時成立的機率值
  - $P(S|O, \lambda)$: 在已知 $O$ 的前提下, 用 model $\lambda$ 計算 $S$ 的機率值

• Useful formula
  - Bayesian Rule:
    
    $$P(A|B) = \frac{P(A,B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$
    
    $$P(A,B, \lambda) = \frac{P(A,B, \lambda)}{P(B, \lambda)} = \frac{P(B|A, \lambda)P(A|\lambda)}{P(B|\lambda)}$$
    
    $\lambda$ : model describing the probability

    $$P(A,B) = P(B|A)P(A) = P(A|B)P(B)$$  chain rule
Hidden Markov Model (cont.)

• Useful formula (Cont.):
  – Total Probability Theorem

\[
P(A) = \begin{cases}
\sum_{B} P(A, B) = \sum_{B} P(A|B)P(B), & \text{if } B \text{ is discrete and disjoint} \\
\int_{B} f(A, B)dB = \int_{B} f(A|B)f(B)dB, & \text{if } B \text{ is continuous}
\end{cases}
\]

if \(x_1, x_2, \ldots, x_n\) are independent,

\(\Rightarrow P(x_1, x_2, \ldots, x_n) = P(x_1)P(x_2)\ldots P(x_n)\)

Expectation

\[
E_z[q(z)] = \begin{cases}
\sum_{k} P(z = k)q(k), & z : \text{discrete} \\
\int_{z} f_z(z)q(z)dz, & z : \text{continuous}
\end{cases}
\]
Three Basic Problems for HMM

• Given an observation sequence $O=(o_1,o_2,\ldots,o_T)$, and an HMM $\lambda=(S,A,B,\pi)$
  
  – Problem 1:
  
    How to efficiently compute $P(O|\lambda)$?
  
    $\Rightarrow$ Evaluation problem
  
  – Problem 2:
  
    How to choose an optimal state sequence $S=(s_1,s_2,\ldots,s_T)$?
  
    $\Rightarrow$ Decoding Problem
  
  – Problem 3:
  
    How to adjust the model parameter $\lambda=(A,B,\pi)$ to maximize $P(O|\lambda)$?
  
    $\Rightarrow$ Learning / Training Problem
Basic Problem 1 of HMM (cont.)

Given \( \mathbf{O} \) and \( \lambda \), find \( P(\mathbf{O}|\lambda) = \text{Prob}[\text{observing } \mathbf{O} \text{ given } \lambda] \)

- **Direct Evaluation**
  - Evaluating all possible state sequences of length \( T \) that generating observation sequence \( \mathbf{O} \)

\[
P(\mathbf{O} | \lambda) = \sum_{\text{all } S} P(\mathbf{O}, S | \lambda) = \sum_{\text{all } S} P(\mathbf{O} | S, \lambda)P(S | \lambda)
\]

- \( P(S | \lambda) \): The probability of each path \( S \)
  - By Markov assumption (First-order HMM)

\[
P(S | \lambda) = P(s_1 | \lambda) \prod_{t=2}^{T} P(s_t | s_{t-1}, \lambda)
\]

\[
\approx P(s_1 | \lambda) \prod_{t=2}^{T} P(s_t | s_{t-1}, \lambda)
\]

\[
= \pi_{s_1} a_{s_1 s_2} a_{s_2 s_3} \ldots a_{s_{T-1} s_T}
\]
Basic Problem 1 of HMM (cont.)

• Direct Evaluation (cont.)
  
  - \[ P(O | S, \lambda) \]: The joint output probability along the path \( S \)
    
    • By output-independent assumption
      
      - The probability that a particular observation symbol/vector is emitted at time \( t \) depends only on the state \( s_t \) and is conditionally independent of the past observations

\[
P(O | S, \lambda) = P(o_1^T | s_1^T, \lambda) \prod_{t=2}^{T} P(o_t | o_1^{t-1}, s_1^T, \lambda) \\
\approx \prod_{t=1}^{T} P(o_t | s_t, \lambda) \\
= \prod_{t=1}^{T} b_{s_t}(o_t)
\]
Basic Problem 1 of HMM (cont.)

- Direct Evaluation (Cont.)

\[
P(O|\lambda) = \sum_{all \, S} P(S|\lambda)P(O|S, \lambda) \]

\[
= \sum_{all \, s} \left[ \pi_{s_1} a_{s_1s_2} a_{s_2s_3} \ldots a_{s_{T-1}s_T} \prod_{s_1} b_{s_1}(o_1)b_{s_2}(o_2)\ldots b_{s_T}(o_T) \right] \]

\[
= \sum_{s_1, s_2, \ldots, s_T} \pi_{s_1} b_{s_1}(o_1)a_{s_1s_2} b_{s_2}(o_2)\ldots a_{s_{T-1}s_T} b_{s_T}(o_T) \]

- Huge Computation Requirements: \(O(N^T)\)
  - Exponential computational complexity

**Complexity** : \((2T-1)N^T\text{ MUL} \approx 2TN^T, N^T-1\text{ ADD}\)

- A more efficient algorithms can be used to evaluate \(P(O|\lambda)\)
  - *Forward/Backward Procedure/Algorithm*
Basic Problem 1 of HMM (cont.)

• Direct Evaluation (Cont.)

State-time Trellis Diagram

- Each node represents a state at a specific time step.
- Arrows indicate transitions between states.
- States are labeled as $s_1$, $s_2$, and $s_3$.
- Observations are represented by $O_1$, $O_2$, and $O_3$.

- Means $b_j(o_j)$ has been computed.
- Means $a_{ij}$ has been computed.

Time
Basic Problem 1 of HMM
- The Forward Procedure

• Base on the HMM assumptions, the calculation of $P \left( s_t \mid s_{t-1}, \lambda \right)$ and $P \left( o_t \mid s_t, \lambda \right)$ involves only $s_{t-1}$, $s_t$ and $o_t$, so it is possible to compute the likelihood with recursion on $t$

• Forward variable: $\alpha_t(i) = P \left( o_1o_2...o_t, s_t = i \mid \lambda \right)$
  - The probability that the HMM is in state $i$ at time $t$ having generating partial observation $o_1o_2...o_t$
Basic Problem 1 of HMM
- The Forward Procedure (cont.)

• Algorithm

  1. Initialization \( a_1(i) = \pi_i b_i(o_1), \ 1 \leq i \leq N \)

  2. Induction \( a_{t+1}(j) = \sum_{i=1}^{N} a_t(i) a_{ij} b_j(o_{t+1}), \ 1 \leq t \leq T-1, 1 \leq j \leq N \)

  3. Termination \( P(O|\lambda) = \sum_{i=1}^{N} a_T(i) \)

    - Complexity: \( O(N^2T) \)

      \[
      \text{MUL} : N(N+1)(T-1) + N \approx N^2T
      \]

      \[
      \text{ADD} : (N-1)N(T-1) + (N-1) \approx N^2T
      \]

• Based on the lattice (trellis) structure

  - Computed in a time-synchronous fashion from left-to-right, where each cell for time \( t \) is completely computed before proceeding to time \( t+1 \)

• All state sequences, regardless how long previously, merge to \( N \) nodes (states) at each time instance \( t \)
Basic Problem 1 of HMM
- The Forward Procedure (cont.)

\[ \alpha_t(j) = P(o_1o_2\ldots o_t, s_t = j|\lambda) \]
\[ = P(o_1o_2\ldots o_t | s_t = j, \lambda)P(s_t = j|\lambda) \]
\[ = P(o_1o_2\ldots o_{t-1} | s_t = j, \lambda)P(o_t | s_t = j, \lambda)P(s_t = j|\lambda) \]
\[ = P(o_1o_2\ldots o_{t-1}, s_t = j|\lambda)P(o_t | s_t = j, \lambda)P(s_t = j|\lambda) \]
\[ = P(o_1o_2\ldots o_{t-1}, s_t = j|\lambda)b_j(o_t) \]
\[ = \sum_{i=1}^{N} P(o_1o_2\ldots o_{t-1}, s_{t-1} = i, s_t = j|\lambda)b_j(o_t) \]
\[ = \sum_{i=1}^{N} P(o_1o_2\ldots o_{t-1}, s_{t-1} = i|\lambda)P(s_t = j|o_1o_2\ldots o_{t-1}, s_{t-1} = i, \lambda)b_j(o_t) \]
\[ = \sum_{i=1}^{N} P(o_1o_2\ldots o_{t-1}, s_{t-1} = i|\lambda)P(s_t = j|s_{t-1} = i, \lambda)b_j(o_t) \]
\[ = \sum_{i=1}^{N} \alpha_{t-1}(i)a_ij b_j(o_t) \]
Basic Problem 1 of HMM
- The Forward Procedure (cont.)

- $\alpha_3(3) = P(o_1, o_2, o_3, s_3=3 | \lambda)$
  
  $= [\alpha_2(1) * a_{13} + \alpha_2(2) * a_{23} + \alpha_2(3) * a_{33}] b_3(o_3)$
Basic Problem 1 of HMM
- The Forward Procedure (cont.)

• A three-state Hidden Markov Model for the *Dow Jones Industrial average*

\[
\begin{align*}
\text{state 1:} & \quad 0.35 \quad \text{to} \quad 0.179 \\
\text{state 2:} & \quad 0.02 \quad \text{to} \quad 0.008 \\
\text{state 3:} & \quad 0.09 \quad \text{to} \quad 0.036
\end{align*}
\]

\[
\begin{align*}
0.6 & \quad \text{to} \quad 0.179 \\
0.5 & \quad \text{to} \quad 0.179 \\
0.4 & \quad \text{to} \quad 0.008 \\
0.7 & \quad \text{to} \quad 0.036 \\
0.1 & \quad \text{to} \quad 0.036
\end{align*}
\]

\[
(0.6 \times 0.35 + 0.5 \times 0.02 + 0.4 \times 0.009) \times 0.7 = 0.1792
\]

*Figure 8.4* The forward trellis computation for the HMM of the *Dow Jones Industrial average.*
Basic Problem 1 of HMM  
- The Backward Procedure  

• Backward variable:  \( \beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots, o_T | s_t = i, \lambda) \)

1. Initialization:  \( \beta_T(i) = 1, \ 1 \leq i \leq N \)

2. Induction:  \( \beta_i(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \ 1 \leq t \leq T-1, 1 \leq i \leq N \)

3. Termination:  \( P(O|\lambda) = \sum_{j=1}^{N} \pi_j b_j(o_1) \beta_1(j) \)

Complexity MUL:  \( 2N^2(T-1) + 2N \approx N^2T \);

ADD:  \( (N-1)N(T-1) + N \approx N^2T \)
Basic Problem 1 of HMM
- Backward Procedure (cont.)

• Why \( P(O, s_t = i \mid \lambda) = \alpha_t(i) \beta_t(i) \) ?

\[
\begin{align*}
\alpha_t(i) \beta_t(i) & = P(o_1, o_2, \ldots, o_t, s_t = i \mid \lambda) \cdot P(o_{t+1}, o_{t+2}, \ldots, o_T \mid s_t = i, \lambda) \\
& = P(o_1, o_2, \ldots, o_t \mid s_t = i, \lambda) P(s_t = i \mid \lambda) P(o_{t+1}, o_{t+2}, \ldots, o_T \mid s_t = i, \lambda) \\
& = P(o_1, \ldots, o_t, \ldots, o_T \mid s_t = i, \lambda) P(s_t = i \mid \lambda) \\
& = P(O, s_t = i \mid \lambda) \\
& = P(O \mid \lambda) = \sum_{i=1}^{N} P(O, s_t = i \mid \lambda) = \sum_{i=1}^{N} \alpha_t(i) \beta_t(i)
\end{align*}
\]
Basic Problem 1 of HMM
- The Backward Procedure (cont.)

\[ \beta_2(3) = P(o_3, o_4, ..., o_T | s_2 = 3, \lambda) = a_{31} \ast b_1(o_3) \ast \beta_3(1) + a_{32} \ast b_2(o_3) \ast \beta_3(2) + a_{33} \ast b_1(o_3) \ast \beta_3(3) \]
Basic Problem 2 of HMM

How to choose an optimal state sequence $S=(s_1, s_2, \ldots, s_T)$?

- The first optimal criterion: Choose the states $s_t$ are individually most likely at each time $t$

Define a posteriori probability variable $\gamma_t(i) = P(s_t = i | O, \lambda)$

$$\gamma_t(i) = \frac{P(s_t = i, O | \lambda)}{P(O | \lambda)} = \frac{P(s_t = i, O | \lambda)}{\sum_{m=1}^{N} P(s_t = m, O | \lambda)} = \frac{\alpha_t(i) \beta_t(i)}{\sum_{m=1}^{N} \alpha_t(m) \beta_t(m)}$$

- Solution: $s_t^* = \text{arg} \max [\gamma_t(i)], 1 \leq t \leq T$

  - Problem: maximizing the probability at each time $t$ individually $S^* = s_1^* s_2^* \ldots s_T^*$ may not be a valid sequence (e.g. $a_{s_t^* s_{t+1}^*} = 0$)
Basic Problem 2 of HMM (cont.)

- \( P(s_3 = 3, O | \lambda) = \alpha_3(3) \beta_3(3) \)
Basic Problem 2 of HMM
- The Viterbi Algorithm

• The second optimal criterion: The Viterbi algorithm can be regarded as the dynamic programming algorithm applied to the HMM or as a modified forward algorithm

  – Instead of summing up probabilities from different paths coming to the same destination state, the Viterbi algorithm picks and remembers the best path
    • Find a single optimal state sequence \( S = (s_1, s_2, \ldots, s_T) \)
      – How to find the second, third, etc., optimal state sequences (difficult ?)

  – The Viterbi algorithm also can be illustrated in a trellis framework similar to the one for the forward algorithm
    • State-time trellis diagram
Basic Problem 2 of HMM
- The Viterbi Algorithm (cont.)

- Algorithm
  Find a best state sequence $S = (s_1, s_2, ..., s_T)$ for a given observation $O = (o_1, o_2, ..., o_T)$?
  Define a new variable
  \[ \delta_t(i) = \max_{s_1, s_2, ..., s_{t-1}} P[s_1, s_2, ..., s_{t-1}, s_t = i, o_1, o_2, ..., o_t | \lambda] \]
  \[ = \text{the best score along a single path at time } t, \text{ which accounts} \]
  \[ \text{for the first } t \text{ observation and ends in state } i \]

  By induction : $\delta_{t+1}(j) = \max_{1 \leq i \leq N} \delta_t(i)a_{ij} b_j(o_{t+1})$
  \[ \psi_{t+1}(j) = \arg \max_{1 \leq i \leq N} \delta_t(i)a_{ij} \text{ .... For backtracing} \]

  We can backtrace from $s_T^* = \arg \max_{1 \leq i \leq N} \delta_T(i)$
  - Complexity: $O(N^2T)$
Basic Problem 2 of HMM
- The Viterbi Algorithm (cont.)

\[ \delta_3(3) \]
Basic Problem 2 of HMM
- The Viterbi Algorithm (cont.)

- A three-state Hidden Markov Model for the *Dow Jones Industrial average*

*Figure 8.5* The Viterbi trellis computation for the HMM of the Dow Jones Industrial average.
Basic Problem 2 of HMM  
- The Viterbi Algorithm (cont.)

- Algorithm in the **logarithmic** form

Find a best state sequence \( S = (s_1, s_2, ..., s_T) \) for a given observation \( O = (o_1, o_2, ..., o_T) \)?

Define a new variable

\[
\delta_t(i) = \max_{s_1, s_2, ..., s_{t-1}} \log P[s_1, s_2, ..., s_{t-1}, s_t = i, o_1, o_2, ..., o_t | \lambda] 
\]

= the best score along a single path at time \( t \), which accounts for the first \( t \) observation and ends in state \( i \)

By induction \( \therefore \) \( \delta_{t+1}(j) = \left[ \max_{1 \leq i \leq N} (\delta_t(i) + \log a_{ij}) \right] + \log b_j(o_{t+1}) \)

\( \psi_{t+1}(j) = \arg \max_{1 \leq i \leq N} (\delta_t(i) + \log a_{ij}) \) .... For backtracing

We can backtrace from \( s_T^* = \arg \max_{1 \leq i \leq N} \delta_T(i) \)
Homework-1

• A three-state Hidden Markov Model for the *Dow Jones Industrial average*

![Diagram of a three-state Hidden Markov Model]

**Figure 8.2** A hidden Markov model for the Dow Jones Industrial average. The three states no longer have deterministic meanings as the Markov chain illustrated in Figure 8.1.

- **Find the probability:**
  \[ P(\text{up, up, unchanged, down, unchanged, down, up}|\lambda) \]

- **Find the optimal state sequence of the model which generates the observation sequence:** (up, up, unchanged, down, unchanged, down, up)
Probability Addition in F-B Algorithm

• In Forward-backward algorithm, operations usually implemented in logarithmic domain

• Assume that we want to add $P_1$ and $P_2$

  if $P_1 \geq P_2$

  $\log_b (P_1 + P_2) = \log P_1 + \log_b \left(1 + b^{\log_b P_2 - \log_b P_1}\right)$

  else

  $\log_b (P_1 + P_2) = \log P_2 + \log_b \left(1 + b^{\log_b P_1 - \log_b P_2}\right)$

The values of $\log_b \left(1 + b^{x}\right)$ can be saved in a table to speedup the operations
Probability Addition in F-B Algorithm (cont.)

• An example code

```c
#define LZERO (-1.0E10)   // ~log(0)
#define LSMALL (-0.5E10)   // log values < LSMALL are set to LZERO
#define minLogExp -log(-LZERO) // ~=-23

double LogAdd(double x, double y)
{
  double temp,diff,z;
  if (x<y)
  {
    temp = x; x = y; y = temp;
  }
  diff = y-x; //notice that diff <= 0
  if (diff<minLogExp) // if y' is far smaller than x'
    return (x<LSMALL) ? LZERO:x;
  else
  {
    z = exp(diff);
    return x+log(1.0+z);
  }
}
```
Basic Problem 3 of HMM
Intuitive View

• How to adjust (re-estimate) the model parameter $\lambda=(A,B,\pi)$ to maximize $P(O|\lambda)$?

  – The most difficult of the three problems, because there is no known analytical method that maximizes the joint probability of the training data in a close form

  – The data is incomplete because of the hidden state sequences

  – Well-solved by the Baum-Welch (known as forward-backward) algorithm and EM (Expectation-Maximization) algorithm
    • Iterative update and improvement
    • Based on Maximum Likelihood (ML) criterion
Maximum Likelihood (ML) Estimation

- Hard Assignment

\[ P(B|S_1) = \frac{2}{4} = 0.5 \]
\[ P(W|S_1) = \frac{2}{4} = 0.5 \]
Maximum Likelihood (ML) Estimation

- Soft Assignment

\[ \gamma_t(1) = P(s_t = s_1 | o, \lambda) \quad \gamma_t(2) = P(s_t = s_2 | o, \lambda) \]

\[ \sum \gamma_t(1) + \gamma_t(2) = 1 \]

\[ P(B|S_1) = \frac{(0.7+0.9)}{(0.7+0.4+0.9+0.5)} \]
\[ = \frac{1.6}{2.5} = 0.64 \]

\[ P(B|S_2) = \frac{(0.3+0.1)}{(0.3+0.6+0.1+0.5)} \]
\[ = \frac{0.4}{1.5} = 0.27 \]

\[ P(W|S_1) = \frac{(0.4+0.5)}{(0.7+0.4+0.9+0.5)} \]
\[ = \frac{0.9}{2.5} = 0.36 \]

\[ P(W|S_2) = \frac{(0.6+0.5)}{(0.3+0.6+0.1+0.5)} \]
\[ = \frac{0.11}{1.5} = 0.73 \]
Basic Problem 3 of HMM
Intuitive View (cont.)

• Relationship between the forward and backward variables

\[
\alpha_t(i) = P(o_1, o_2, \ldots, o_t, s_t = i | \lambda) = \sum_{j=1}^{N} \alpha_{t-1}(j)a_{ji}b_j(o_t)
\]

\[
\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots, o_T | s_t = i, \lambda) = \sum_{j=1}^{N} \beta_{t+1}(j)b_j(o_{t+1})a_{ij}
\]

\[
\sum_{i=1}^{N} \alpha_t(i) \beta_t(i) = P(O, s_t = i | \lambda)
\]

*Figure 8.6* The relationship of \(\alpha_{t-1}\) and \(\alpha_t\) and \(\beta_t\) and \(\beta_{t+1}\) in the forward-backward algorithm.
Basic Problem 3 of HMM
Intuitive View (cont.)

- Define a new variable:
  \[ \xi_t(i, j) = P(s_t = i, s_{t+1} = j | O, \lambda) \]
  - Probability being at state \(i\) at time \(t\) and at state \(j\) at time \(t+1\)

  \[
  \xi_t(i, j) = \frac{P(s_t = i, s_{t+1} = j, O | \lambda)}{P(O | \lambda)}
  = \frac{\alpha_t(i) a_{ij} b_{j}(o_{t+1}) \beta_{t+1}(j)}{P(O | \lambda)}
  = \frac{\alpha_t(i) a_{ij} b_{j}(o_{t+1}) \beta_{t+1}(j)}{\sum_{m=1}^{N} \sum_{n=1}^{N} \alpha_t(m) a_{mn} b_{n}(o_{t+1}) \beta_{t+1}(n)}
  \]

- Recall the posteriori probability variable:

  \[ \gamma_t(i) = P(s_t = i | O, \lambda) \]
  Note: \(\gamma_t(i)\) also can be represented as

  \[ \gamma_t(i) = \sum_{j=1}^{N} P(s_t = i, s_{t+1} = j | O, \lambda) = \sum_{j=1}^{N} \xi_t(i, j) \quad (\text{for} \ t < T) \]
Basic Problem 3 of HMM
Intuitive View (cont.)

• \( P(s_3 = 3, s_4 = 1, O | \lambda) = \alpha_3(3) * a_{31} * b_1(o_4) * \beta_1(4) \)
Basic Problem 3 of HMM
Intuitive View (cont.)

- \( \xi_t(i, j) = P(s_t = i, s_{t+1} = j | O, \lambda) \)
  \[ \sum_{t=1}^{T-1} \xi_t(i, j) = \text{expected number of transitions from state } i \text{ to state } j \text{ in } O \]

- \( \gamma_t(i) = P(s_t = i | O, \lambda) \)
  \[ \sum_{t=1}^{T-1} \gamma_t(i) = \sum_{t=1}^{T-1} \sum_{j=1}^{N} \xi_t(i, j) = \text{expected number of transitions from state } i \text{ in } O \]

- A set of reasonable re-estimation formula for \( \{A, \pi\} \) is

- \( \overline{\pi}_i = \text{expected frequency (number of times) in state } i \text{ at time } t = 1 \)
  \[ \overline{\pi}_i = \gamma_1(i) \]

- \( \overline{a}_{ij} = \frac{\text{expected number of transition from state } i \text{ to state } j}{\text{expected number of transition from state } i} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \)

**Formulae for Single Training Utterance**
Basic Problem 3 of HMM
Intuitive View (cont.)

• A set of reasonable re-estimation formula for \( \{B\} \) is
  
  – For discrete and finite observation \( b_j(v_k) = P(o_t = v_k | s_t = j) \)

  \[
  \overline{b}_j(v_k) = P(o = v_k | s = j) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j} = \frac{\sum_{t=1}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}
  \]

  – For continuous and infinite observation \( b_j(v) = f_{o|s}(o_t = v | s_t = j) \),

  \[
  \overline{b}_j(v) = \sum_{k=1}^{M} \overline{c}_{jk} N(v; \mu_{jk}, \Sigma_{jk}) = \sum_{k=1}^{M} \overline{c}_{jk} \left( \frac{1}{(\sqrt{2\pi})^L |\Sigma_{jk}|^{1/2}} \exp \left( -\frac{1}{2} (v - \mu_{jk})^t \Sigma_{jk}^{-1} (v - \mu_{jk}) \right) \right)
  \]

  Modeled as a mixture of multivariate Gaussian distributions
Basic Problem 3 of HMM
Intuitive View (cont.)

– For continuous and infinite observation (Cont.)

• Define a new variable $\gamma_t(j,k)$
  – $\gamma_t(j,k)$ is the probability of being in state $j$ at time $t$
    with the $k$-th mixture component accounting for $o_t$

\[
\gamma_t(j,k) = P(s_t = j, m_t = k | O, \lambda)
= P(s_t = j | O, \lambda) P(m_t = k | s_t = j, O, \lambda)
= \gamma_t(j) P(m_t = k | s_t = j, O, \lambda)
= \gamma_t(j) \frac{P(m_t = k, O | s_t = j, \lambda)}{P(O | s_t = j, \lambda)}
= \gamma_t(j) \frac{P(m_t = k | s_t = j, \lambda) P(O | s_t = j, m_t = k, \lambda)}{P(O | s_t = j, \lambda)}
\]

...... (observation independence assumption is applied)

\[
= \gamma_t(j) \frac{P(m_t = k | s_t = j, \lambda) P(o_t | s_t = j, m_t = k, \lambda)}{P(o_t | s_t = j, \lambda)}
\]

\[
= \left[ \frac{\alpha_t(j) \beta_t(j)}{\sum_{s=1}^{N} \alpha_t(s) \beta_t(s)} \right] \frac{c_{jk} N(o_t; \mu_{jk}, \Sigma_{jk})}{\sum_{m=1}^{M} c_{jm} N(o_t; \mu_{jm}, \Sigma_{jm})}
\]

Note: $\gamma_t(j) = \sum_{m=1}^{M} \gamma_t(j,m)$
Basic Problem 3 of HMM
Intuitive View (cont.)

– For continuous and infinite observation (Cont.)

\[
\bar{c}_{jk} = \frac{\text{expected number of times in state } j \text{ and mixture } k}{\text{expected number of times in state } j} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T} \sum_{m=1}^{M} \gamma_t(j,m)}
\]

\[
\bar{\mu}_{jk} = \text{weighted average (mean) of observations at state } j \text{ and mixture } k = \frac{\sum_{t=1}^{T} \gamma_t(j,k) \cdot o_t}{\sum_{t=1}^{T} \gamma_t(j,k)}
\]

\[
\bar{\Sigma}_{jk} = \text{weighted covariance of observations at state } j \text{ and mixture } k
\]

\[
= \frac{\sum_{t=1}^{T} \gamma_t(j,k) \cdot (o_t - \bar{\mu}_{jk}) (o_t - \bar{\mu}_{jk})^T}{\sum_{t=1}^{T} \gamma_t(j,k)}
\]

**Formulae for Single Training Utterance**
Basic Problem 3 of HMM
Intuitive View (cont.)

• Multiple Training Utterances
Basic Problem 3 of HMM
Intuitive View (cont.)

- For continuous and infinite observation (Cont.)

\[ \overline{\pi}_i = \text{expected frequency (number of times) in state } i \text{ at time } (t = 1) = \frac{1}{L} \sum_{l=1}^{L} \gamma_1^l(i) \]

\[ \overline{a}_{ij} = \frac{\text{expected number of transition from state } i \text{ to state } j}{\text{expected number of transition from state } i} = \frac{\sum_{l=1}^{L} \sum_{t=1}^{T_l} \gamma_i^l(i,j)}{\sum_{l=1}^{L} \sum_{t=1}^{T_l} \gamma_i^l(i)} \]

\[ \overline{c}_{jk} = \frac{\text{expected number of times in state } j \text{ and mixture } k}{\text{expected number of times in state } j} = \frac{\sum_{l=1}^{L} \sum_{t=1}^{T_l} \gamma_i^l(j,k)}{\sum_{l=1}^{L} \sum_{t=1}^{T_l} \sum_{m=1}^{M} \gamma_i^l(j,m)} \]

\[ \mu_{jk} = \text{weighted average (mean) of observations at state } j \text{ and mixture } k = \frac{\sum_{l=1}^{L} \sum_{t=1}^{T_l} \gamma_i^l(j,k) \cdot o_t}{\sum_{l=1}^{L} \sum_{t=1}^{T_l} \gamma_i^l(j,k)} \]

\[ \Sigma_{jk} = \text{weighted covariance of observations at state } j \text{ and mixture } k \]

\[ = \frac{\sum_{l=1}^{L} \sum_{t=1}^{T_l} \gamma_i^l(j,k) \cdot (o_t - \mu_{jk})(o_t - \mu_{jk})'}{\sum_{l=1}^{L} \sum_{t=1}^{T_l} \gamma_i^l(j,k)} \]

**Formulae for Multiple (L) Training Utterances**
Basic Problem 3 of HMM
Intuitive View (cont.)

– For discrete and finite observation (cont.)

\[ \bar{\pi}_i = \text{expected frequency (number of times) in state } i \text{ at time } (t = 1) = \frac{1}{L} \sum_{l=1}^{L} \gamma^l(i) \]

\[ \bar{a}_{ij} = \frac{\text{expected number of transition from state } i \text{ to state } j}{\text{expected number of transition from state } i} = \frac{\sum_{l=1}^{L} \sum_{t=1}^{T_l} \xi_t^l(i,j)}{\sum_{l=1}^{L} \sum_{t=1}^{T_l} \gamma^l(i)} \]

\[ \bar{b}_j(v_k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j} = \frac{\sum_{l=1}^{L} \sum_{t=1}^{T_l} \gamma^l(j)}{\sum_{l=1}^{L} \sum_{t=1}^{T_l} \gamma^l(j)} \]
Semicontinuous HMMs

• The HMM state mixture density functions are tied together across all the models to form a set of shared kernels
  – The semicontinuous or tied-mixture HMM
    \[
    b_j(o) = \sum_{k=1}^{M} b_j(k) f(o|\nu_k) = \sum_{k=1}^{M} b_j(k) N(o, \mu_k, \Sigma_k)
    \]
    state output
    Probability of state \(j\)
    \(k\)-th mixture weight of state \(j\)
    \(k\)-th mixture density function or \(k\)-th codeword (shared across HMMs, \(M\) is very large)
    (discrete, model-dependent)
  – A combination of the discrete HMM and the continuous HMM
    • A combination of discrete model-dependent weight coefficients and continuous model-independent codebook probability density functions
  – Because \(M\) is large, we can simply use the \(L\) most significant values \(f(o|\nu_k)\)
    • Experience showed that \(L\) is 1~3% of \(M\) is adequate
  – Partial tying of \(f(o|\nu_k)\) for different phonetic class
Semicontinuous HMMs (cont.)

$N(\mu_1, \Sigma_1)$

$N(\mu_2, \Sigma_2)$

$N(\mu_k, \Sigma_k)$

$N(\mu_M, \Sigma_M)$
HMM Topology

• Speech is time-evolving non-stationary signal
  – Each HMM state has the ability to capture some quasi-stationary segment in the non-stationary speech signal
  – A left-to-right topology is a natural candidate to model the speech signal

  [Diagram of HMM topology]

  Figure 8.8 A typical hidden Markov model used to model phonemes. There are three states (0-2) and each state has an associated output probability distribution.

  – It is general to represent a phone using 3~5 states (English) and a syllable using 6~8 states (Mandarin Chinese)
Initialization of HMM

- A good initialization of HMM training:

  **Segmental K-Means Segmentation into States**

  Assume that we have a training set of observations and an initial estimate of all model parameters.

  Step 1: The set of training observation sequences is segmented into states, based on the initial model (finding the optimal state sequence by Viterbi Algorithm).

  Step 2:
  - For discrete density HMM (using M-codeword codebook)
    \[
    b_j(k) = \frac{\text{the number of vectors with codebook index } k \text{ in state } j}{\text{the number of vectors in state } j}
    \]
  - For continuous density HMM (M Gaussian mixtures per state)
    \[\Rightarrow \text{cluster the observation vectors within each state } j \text{ into a set of } M \text{ clusters}\]
    \[
    w_{jm} = \text{number of vectors classified in cluster } m \text{ of state } j
    \]
    divided by the number of vectors in state \(j\)
    \[
    \mu_{jm} = \text{sample mean of the vectors classified in cluster } m \text{ of state } j
    \]
    \[
    \Sigma_{jm} = \text{sample covariance matrix of the vectors classified in cluster } m \text{ of state } j
    \]

  Step 3: Evaluate the model score
  - If the difference between the previous and current model scores is greater than a threshold, go back to Step 1, otherwise stop, the initial model is generated.
Initialization of HMM (cont.)

Training Data

Model Reestimation

Estimate parameters of Observation via Segmental K-means

StateSequence Segmentation

Initial Model

Model Convergence?

YES

Model Parameters

NO
Initialization of HMM (cont.)

- An example for discrete HMM
  - 3 states and 2 codeword

\[ \begin{align*}
    &b_1(v_1) = \frac{3}{4}, \quad b_1(v_2) = \frac{1}{4} \\
    &b_2(v_1) = \frac{1}{3}, \quad b_2(v_2) = \frac{2}{3} \\
    &b_3(v_1) = \frac{2}{3}, \quad b_3(v_2) = \frac{1}{3}
\end{align*} \]
Initialization of HMM (cont.)

- An example for Continuous HMM
  - 3 states and 4 Gaussian mixtures per state

```
\begin{align*}
\text{State } & = \{s_1, s_2, s_3\} \\
\text{Cluster 1 mean } & = \{\mu_{11}, \Sigma_{11}, \omega_{11}\} \\
\text{Cluster 2 mean } & = \{\mu_{12}, \Sigma_{12}, \omega_{12}\} \\
\text{Cluster 3 mean } & = \{\mu_{13}, \Sigma_{13}, \omega_{13}\} \\
\text{Cluster 4 mean } & = \{\mu_{14}, \Sigma_{14}, \phi_{14}\}
\end{align*}
```
Known Limitations of HMMs

- The assumptions of conventional HMMs in Speech Processing
  - The state duration follows an exponential distribution
    - Don’t provide adequate representation of the temporal structure of speech
      \[ d_i(t) = a_{ii}^{t-1}(1 - a_{ii}) \]
  - **First-order (Markov) assumption**: the state transition depends only on the origin and destination
  - **Output-independent assumption**: all observation frames are dependent on the state that generated them, not on neighboring observation frames

Researchers have proposed a number of techniques to address these limitations, albeit these solution have not significantly improved speech recognition accuracy for practical applications.
Known Limitations of HMMs (cont.)

• Duration modeling

Duration distributions for the seventh state of the word “seven:” empirical distribution (solid line); Gauss fit (dashed line); gamma fit (dotted line); and (d) geometric fit (dash-dot line).
Known Limitations of HMMs (cont.)

- The HMM parameters trained by the *Baum-Welch* algorithm and *EM* algorithm were only locally optimized.
Homework-2

TrainSet 1:
1. ABBCABCAABC
2. ABCABC
3. ABCA ABC
4. BBABCB
5. BCAABCCAB
6. CACCABCA
7. ABCABCA
8. CABCA
9. CABCA

TrainSet 2:
1. BBBCCBC
2. CCBABB
3. AACCBBB
4. BBABBAC
5. CCAABBAB
6. BBBCCBAA
7. ABBBBABA
8. CCCCC
9. BBAAA

{s1} {A:.34,B:.33,C:.33} {A:.33,B:.34,C:.33} {A:.33,B:.34,C:.33}
Homework-2 (cont.)

P1. Please specify the model parameters after the first and 50th iterations of Baum-Welch training.

P2. Please show the recognition results by using the above training sequences as the testing data (The so-called inside testing).
   *You have to perform the recognition task with the HMMs trained from the first and 50th iterations of Baum-Welch training, respectively.

P3. Which class do the following testing sequences belong to?
   
   ABCABCCAB  
   AABABCCCCBBB

P4. What are the results if Observable Markov Models were instead used in P1, P2 and P3?
Isolated Word Recognition

Speech Signal → Feature Extraction → Feature Sequence X → Feature Extraction

- **Word Model** $M_1$
  \[ P(X | M_1) \]
  - Likelihood of $M_1$

- **Word Model** $M_2$
  \[ P(X | M_2) \]
  - Likelihood of $M_2$

- **Word Model** $M_v$
  \[ P(X | M_v) \]
  - Likelihood of $M_v$

- **Word Model** $M_{Sil}$
  \[ P(X | M_{Sil}) \]
  - Likelihood of $M_{Sil}$

**Most Like Word Selector**

Label $X = \arg \max_k P(X | M_k)$

**Viterbi Approximation**

$Label(X) = \arg \max_k \left[ \max_s P(X, S | M_k) \right]$
Measures of ASR Performance

• Evaluating the performance of automatic speech recognition (ASR) systems is critical, and the Word Recognition Error Rate (WER) is one of the most important measures.

• There are typically three types of word recognition errors:
  – Substitution
    • An incorrect word was substituted for the correct word
  – Deletion
    • A correct word was omitted in the recognized sentence
  – Insertion
    • An extra word was added in the recognized sentence

• How to determine the minimum error rate?
Measures of ASR Performance (cont.)

• Calculate the WER by aligning the correct word string against the recognized word string
  – A maximum substring matching problem
  – Can be handled by dynamic programming

• Example:
  Correct: “the effect is clear”
  Recognized: “effect is not clear”
  – Error analysis: one deletion and one insertion
  – Measures: word error rate (WER), word correction rate (WCR), word accuracy rate (WAR)

<table>
<thead>
<tr>
<th></th>
<th>Sub. + Del. + Ins. words</th>
<th>Matched words</th>
<th>Matched - Ins. words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word Error Rate = 100%</td>
<td></td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>No. of words in the correct sentence</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Word Correction Rate = 100%</td>
<td></td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>No. of words in the correct sentence</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Word Accuracy Rate = 100%</td>
<td></td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>No. of words in the correct sentence</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

 Might be higher than 100%
 Might be negative
Measures of ASR Performance (cont.)

• A Dynamic Programming Algorithm (Textbook)

Algorithm 9.1: Algorithm to Measure the Word Error Rate

Step 1: Initialization \( R[0,0] = 0 \) \( R[i,j] = \infty \) if \((i < 0)\) or \((j < 0)\) \( B[0,0] = 0 \)

Step 2: Iteration

\[
\text{for } i = 1, \ldots, n \{ \text{ //denotes for the word length of the correct/reference sentence} \\
\text{ for } j = 1, \ldots, m \{ \text{ //denotes for the word length of the recognized/test sentence} \\
R[i, j] = \min \left[ \begin{array}{c}
R[i-1, j] + 1 \text{ (deletion)} \\
R[i-1, j-1] \text{ (match)/hit} \\
R[i-1, j-1] + 1 \text{ (substitution)} \\
R[i, j-1] + 1 \text{ (insertion)}
\end{array} \right]
\]
\]

\[
B[i, j] = \begin{cases}
1 & \text{if deletion} \\
2 & \text{if insertion} \\
3 & \text{if match /hit} \\
4 & \text{if substitution}
\end{cases}
\]

Step 3: Backtracking and termination

\[
\text{word error rate} = 100\% \times \frac{R(n,m)}{n}
\]

optimal backward path = \((s_1, s_2, \ldots, 0)\)

where \( s_1 = B[n,m] \), \( s_t = \left[ \begin{array}{c}
B[i-1, j] \text{ if } s_{t-1} = 1 \\
B[i, j-1] \text{ if } s_{t-1} = 2 \\
B[i-1, j-1] \text{ if } s_{t-1} = 3 \text{ or } 4
\end{array} \right]
\]

for \( t = 2, \ldots \) until \( s_t = 0 \)
Measures of ASR Performance (cont.)

- **Algorithm (by Berlin Chen)**

**Step 1: Initialization:**

<table>
<thead>
<tr>
<th>Ref j</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

```
Step 1: Initialization:
G[0][0] = 0;
for i = 1,..., n  //test
  G[i][0] = G[i-1][0] + 1;
B[i][0] = 1;  //Insertion
}  (Horizontal Direction)
for j = 1,..., m  //reference
  G[0][j] = G[0][j-1] + 1;
B[0][j] = 2;  //Deletion
}  (Vertical Direction)
```

**Step 2: Iteration:**

```
Step 2: Iteration:
for i = 1,..., n  //test
  for j = 1,..., m  //reference
    G[i][j] = min
      [G[i-1][j] + 1 (Insertion)
      G[i][j-1] + 1 (Deletion)
      G[i-1][j-1] + 1 (if LR[j]!= LT[i], Substitution)
      G[i-1][j-1] (if LR[j] = LT[i], Match)
    ]
    B[i][j] =
      1;  //Insertion, (Horizontal Direction)
      2;  //Deletion , (Vertical Direction)
      3;  //Substitution (Diagonal Direction)
      4;  //match (Diagonal Direction)
  //for j, reference
//for i, test
```

**Step 3: Measure and Backtrace:**

```
Step 3: Measure and Backtrace:

Word Error Rate = 100% × \frac{G[n][m]}{m} 

Word Accuracy Rate = 100% – Word Error Rate 

Optimal backtrace path = (B[n][m] → ..... → B[0][0])
if B[i][j] = 1  print " LT[i]";  //Insertion, then go left
else if B[i][j] = 2 print " LR[j]";  //Deletion, then go down
else print " LR[j] LR[i]";  //Hit/Match or Substitution, then go down diagonally
```

*Note: the penalties for substitution, deletion and insertion errors are all set to be 1 here*
Measures of ASR Performance (cont.)

- A Dynamic Programming Algorithm
  - Initialization

```
for (j=1;j<=m;j++)
    { //reference
        grid[0][j] = grid[0][j-1];
        grid[0][j].dir = VERT;
        grid[0][j].score += DelPen;
        grid[0][j].del ++;
    }
```

```
grid[0][0].score = grid[0][0].ins
    = grid[0][0].del = 0;
grid[0][0].sub = grid[0][0].hit = 0;
grid[0][0].dir = NIL;
```

```
for (i=1;i<=n;i++) { //test
    grid[i][0] = grid[i-1][0];
    grid[i][0].dir = HOR;
    grid[i][0].score += InsPen;
    grid[i][0].ins ++;
}
```
• Program

```c
for (i=1;i<=n;i++) //test
    {  
        gridi = grid[i]; gridi1 = grid[i-1];
        for (j=1;j<=m;j++) //reference
            {  
                h = gridi[j].score + insPen;
                d = gridi1[j].score;
                if ((lRef[j] != lTest[i])
                    h += subPen;
                v = gridi[j-1].score + delPen;
                if (d<=h & & d<=v) /* DIAG = hit or sub */
                    {  
                        gridi[j].dir = DIAG;
                        gridi[j].score = d;
                        if (lRef[j] == lTest[i])  ++gridi[j].hit;else  ++gridi[j].sub;
                    }
                else if (h<v) /* HOR = ins */
                    {  
                        gridi[j].dir = HOR;
                        gridi[j].score = h;
                        ++ gridi[j].ins;
                    }
                else /* VERT = del */
                    {  
                        gridi[j].dir = VERT;
                        gridi[j].score = v;
                        ++gridi[j].del;
                    } /* for j */
            } /* for i */
```

**Example 1**

```
(0,0,0,0) Ins B (1,0,0,0) B A (2,0,0,0) C (3,0,0,0) (4,0,0,0)
A B C
```

**Alignment 1**: WER = 60%

- **Correct**: B A C B C C
- **Test**: B A C B C C

Still have an Other optimal alignment!
Measures of ASR Performance (cont.)

- Example 2

Note: the penalties for substitution, deletion and insertion errors are all set to be 1 here

(Ins, Del, Sub, Hit)

Alignment 1: WER = 80%

Correct: B A C B C C C
Test: B A A B A C C

Alignment 2: WER = 80%

Correct: A C B C C
Test: B A A A C

Alignment 3: WER = 80%

Correct: A C B C C
Test: B A A A C

Note: the penalties for substitution, deletion and insertion errors are all set to be 1 here

(Ins, Del, Sub, Hit)
Measures of ASR Performance (cont.)

- Two common settings of different penalties for substitution, deletion, and insertion errors

```c
/* HTK error penalties */
subPen = 10;
delPen = 7;
insPen = 7;

/* NIST error penalties*/
subPenNIST = 4;
delPenNIST = 3;
insPenNIST = 3;
```
Self-Exercise-1

- Measures of ASR Performance

<table>
<thead>
<tr>
<th>Reference</th>
<th>ASR Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>100000 100000 桃</td>
<td>100000 100000 桃</td>
</tr>
<tr>
<td>100000 100000 芝</td>
<td>100000 100000 芝</td>
</tr>
<tr>
<td>100000 100000 颱</td>
<td>100000 100000 颱</td>
</tr>
<tr>
<td>100000 100000 風</td>
<td>100000 100000 風</td>
</tr>
<tr>
<td>100000 100000 重</td>
<td>100000 100000 重</td>
</tr>
<tr>
<td>100000 100000 創</td>
<td>100000 100000 創</td>
</tr>
<tr>
<td>100000 100000 花</td>
<td>100000 100000 花</td>
</tr>
<tr>
<td>100000 100000 蓮</td>
<td>100000 100000 蓮</td>
</tr>
<tr>
<td>100000 100000 光</td>
<td>100000 100000 光</td>
</tr>
<tr>
<td>100000 100000 復</td>
<td>100000 100000 復</td>
</tr>
<tr>
<td>100000 100000 鄉</td>
<td>100000 100000 鄉</td>
</tr>
<tr>
<td>100000 100000 大</td>
<td>100000 100000 打</td>
</tr>
<tr>
<td>100000 100000 新</td>
<td>100000 100000 新</td>
</tr>
<tr>
<td>100000 100000 村</td>
<td>100000 100000 村</td>
</tr>
<tr>
<td>100000 100000 死</td>
<td>100000 100000 大</td>
</tr>
<tr>
<td>100000 100000 傷</td>
<td>100000 100000 傷</td>
</tr>
<tr>
<td>100000 100000 殘</td>
<td>100000 100000 殘</td>
</tr>
<tr>
<td>100000 100000 重</td>
<td>100000 100000 周</td>
</tr>
<tr>
<td>100000 100000 感</td>
<td>100000 100000 感</td>
</tr>
<tr>
<td>100000 100000 觸</td>
<td>100000 100000 觸</td>
</tr>
<tr>
<td>100000 100000 最</td>
<td>100000 100000 最</td>
</tr>
<tr>
<td>100000 100000 多</td>
<td>100000 100000 多</td>
</tr>
</tbody>
</table>

……
Self-Exercise-1 (count.)

- 506 BN stories of ASR outputs
  - Report the CER (character error rate) of the first one, 100, 200, and 506 stories
  - The result should show the number of substitution, deletion and insertion errors

<table>
<thead>
<tr>
<th>SENT: %Correct=0.00 [H=0, S=1, N=1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD: %Corr=81.52, Acc=81.52 [H=75, D=4, S=13, I=0, N=92]</td>
</tr>
<tr>
<td>=================================================================================</td>
</tr>
<tr>
<td>SENT: %Correct=0.00 [H=0, S=100, N=100]</td>
</tr>
<tr>
<td>WORD: %Corr=87.66, Acc=86.83 [H=10832, D=177, S=1348, I=102, N=12357]</td>
</tr>
<tr>
<td>=================================================================================</td>
</tr>
<tr>
<td>SENT: %Correct=0.00 [H=0, S=200, N=200]</td>
</tr>
<tr>
<td>WORD: %Corr=87.91, Acc=87.18 [H=22657, D=293, S=2824, I=186, N=25774]</td>
</tr>
<tr>
<td>=================================================================================</td>
</tr>
<tr>
<td>SENT: %Correct=0.00 [H=0, S=506, N=506]</td>
</tr>
<tr>
<td>WORD: %Corr=86.83, Acc=86.06 [H=57144, D=829, S=7839, I=504, N=65812]</td>
</tr>
<tr>
<td>=================================================================================</td>
</tr>
</tbody>
</table>
Symbols for Mathematical Operations

<table>
<thead>
<tr>
<th>Greek Letters</th>
<th>Mathematical Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>A α alpha</td>
<td>I iota</td>
</tr>
<tr>
<td>B β beta</td>
<td>K kappa</td>
</tr>
<tr>
<td>Γ γ gamma</td>
<td>Λ lambda</td>
</tr>
<tr>
<td>ϵ epsilon</td>
<td>Μ μ mu</td>
</tr>
<tr>
<td>Δ δ delta</td>
<td>Ν ν nu</td>
</tr>
<tr>
<td>Ζ ζ zeta</td>
<td>Ξ ξ xi</td>
</tr>
<tr>
<td>H η eta</td>
<td>O o omicron</td>
</tr>
<tr>
<td>Θ θ theta</td>
<td>Π π pi</td>
</tr>
<tr>
<td>P ρ rho</td>
<td>Σ σ sigma</td>
</tr>
<tr>
<td>T τ tau</td>
<td>Υ υ upsilon</td>
</tr>
<tr>
<td>Φ φ phi</td>
<td>Χ χ chi</td>
</tr>
<tr>
<td>Ψ ψ psi</td>
<td>Ω ω omega</td>
</tr>
</tbody>
</table>