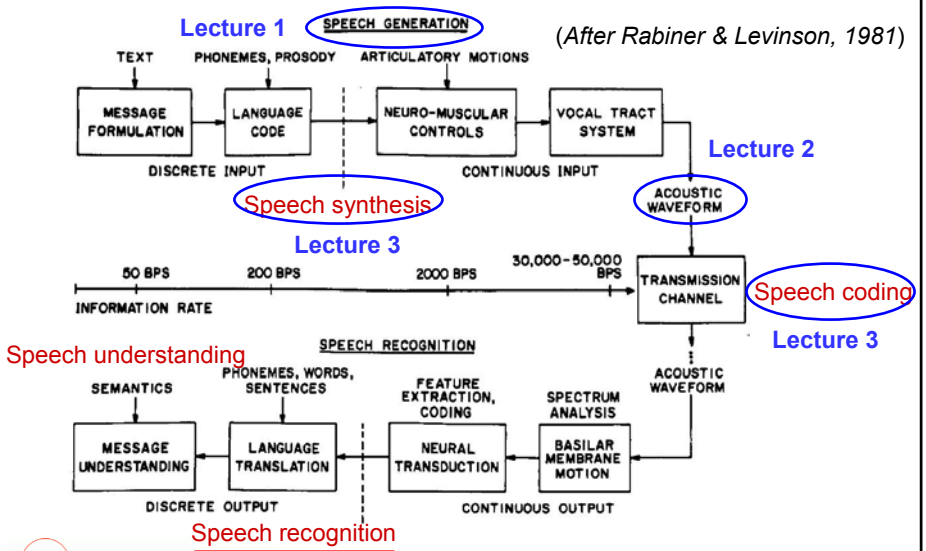
Speech Communication, Spring 2006

Lecture 4: Speech Recognition, Part I

Zheng-Hua Tan

Department of Communication Technology
Aalborg University, Denmark
zt@kom.aau.dk

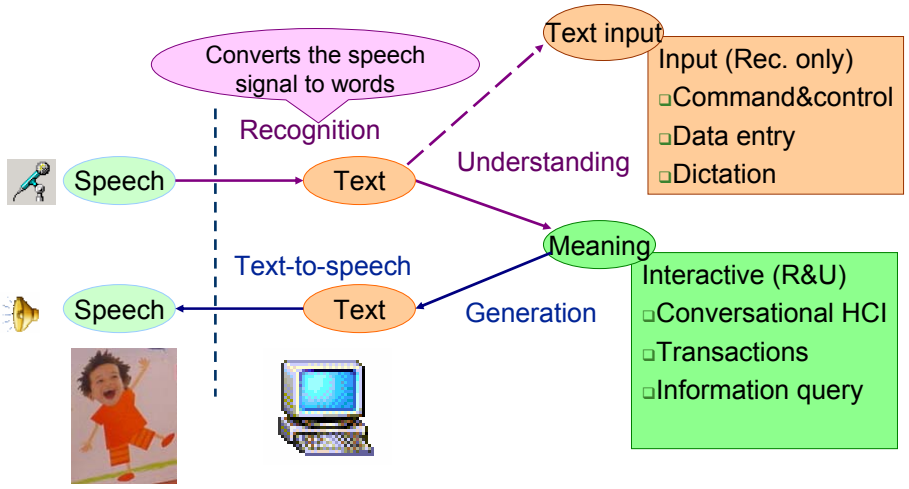
Human speech communication process



Part I: Automatic speech recognition

- Automatic speech recognition (ASR)
- Dynamic programming
- Hidden Markov model

Human-computer interaction via speech



Attributes of ASR systems

- Vocabulary: small (<20 words) to large (>50K words)
- Perplexity: small (< 10) to large (> 200)
- Enrollment: speaker-dependent to speaker-independent
- Speaking mode: isolated-word to continuous-speech
- Speaking style: read speech to spontaneous speech
- SNR: high (> 30 dB) to low (< 10 dB)
- Transducer: noise-concelling microphone to cell phone

Where are we?

Word error rates for several ASR tasks

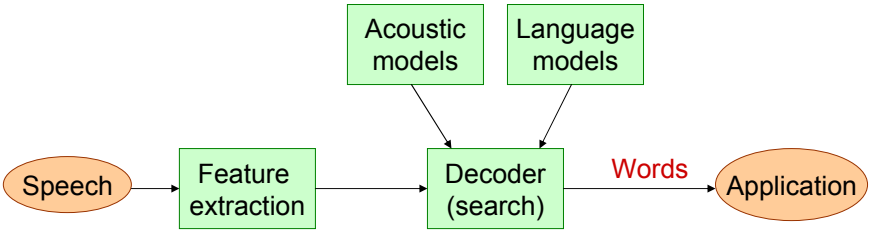
Corpus	Speaking style	Vocabulary size	Word error rate (%)	Huamn error rate (%)
Connected digit strings	Spontaneous	11	0.3	0.009
Resource management	Read speech	1000	3.6	0.1
Wall Street Journal	Read text	64,000	6.6	1
Switchboard	Conversational telephone	10,000	19.3	4

ASR Trends

	~ mid 70's	mid 70's ~mid 80's	mid 80's ~
Recognition units	Word & sub-word	Sub-word	Sub-word
Modelling appraoches	Heuristic; Rule-based	Template matching; Deterministic & data-driven	Mathematical; Probabilistic & data-driven
Knowledge representation	Heterogeneous and complex	Homogeneous and simple	Homogeneous and simple
Knowledge acquisition	Knowledge engineering	Embedded in simple structure	Automatic learning

- Statistical modelling and data-driven approaches have proved to be powerful

Components of ASR system



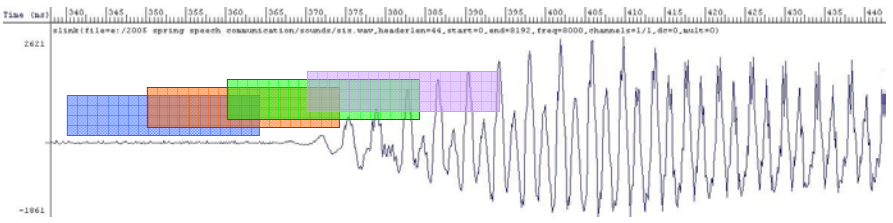
- Speech recognition involves:
  - How to represent the signal
  - How to model both acoustic and language constraints
  - How to search for the optimal answer

Part II: Dynamic programming

- Speech recognition
- Dynamic programming
- Hidden Markov model

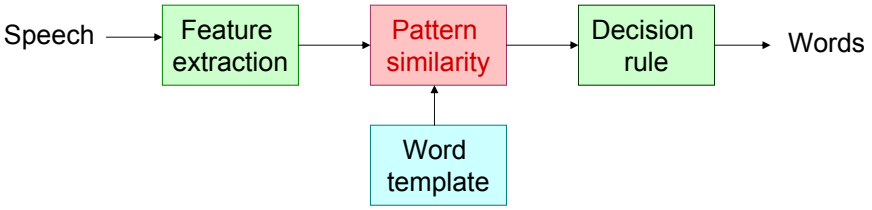
Dimension & speech representation

- The curse of dimension – the computational cost increases exponentially with the dimension of the problem
- The frame-based analysis yields a sequence as a new representation of the speech signal
  - samples at 8000/sec → vectors at 100/sec



Word template based ASR

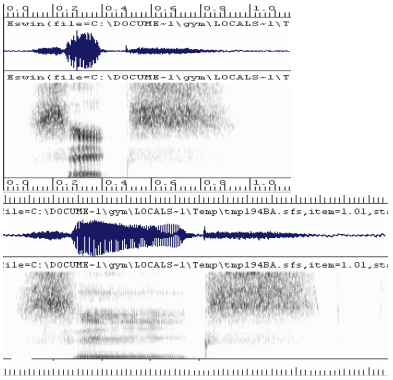
- Used for both isolated- and connected-word speech recognition



- Template matching mechanism
  - Calculate the distance btw two patterns
  - Dynamic time warping (DTW)

Speaking rate and time-normalization

- Speaking rate variation causes nonlinear fluctuation in a speech pattern time axis



- Time-normalization is needed.

DP based time-normalization

- Dynamic programming is a pattern matching algorithm with a nonlinear time-normalization effect.
  - Time differences btw two speech patterns are eliminated by warping the time axis of one so that the maximum coincidence is attained with the other, also called dynamic time warping (DTW)
  - The time-normalized distance is calculated as the minimized residual distance between them, remaining still after eliminating the timing differences.

Dynamic programming

- Consider two speech patterns expressed as a sequence of feature vectors :

$$A = a_1, a_2, ..., a_i, ..., a_I$$

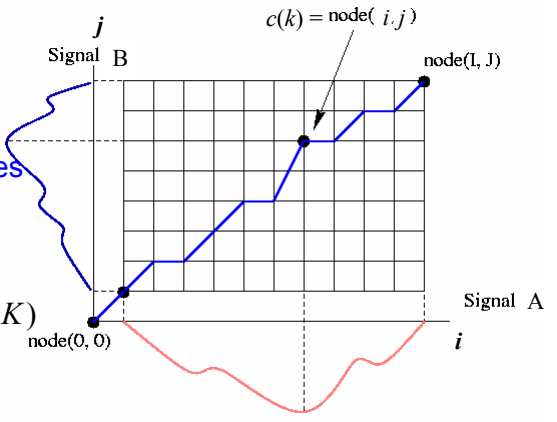
$$B = b_1, b_2, ..., b_j, ..., b_J$$

- Consider an *i-j* plane, then the time differences can be depicted by a sequence of points *c*=(*i*,*j*):

$$F = c(1), c(2), ..., c(k), ..., c(K)$$

where

$$c(k) = (i(k), j(k))$$



Dynamic programming (cont'd)

- The sequence *c* is called a warping function.
- A distance btw two feature vectors is

$$d(c) = d(i, j) = \| a_i - b_j \|$$

- The weighted summation of distances on warping function *F* becomes

$$E(F) = \sum_{k=1}^{K} d(c(k)).w(k)$$

- The time-normalized distance btw *A* and *B* is defined as the minimum residual distance btw them

$$D(A, B) = \min \left[\frac{\sum_{k=1}^{K} d(c(k)).w(k)}{\sum_{k=1}^{K} w(k)} \right]$$

Restrictions on warping function

- Warping function F (or points *c*(*k*)), as a model of time-axis fluctuation in speech, has restrictions:

1) Monotonic conditions :

$$i(k-1) \le i(k) \text{ and } j(k-1) \le j(k)$$

2) Continuity conditions :

$$i(k) - i(k-1) \le 1 \text{ and } j(k) - j(k-1) \le 1$$

3 Boundary conditions :

$$i(1) = 1, j(1) = 1 \text{ and } i(K) = I, j(K) = J.$$

4) Adjustment window condition

$$| i(k) - j(k) | \le r$$

5) Slope constraint condition :

A gradient should be neither too steep nor too gentle.

The simplest DP of symmetric form

- Step 1: Initialisation:

$$g(1,1) = 2d(1,1)$$

- Step 2: Iteration (DP equation):

$$g(i,j) = \min \begin{bmatrix} g(i,j-1) + d(i,j) \\ g(i-1,j-1) + 2d(i,j) \\ g(i-1,j) + d(i,j) \end{bmatrix}$$

Adjustment window: $j - r \le i \le j + r$

- Step 3: Termination:

Time-normalised distance

$$D(A,B) = \frac{1}{N} g(I,J), \quad \text{where } N = I + J$$

Part III: Hidden Markov model

- Speech recognition
- Dynamic programming
- Hidden Markov model
  - Markov chain
  - Hidden Markov model
  - Difference between an observable and a hidden Markov model
  - Basic calculations
    - How to evaluate an HMM – the forward algorithm
    - How to decode an HMM – the Viterbi algorithm
    - How to estimate HMM parameters – Baum-Welch Algorithm

From template to statistical method

- The template method with DP alignment is a simplified, non-parametric method which is hard to characterise the variation among utterances
- Hidden Markov model (HMM) is a powerful statistical method of characterising the observed data samples of a discrete-time series
- The underlying assumption of the HMM is
  - The speech signal can be well characterised as a parametric random process
  - The parameters of the stochastic process can be estimated in a precise, well-defined manner

The Markov chain

- Consider a system described at any time *t* as being in one of a set of *N* states indexed by {1,2,…,*N*}
- Denote the actual state at time *t* as q_t
- So, the first-order Markov chain is

$$P[q_t = j \mid q_{t-1} = i, q_{t-2} = k, ...] = P[q_t = j \mid q_{t-1} = i]$$

- We consider those processes in which the right-side of the equation above is independent of time, leading to state-transition probabilities

$$a_{ij} = P[q_t = j \mid q_{t-1} = i], \quad 1 \le i, j \le N$$

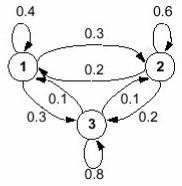
with constraints:

$$a_{ij} \ge 0 \qquad \forall j, i$$

$$\sum_{j=1}^{N} a_{ij} = 1 \qquad \forall i$$

Example: A three-state model of the weather

State 1: precipitation (rain, snow, hail, etc.)
State 2: cloudy
State 3: sunny

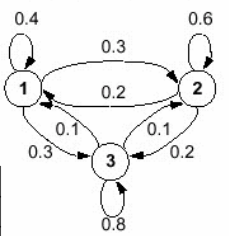


A three-state Markov model

- The weather on day *t* is characterised by a single one of the three states

Example:A three-state model of the weather

State 1: precipitation (rain, snow, hail, etc.)
State 2: cloudy
State 3: sunny



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

- The above stochastic process is considered an observable Markov model since the output process is a set of states at each instant of time, where each state corresponds to an observable event.

Basic calculations

Example: What is the probability that the weather for eight consecutive days is "sun-sun-sun-rain-rain-sun-cloudy-sun"?

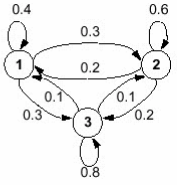
Solution:

O =	sun	sun	sun	rain	rain	sun	cloudy	sun
	3	3	3	1	1	3	2	3

$$\begin{aligned} P(\bar{O}|Model) &= P[3]P[3|3]P[3|3]P[1|3]P[1|1]P[3|1]P[2|3]P[3|2] \\ &= \pi_3 a_{33} a_{31} a_{11} a_{13} a_{32} a_{23} \\ &= 1.536 \times 10^{-4} \end{aligned}$$

Example:A three-state model of the weather

State 1: precipitation (rain, snow, hail, etc.)
State 2: cloudy
State 3: sunny



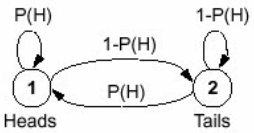
(*After Joseph Picone*)

"Hidden" Markov model

Consider the problem of predicting the outcome of a coin toss experiment. You observe the following sequence:

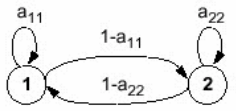
$$\bar{O} = (HHTTTHTTH\ldots H)$$

What is a reasonable model of the system?



1-Coin Model
(Observable Markov Model)

O = H H T T H T H H T T H ...
S = 1 1 2 2 1 2 1 1 2 2 1 ...



2-Coins Model
(Hidden Markov Model)

O = H H T T H T H H T T H ...
S = 2 1 1 2 2 2 1 2 2 1 2 ...

$P(H) = P_1$ $P(H) = P_2$
$P(T) = 1\text{-}P_1$ $P(T) = 1\text{-}P_2$

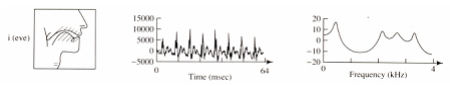
The Urn-and-Ball model

The Urn-and-Ball Model — doubly stochastic systems

P(red) = $b_1(1)$	P(red) = $b_2(1)$	P(red) = $b_3(1)$
P(green) = $b_1(2)$	P(green) = $b_2(2)$	P(green) = $b_3(2)$
P(blue) = $b_1(3)$	P(blue) = $b_2(3)$	P(blue) = $b_3(3)$
P(yellow) = $b_1(4)$	P(yellow) = $b_2(4)$	P(yellow) = $b_3(4)$
...	...	...

$\hat{O}$ = {green, blue, green, yellow, red, ..., blue}

How can we determine the appropriate model for the observation sequence given the system above?



Elements of a discrete HMM

- N: the number of states
  - states, $s = \{s_1, s_2, ..., s_N\}$
  - state at time t, $q_t \in s$
- M: the number of observation symbols
  - observation symbols, $v = \{v_1, v_2, ..., v_M\}$
  - observation at time t, $o_t \in v$
- $A = \{a_{ij}\}$: state transition probability distribution
  - $a_{ij} = P(q_{t+1} = s_j \mid q_t = s_i)$, $1 \le i,j \le N$
- $B = \{b_j(k)\}$: observation probability distribution in state j
  - $b_j(k) = P(O_t = v_k \mid q_t = s_j)$, $1 \le j \le N$, $1 \le k \le M$
- $\pi = \{\pi_i\}$: initial state distribution
- For convenience, we use the notation: $\lambda = (A, B, \pi)$

Three basic HMM problems

1. **Scoring:** Given an observation sequence $\boldsymbol{O} = \{o_1, o_2, ..., o_T\}$ and a model $\lambda = \{\boldsymbol{A}, \boldsymbol{B}, \pi\}$, how to compute $P(\boldsymbol{O} \mid \lambda)$, the probability of the observation sequence? → The Forward-Backward Algorithm
2. **Matching:** Given an observation sequence $\boldsymbol{O} = \{o_1, o_2, ..., o_T\}$ and the model λ,how to choose a state sequence $\boldsymbol{q} = \{q_1, q_2, ..., q_T\}$ which is optimum in some sense? → The Viterbi Algorithm
3. **Training:** How to adjust the model parameters $\lambda = \{A, B, \pi\}$ to maximize $P(\boldsymbol{O} \mid \lambda)$? → The Baum-Welch Re-estimation Procedures

Problem 1: Scoring

- Given $\boldsymbol{O} = \{o_1, o_2, \ldots, o_T\}$ and $\lambda = \{\boldsymbol{A}, \boldsymbol{B}, \pi\}$, how to compute $P(\boldsymbol{O} \mid \lambda)$, the probability of the observation sequence? (probability evaluation)
  - Consider all possible state sequences (N^T) of length T:

$$P(O \mid \lambda) = \sum_{all\ \mathbf{q}} P(O \mid q, \lambda) P(q \mid \lambda)$$
$$= \sum_{q_1, q_2, \ldots, q_T} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \ldots a_{q_{T-1} q_T} b_{q_T}(O_T)$$

- Calculation required $\approx 2T \cdot N^T$
  - For N =5,T = 100, $2 \cdot 100 \cdot 5^{100} \approx 10^{72}$ computations!

The forward algorithm

- Consider the forward variable $\alpha_t(i)$ defined as

$$\alpha_t(i) = P(o_1 o_2 \ldots o_t, q_t = i \mid \lambda)$$

i.e., the probability of the partial observation sequence until time *t* and state *i* at time *t*, given the model λ

- We can solve for $\alpha_t(i)$ inductively as follows:

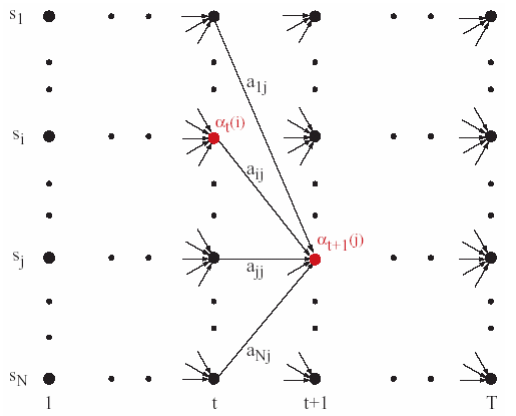
1. Initialisation $\alpha_1(i) = \pi_i b_i(o_1), \quad 1 \le i \le N$

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

3. Termination $P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$

- Calculation$\approx N^2 \cdot T$. For N=5,T=100, 2500, instead of 10^{72}

Illustration of forward algorithm



The backward algorithm

- Similarly, consider the backward variable $\beta_t(i)$ defined as

$$\beta_t(i) = P(o_{t+1}o_{t+2}...o_T \mid q_t = i, \lambda)$$

i.e., the probability of the partial observation sequence from time *t* +1 to the end, given state *i* at time *t* and model *λ*

- We can solve for $\beta_t(i)$ inductively as follows:

1. Initialisation

$$\beta_T(i) = 1, \qquad 1 \le i \le N$$

2. Induction

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \qquad \begin{matrix} t = T-1, T-2, ..., 1 \\ 1 \le i \le N \end{matrix}$$

3. Termination

$$P(O \mid \lambda) = \sum_{i=1}^{N} \pi_i b_i(o_1) \beta_1(i)$$

- Again, calculation≈$N^2 \cdot T$.

Problem 2: Matching

- Given $\boldsymbol{O} = \{o_1, o_2, \ldots, o_T\}$, how to choose a state sequence $\boldsymbol{q} = \{q_1, q_2, \ldots, q_T\}$ which is optimum in some sense? ("Optimal" state sequence)
- Several possible optimality criteria:
  - Choose the states q_i that are individually most likely at each time t, which maximize the expected number of correct individual states (by choosing the most likely state for each t). We define the a posteriori probability variable

$$\gamma_t(i) = P(q_t = i \mid O, \lambda)$$

  i.e., the probability of being in state i at time t, given the observation sequence O, and the model λ

Finding optimal state sequence

$$\gamma_t(i) = P(q_t = i \mid O, \lambda)$$

$$= \frac{P(O, q_t = i \mid \lambda)}{P(O \mid \lambda)} \quad = \frac{P(O, q_t = i \mid \lambda)}{\sum_{i=1}^{N} P(O, q_t = i \mid \lambda)}$$

Since $P(O, q_t = i \mid \lambda) = \alpha_t(i)\beta_t(i)$

So $$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}$$

Then the individually most likely state q_t^* at time t is

$$q_t^* = \arg \min_{1 \le i \le N} [\gamma_t(i)]$$

The Viterbi algorithm

- The individual optimality criterion has the problem that the "optimal" state sequence may not even be a valid state sequence →
- Another criterion is to find the single best state sequence (path), i.e., to maximize $P(q, O|\lambda)$.→
- A formal technique to do so, based on DP methods, is called the Viterbi algorithm
- To find the best path $\boldsymbol{q} = \{q_1,q_2,...,q_T\}$, for given $\boldsymbol{O} = \{o_1,o_2,...,o_T\}$, we define the best score (highest probability) along a single path, at time t,

$$\delta_t(i) = \max_{q_1,q_2,...,q_{t-1}} P(q_1 q_2 ... q_{t-1}, q_t = i, o_1 o_2 ... o_t \mid \lambda)$$

which accounts for the first t observations and ends in state *i*.

Then $\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}].b_j(o_{t+1})$

The Viterbi algorithm (cont'd)

1. Initialisation $\delta_1(i) = \pi_i b_i(o_1), \quad 1 \le i \le N$

   $\psi_1(i) = 0$

2. Recursion

   $\delta_t(i) = \max_{1\le i\le N}[\delta_{t-1}(i) a_{ij} b_i(o_t)], \quad 2 \le t \le T \quad 1 \le j \le N$

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

3. Termination

   $P^* = \max_{1\le i\le N}[\delta_T(i)]$

   $q_T^* = \arg\max_{1\le i\le N}[\delta_T(i)]$

4. Path (state sequence) backtracking

   $q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, ..., 1$

The power of recursive equation

Computing factorials $n!$

1. simply caculate $n!$ for each n
2. use $n! = n(n-1)!$

if $F(n) = n!$ then

$F(n) = nF(n-1)$ for $n \geq 1$ Recursive Equation

Problem 3: Training

- How to adjust the model parameters $\lambda = \{A, B, \pi\}$ to maximize $P(\boldsymbol{O}|\lambda)$? → The Baum-Welch Re-estimation Procedures

(next lecture)

Summary

- Introduction to speech recognition
- Dynamic programming
- Hidden Markov model

- Next lectures: Speech Recognition, Part II