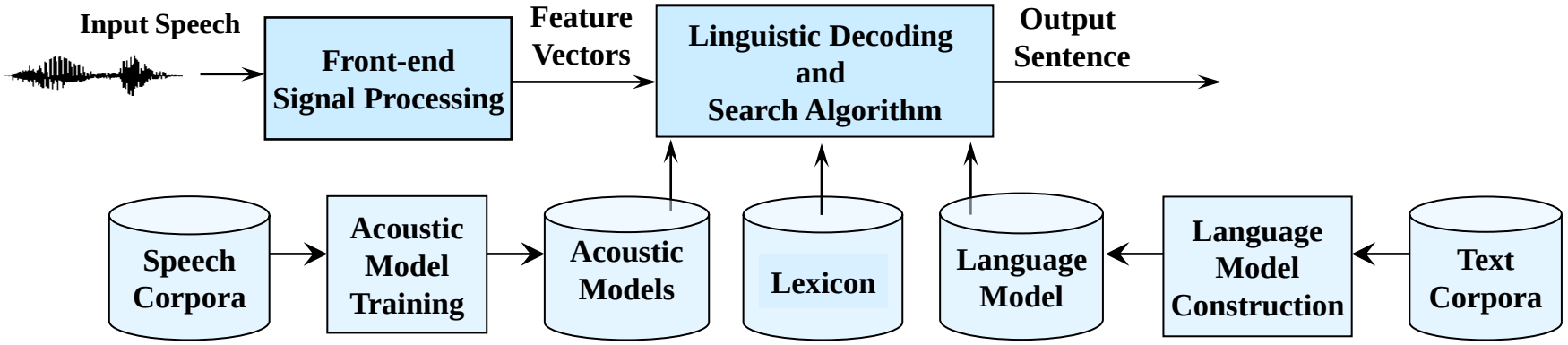
8.0 Search Algorithms for Speech Recognition

References: 1. 12.1-12.5 of Huang, or

2. 7.2-7.6 of Becchetti, or
3. 5.1-5.7, 6.1-6.5 of Jelinek
4. " Progress in Dynamic Programming Search for LVCSR (Large Vocabulary Continuous Speech Recognition)", Proceedings of the IEEE, Aug 2000

Basic Approach for Large Vocabulary Speech Recognition

- **A Simplified Block Diagram**



- **Example Input Sentence**

  this is speech

- **Acoustic Models**

  (th-ih-s-ih-z-s-p-ih-ch)

- **Lexicon** (th-ih-s) → this

  (ih-z) → is

  (s-p-iy-ch) → speech

- **Language Model** (this) – (is) – (speech)

  P(this) P(is | this) P(speech | this is)

  $P(w_i|w_{i-1})$ bi-gram language model

  $P(w_i|w_{i-1},w_{i-2})$ tri-gram language model,etc

DTW and Dynamic Programming

- **Dynamic Time Warping (DTW)**
  - well accepted pre-HMM approach
  - find an optimal path for matching two templates with different length
  - good for small-vocabulary isolated-word recognition even today
- **Test Template [y_j, j=1,2,...N] and Reference Template [x_i, i=1,2,..M]**
  - warping both templates to a common length L

    warping functions: $f_x(m)= i$, $f_y(m)= j$, m=1,2,...L
  - endpoint constraints: $f_x(1)= f_y(1)= 1$, $f_x(L)=M$, $f_y(L)= N$

    monotonic constraints: $f_x(m+1) \geq f_x(m)$, $f_y(m+1) \geq f_y(m)$
  - matching pair: $x_{f_x(m)} \leftrightarrow y_{f_y(m)}$ for every m, m: index for matching pairs
  - recursive relationship:

$$D(i, j)=\min_{(i', j')}\{D(i', j') + \bar{d}[(i', j'); (i, j)]\}, \quad \text{D(i, j) : accumulated minimum distance up to (i, j)}$$

$$\bar{d}[(i', j'); (i, j)]: \text{additional distance extending } (i', j') \text{ to } (i, j)$$

$$d(i, j) = \text{distance measure for } x_i \text{ and } y_j$$

$$\text{examples: } \bar{d}[(i-1, j-1); (i, j)] = d(i, j)$$

$$\bar{d}[(i-k, j-1); (i, j)] = \frac{1}{k}[d(i-k+1, j) + \cdots + d(i-1, j) + d(i, j)]$$

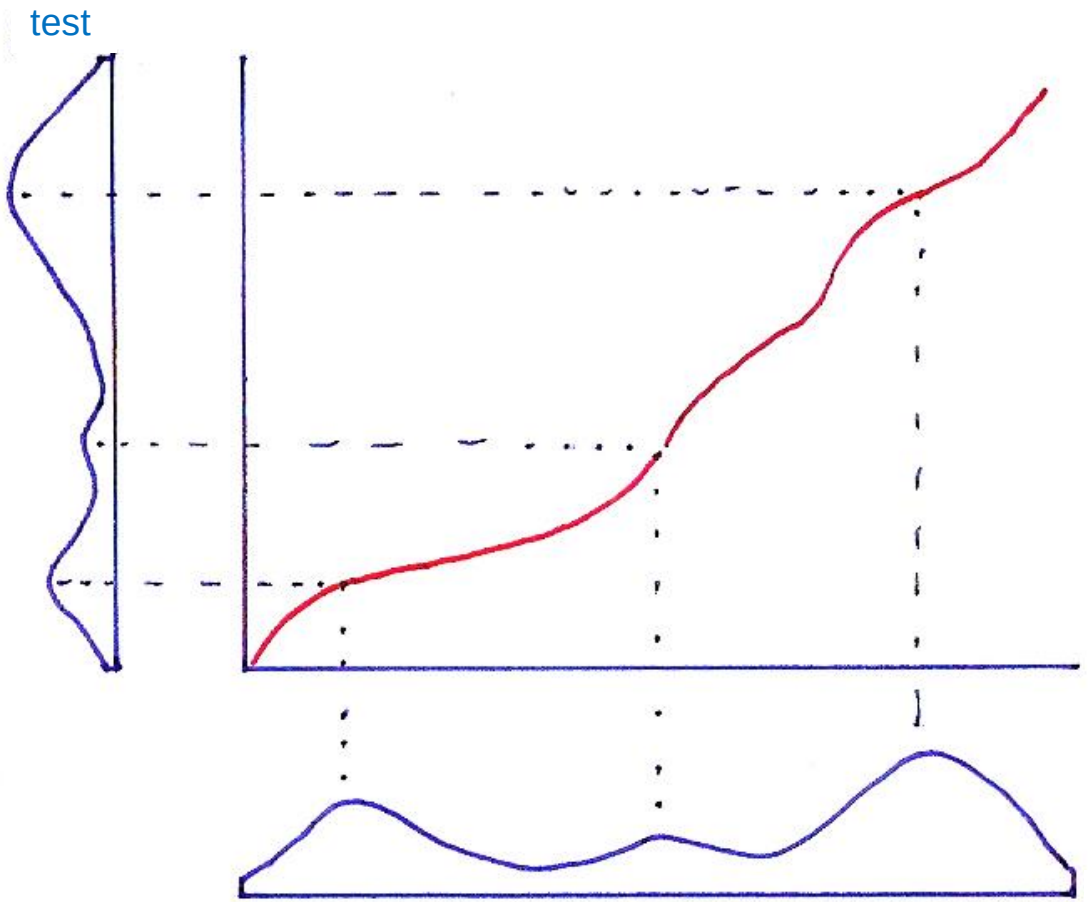
  - global constraints/local constraints
  - lack of a good approach to train a good reference pattern
- **Dynamic Programming**
  - replacing the problem by a smaller sub-problem and formulating an iterative procedure
- **Reference: 4.7 up to 4.7.3 of Rabiner and Juang**

DTW

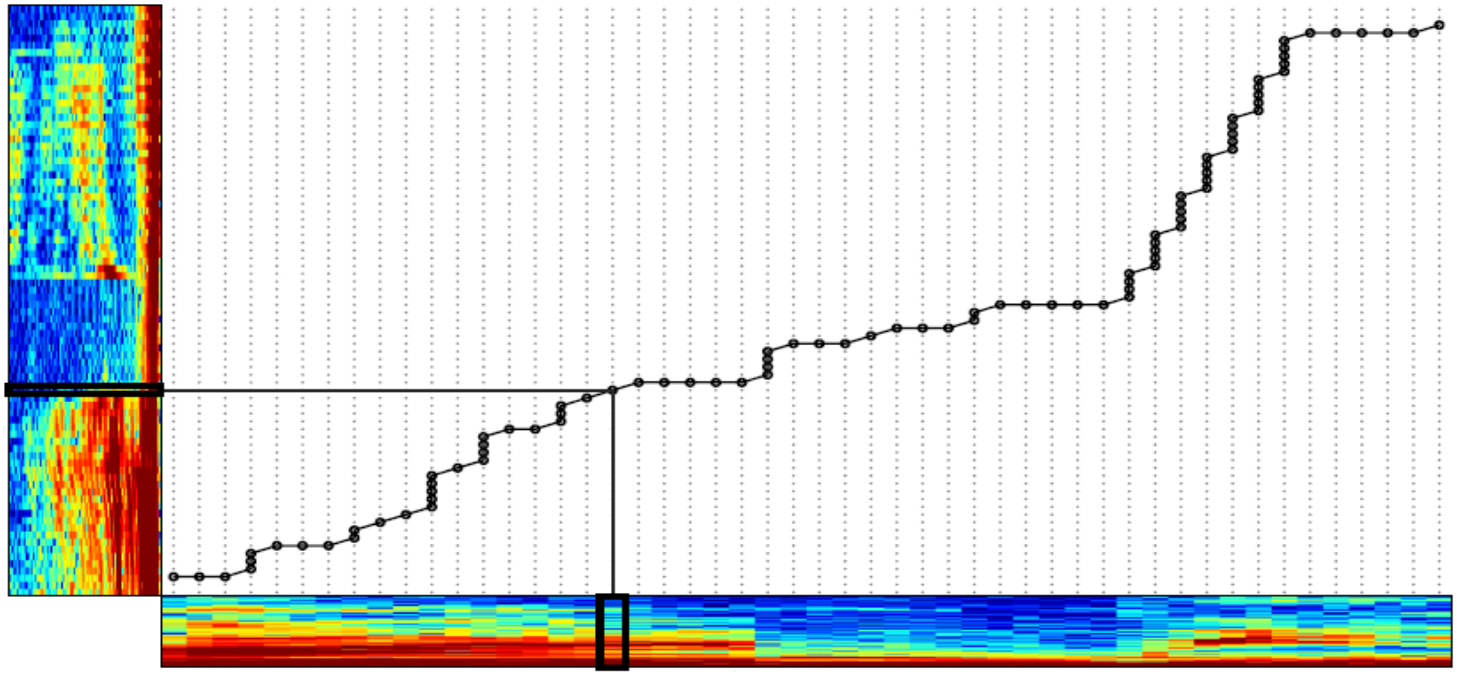
programming

(reference)

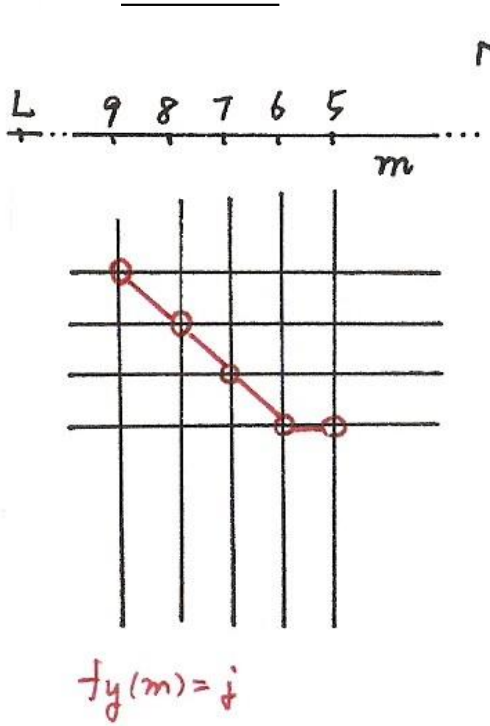
(test)

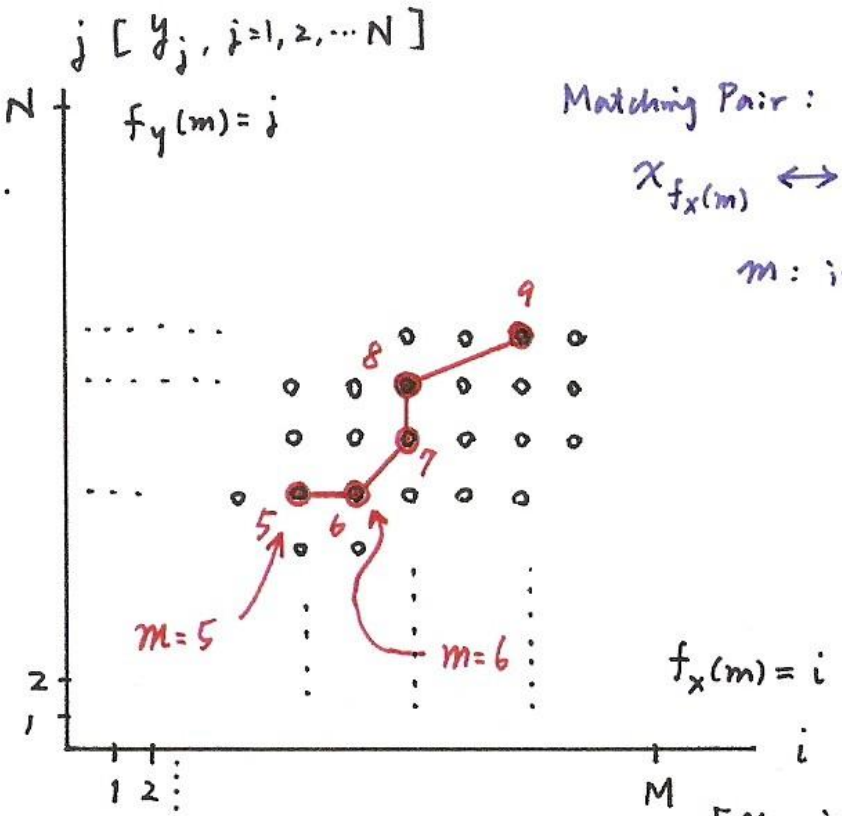


DTW



DTW

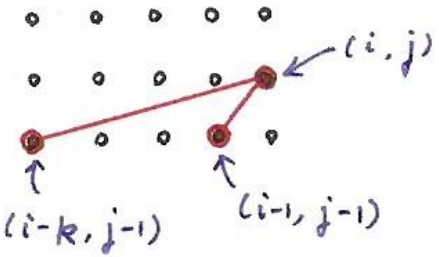




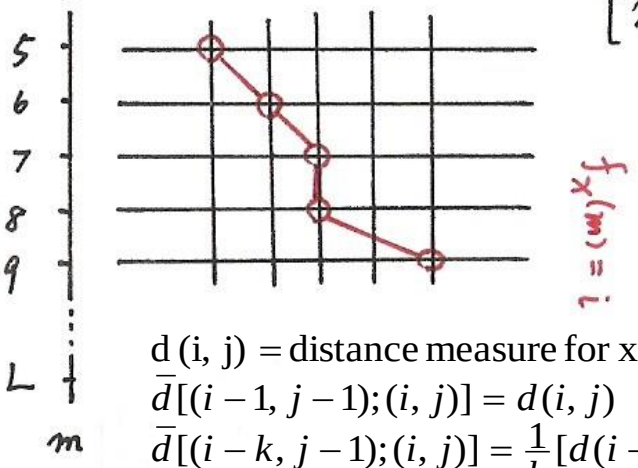
Matching Pair :

$x_{f_x(m)} \leftrightarrow y_{f_y(m)}$ for every m

m : index for matching pairs



$[x_i, i=1,2,\cdots M]$



$f_x(m) = i$, $f_y(m) = j$, $m = 1,2,\ldots L$

$f_x(1) = f_y(1) = 1$, $f_x(L) = M$, $f_y(L) = N$

$f_x(m+1) \geq f_x(m)$, $f_y(m+1) \geq f_y(m)$

$$D(i, j) = \min_{(i', j')} \{D(i', j') + \bar{d}[(i', j'); (i, j)]\},$$

D(i, j) : accumulated minimum distance up to (i, j)

$\bar{d}[(i', j'); (i, j)]$: additional distance extending (i′, j′) to (i, j)

d (i, j) = distance measure for x_i and y_j

$$\bar{d}[(i-1, j-1); (i, j)] = d(i, j)$$

$$\bar{d}[(i-k, j-1); (i, j)] = \frac{1}{k}[d(i-k+1, j) + \cdots + d(i-1, j) + d(i, j)]$$

Basic Problem 2 for HMM (P.20 of 4.0)

- **Approach 2 —Viterbi Algorithm - finding the single best sequence $\bar{q}^* = q_1^* q_2^* \ldots q_T^*$**
  - Define a new variable $\delta_t(i)$

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

    = the highest probability along a certain single path ending at state i at time t for the first t observations, given λ

  - Induction

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

  - Backtracking

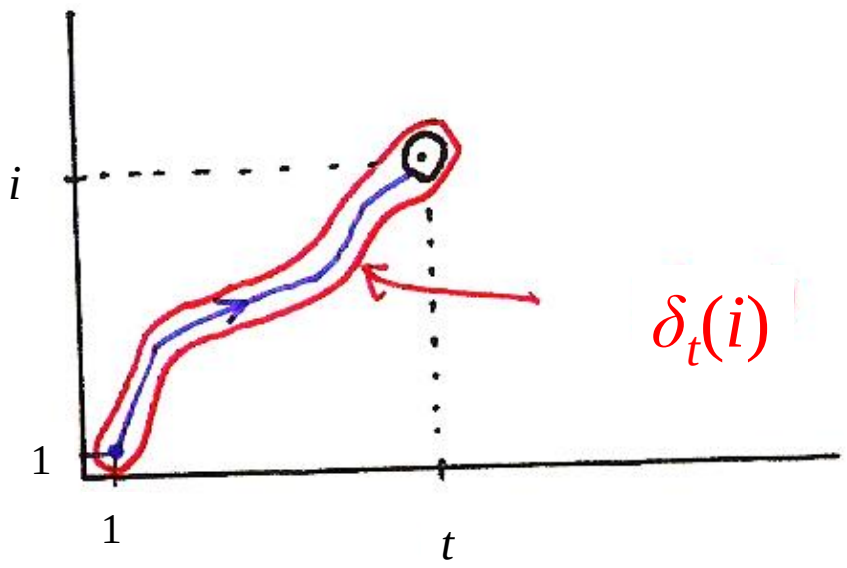
    $\psi_t(j) = \arg \max_{1 \le i \le N} [\delta_{t-1}(i)a_{ij}]$

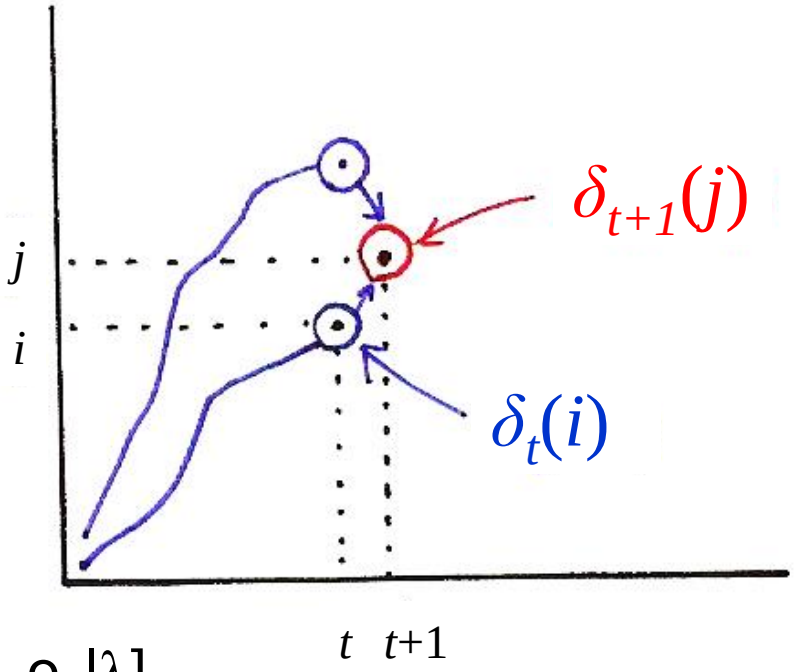
    the best previous state at t–1 given at state j at time t

    keeping track of the best previous state for each j and t

Viterbi Algorithm (P.21 of 4.0)

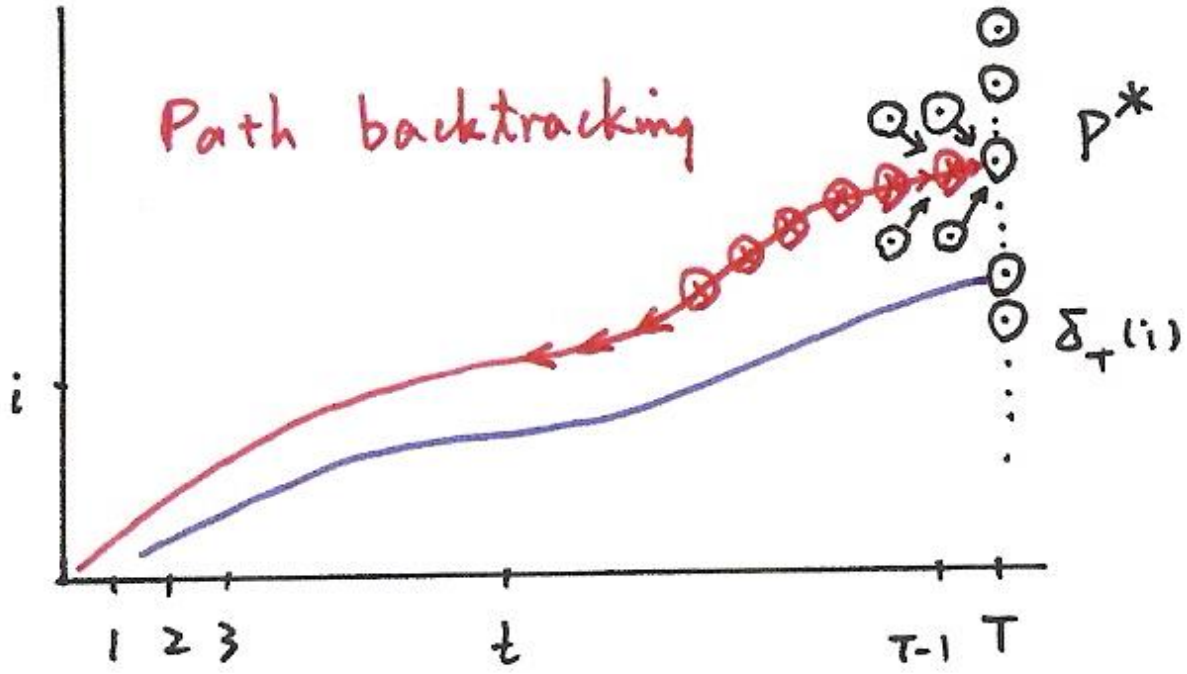
$$\delta_{t+1}(j) = \max_i [\delta_t(i) a_{ij}] \bullet b_j(o_{t+1})$$





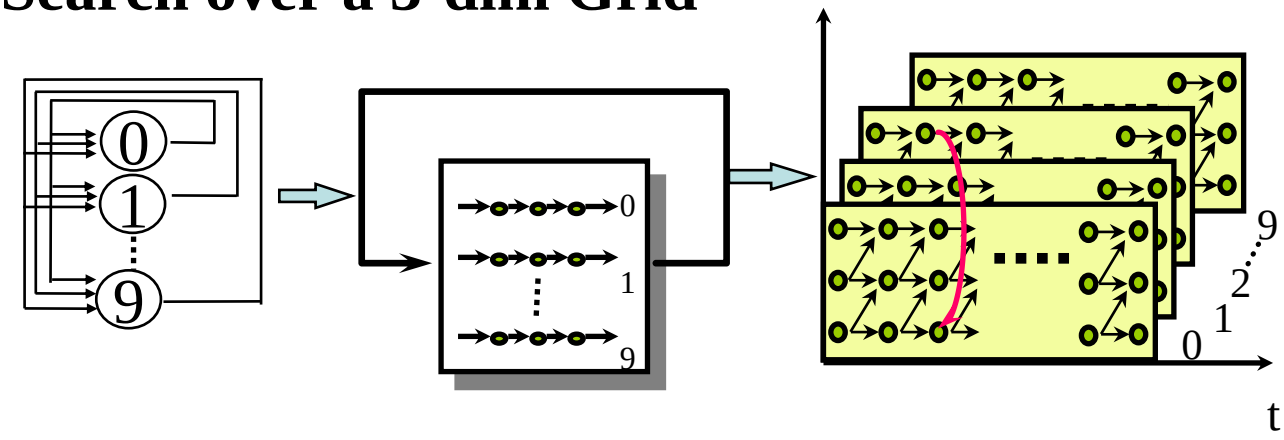
$$\delta_t(i) = \max_{q_1,q_2,\ldots q_{t-1}} P[q_1,q_2,\ldots q_{t-1}, q_t = i, o_1,o_2,\ldots,o_t \,|\lambda]$$

Viterbi Algorithm (P.22 of 4.0)

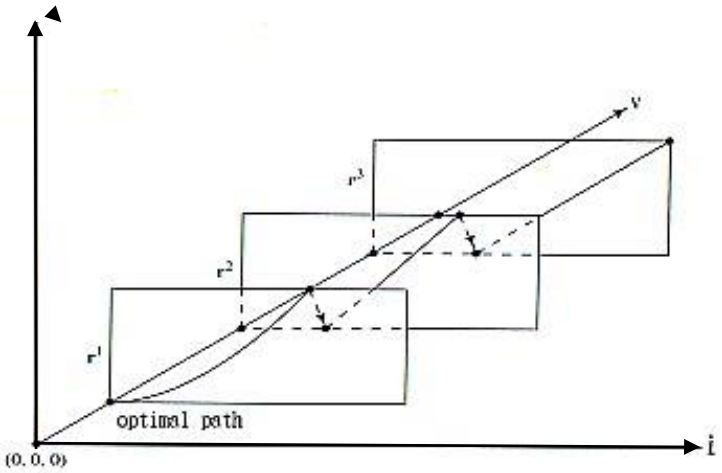


Continuous Speech Recognition Example: Digit String Recognition— One-stage Search

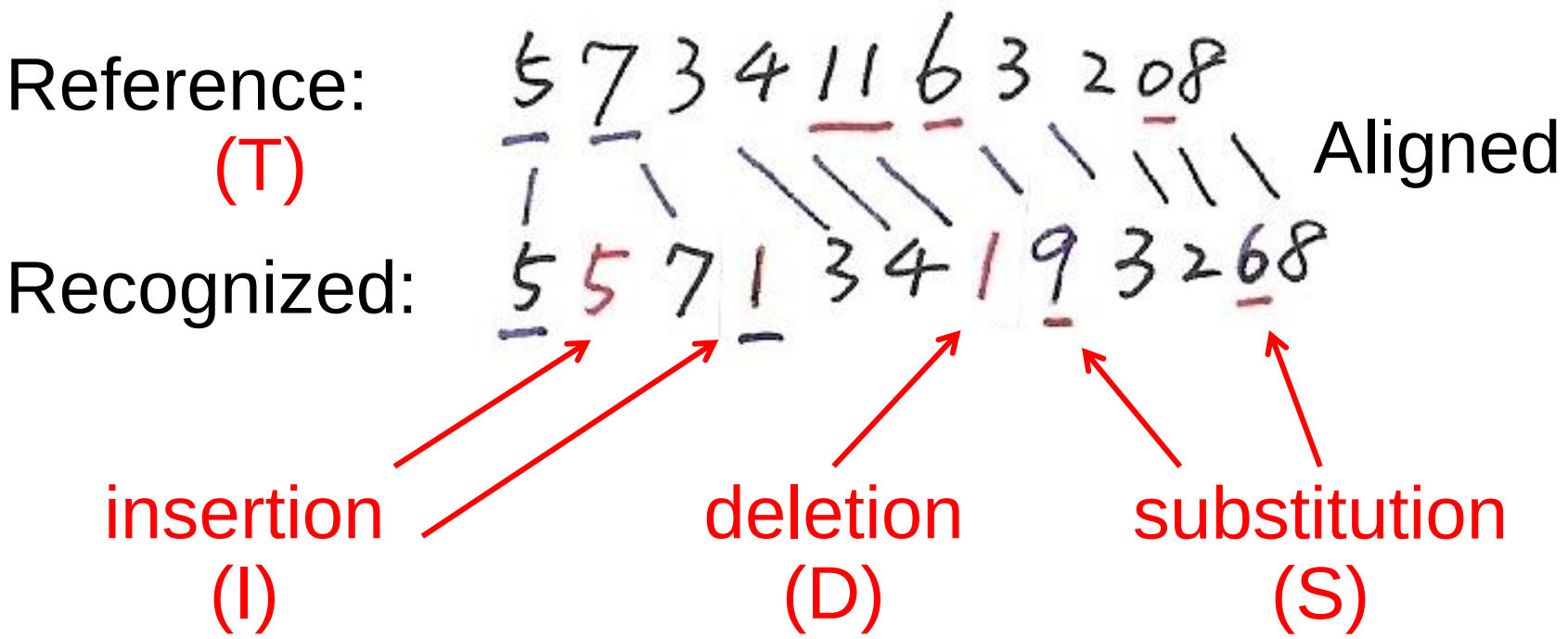
- Unknown Number of Digits
- No Lexicon/Language Model Constraints
- Search over a 3-dim Grid



- Switched to the First State of the Next Model at the End of the Previous Model
- May Result with Substitution, Deletion and Insertion



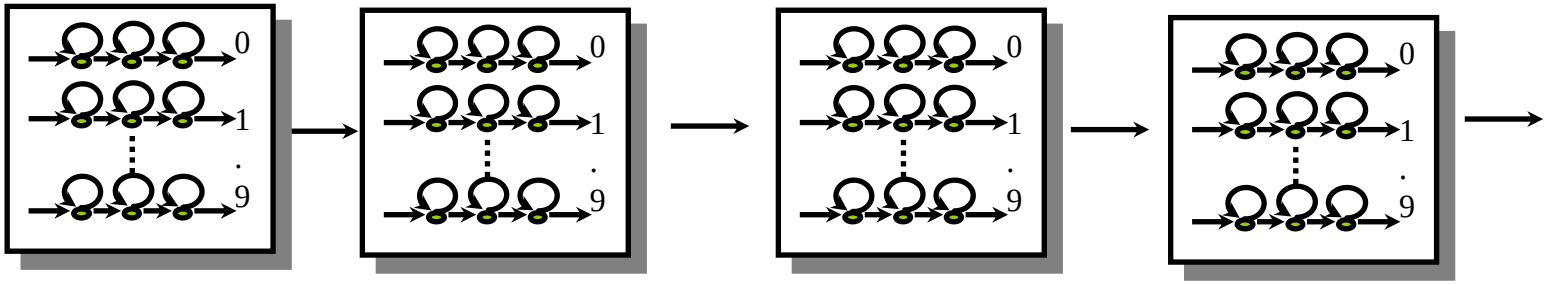
Recognition Errors

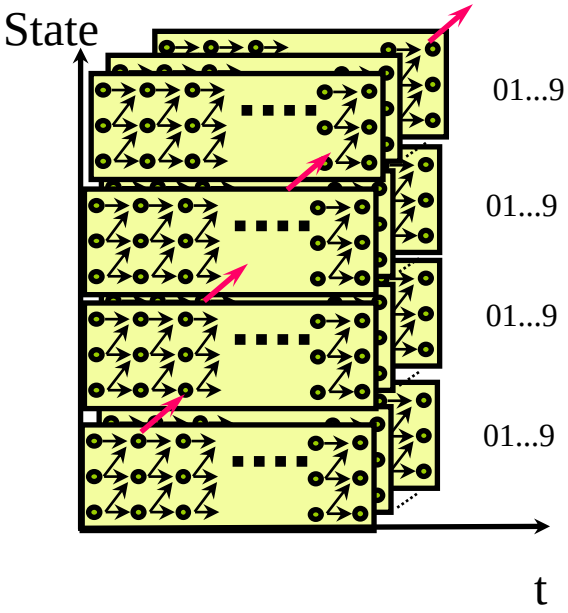


$$\frac{T - D - S - I}{T} \times 100\% = \text{Accuracy}$$

Continuous Speech Recognition Example: Digit String Recognition — Level-Building

- **Known Number of Digits**
- **No Lexicon/Language Model Constraints**
- **Higher Computation Complexity, No Deletion/Insertion**





– number of levels = number of digits in an utterance
– automatic transition from the last state of the previous model to the first state of the next model

Time (Frame)- Synchronous Viterbi Search for Large-Vocabulary Continuous Speech Recognition

- **MAP Principle**

$$W^* = \arg\max_W [p(W|O)] = \arg\max_W [\frac{p(O|W)p(W)}{p(O)}] = \arg\max_W [p(O|W)p(W)]$$

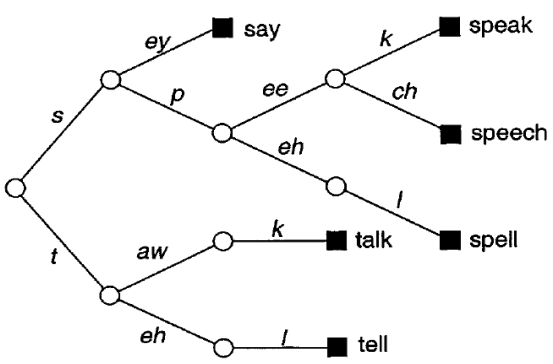
$$p(O|W) = \sum_{all\ \bar{q}} p(O,\bar{q}|W), \bar{q}: a\ state\ \ sequence$$

from HMM; from Language Model

- **An Approximation**

$$W^* = \arg\max_W [p(W)\sum_{all\ \bar{q}} p(O,\bar{q}|W)] \cong \arg\max_W [p(W)\cdot \max_{\bar{q}} p(O,\bar{q}|W)]$$

  - the word sequence with the highest probability for the most probable state sequence usually has approximately the highest probability for all state sequences
  - Viterbi search, a sub-optimal approach
- **Viterbi Search—Dynamic Programming**
  - replacing the problem by a smaller sub-problem and formulating an iterative procedure
  - time (frame)- synchronous: the best score at time t is updated from all states at time t-1
- **Tree Lexicon as the Basic Working Structure**



  - each arc is an HMM (phoneme, tri-phone, etc.)
  - each leaf node is a word
  - search processes for a segment of utterance through some common units for different words can be shared
  - search space constrained by the lexicon
  - the same tree copy reproduced at each leaf node in principle

Basic Problem 2 for HMM (P.24 of 4.0)

- **Application Example of Viterbi Algorithm**

  - Isolated word recognition

$$\lambda_0 = (A_0, B_0, \pi_0)$$
$$\lambda_1 = (A_1, B_1, \pi_1)$$
$$\vdots$$
$$\lambda_n = (A_n, B_n, \pi_n)$$

observation

$$\overline{O} = (o_1, o_2, \ldots o_T)$$

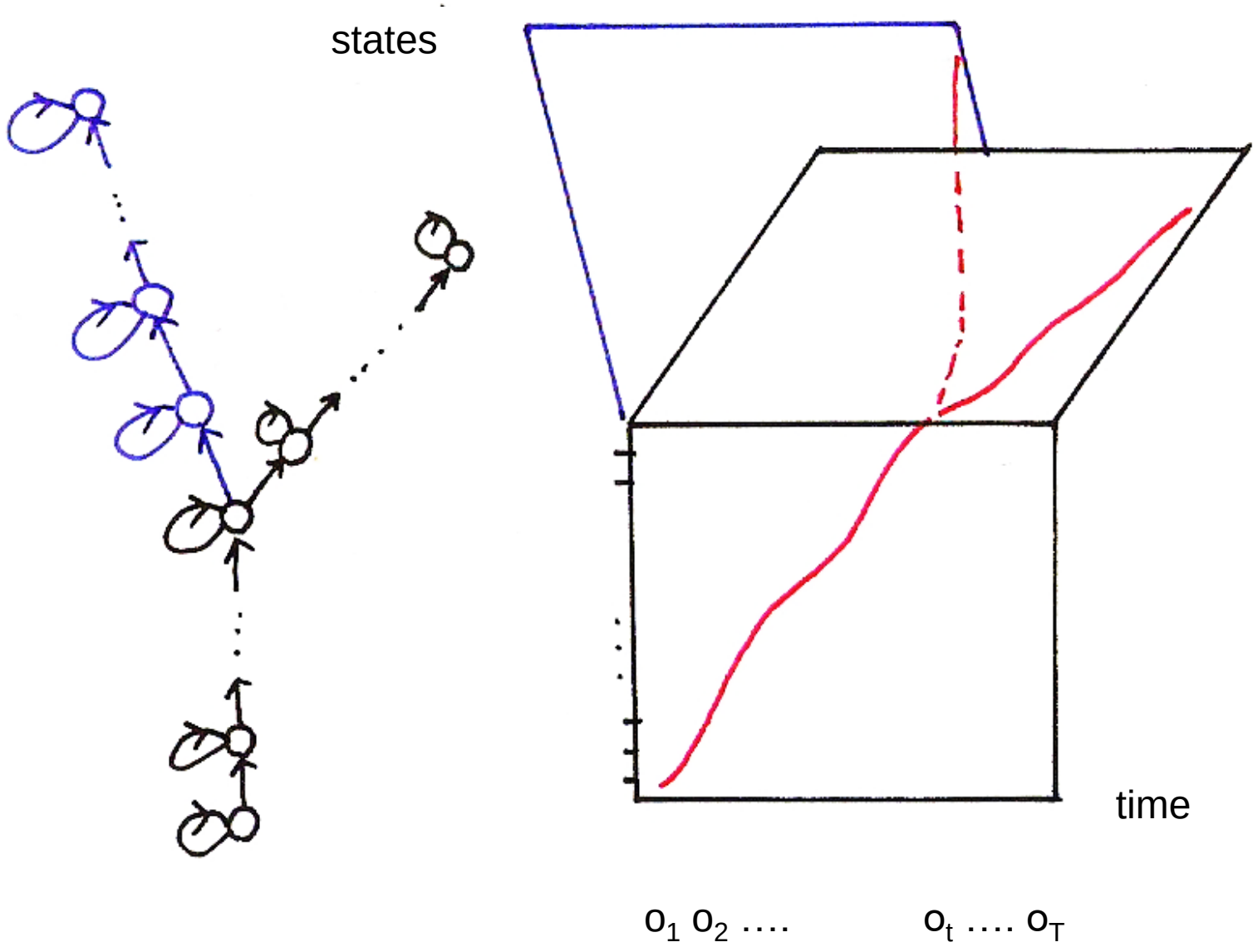
$$k^* = \arg\max_{1 \le i \le n} P[\overline{O} \mid \lambda_i] \approx \arg\max_{1 \le i \le n} [P^* \mid \lambda_i]$$

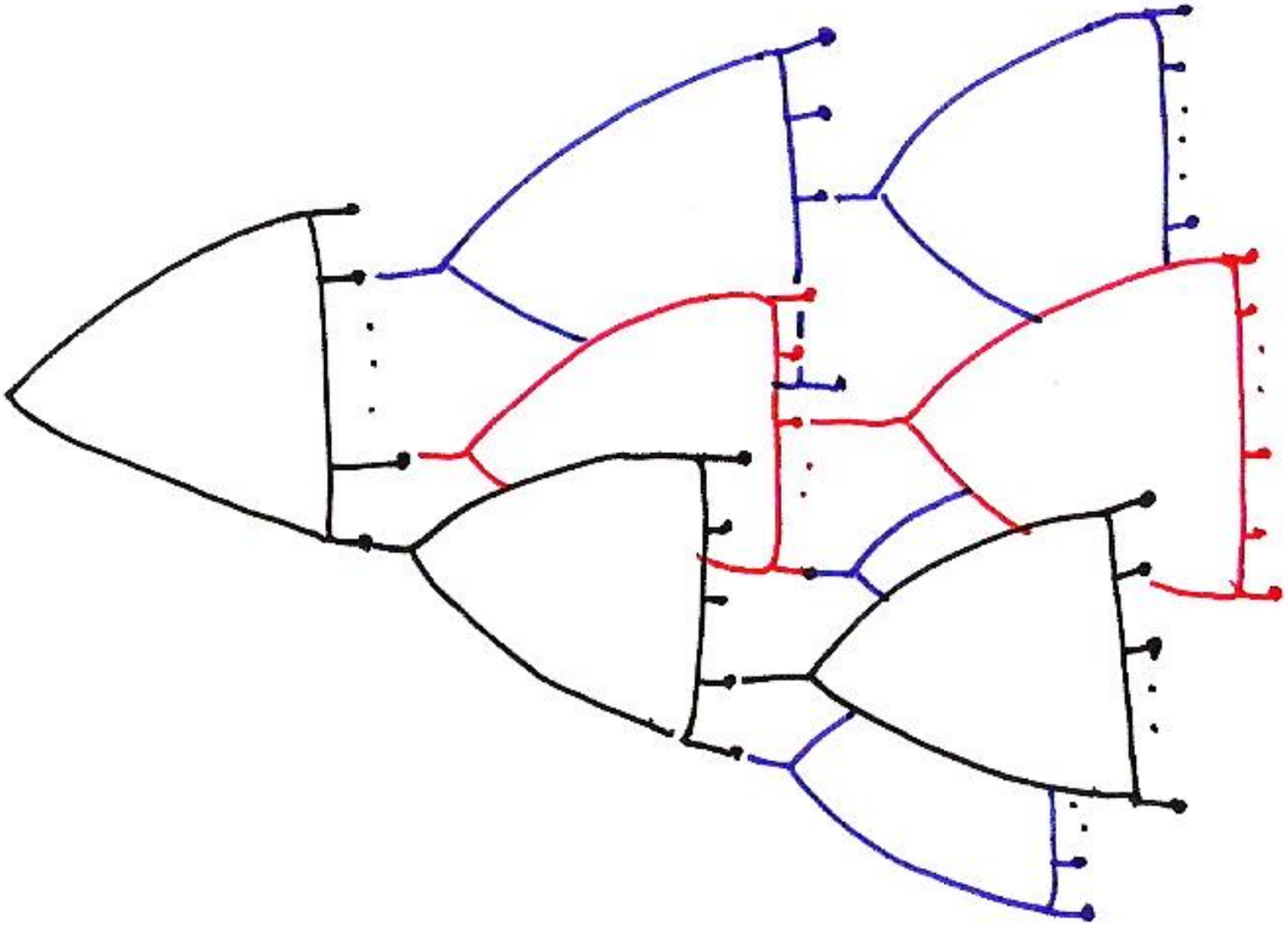
Basic Problem 1
Forward Algorithm
(for all paths)

Basic Problem 2
Viterbi Algorithm
(for a single best path)

-The model with the highest probability for the most probable path usually also has the highest probability for all possible paths.

Tree Lexicon





Time (Frame)- Synchronous Viterbi Search for Large –Vocabulary Continuous Speech Recognition

- **Define Key Parameters**

  D (t, q_t, w) : objective function for the best partial path ending at time t in state q_t for the word w

  h (t, q_t, w) : backtrack pointer for the previous state at the pervious time when the best partial path ends at time t in state q_t for the word w

- **Intra-word Transition—HMM only, no Language Model**

$$D(t, q_t, w) = \max_{q_{t-1}} [d(o_t, q_t | q_{t-1}, w) + D(t-1, q_{t-1}, w)]$$

$$d(o_t, q_t | q_{t-1}, w) = \log p(o_t | q_t, w) + \log p(q_t | q_{t-1}, w)$$

$$\bar{q}(t, q_t, w) = \arg\max_{q_{t-1}} \left[d(o_t, q_t | q_{t-1}, w) + D(t-1, q_{t-1}, w) \right]$$

$$h(t, q_t, w) = \bar{q}(t, q_t, w)$$

- **Inter-word Transition—Language Model only, no HMM (bi-gram as an example)**

$$D(t, Q, w) = \max_{v} [\log p(v|u) + D(t, q_f(v), v)]$$

u : the word before v

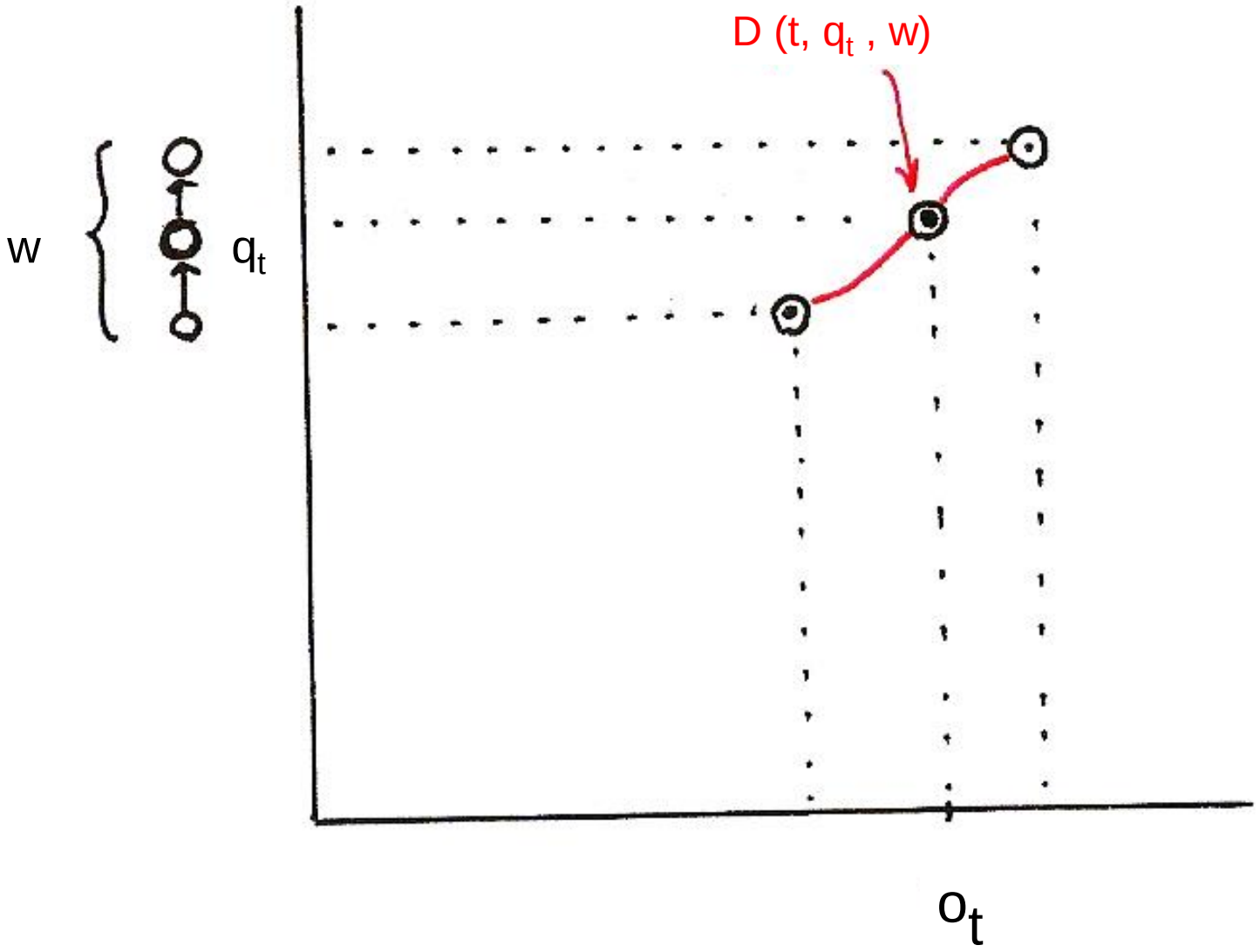
Q : a pseudo initial state for the word w

$q_f(v)$: the final state for the word v

$$\bar{v} : \arg\max_{v} [\log p(v|u) + D(t, q_f(v), v)]$$

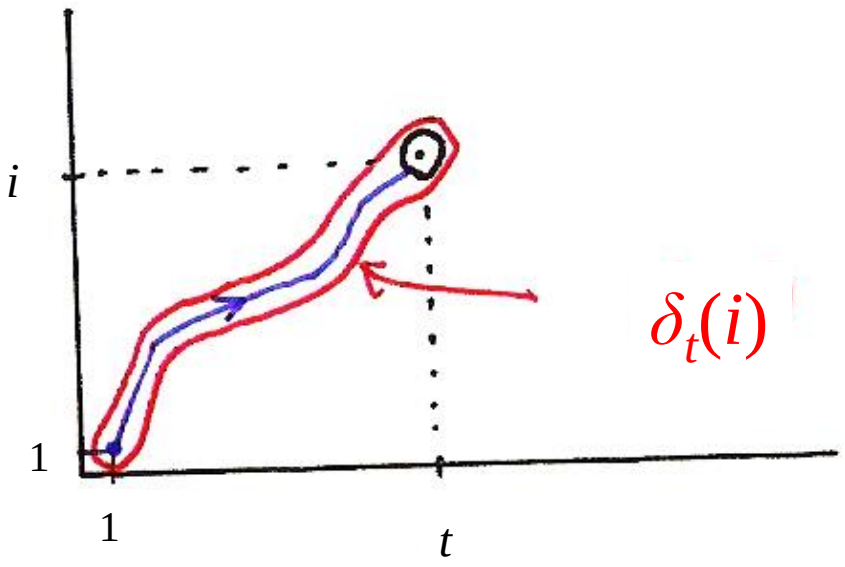
$$h(t, Q, w) = q_f(\bar{v})$$

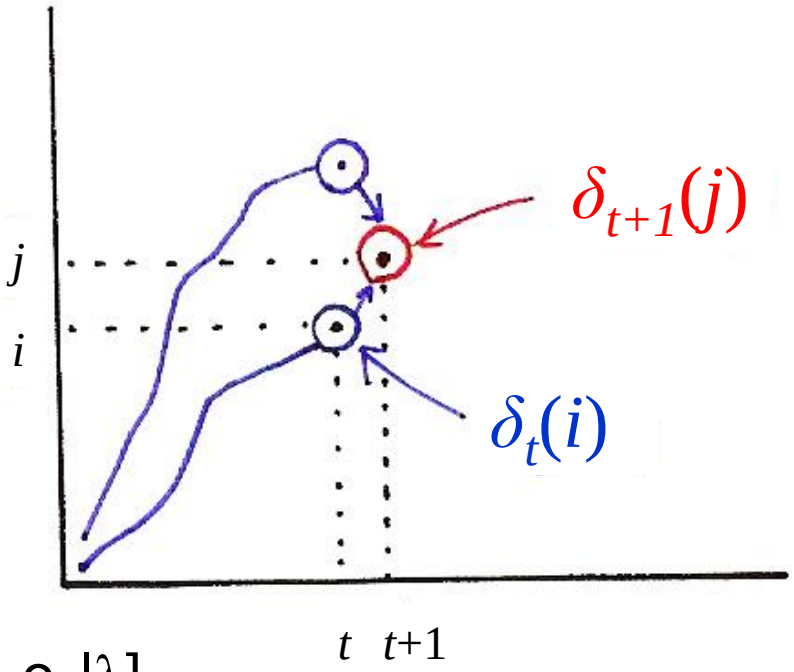
Time Synchronous Viterbi Search



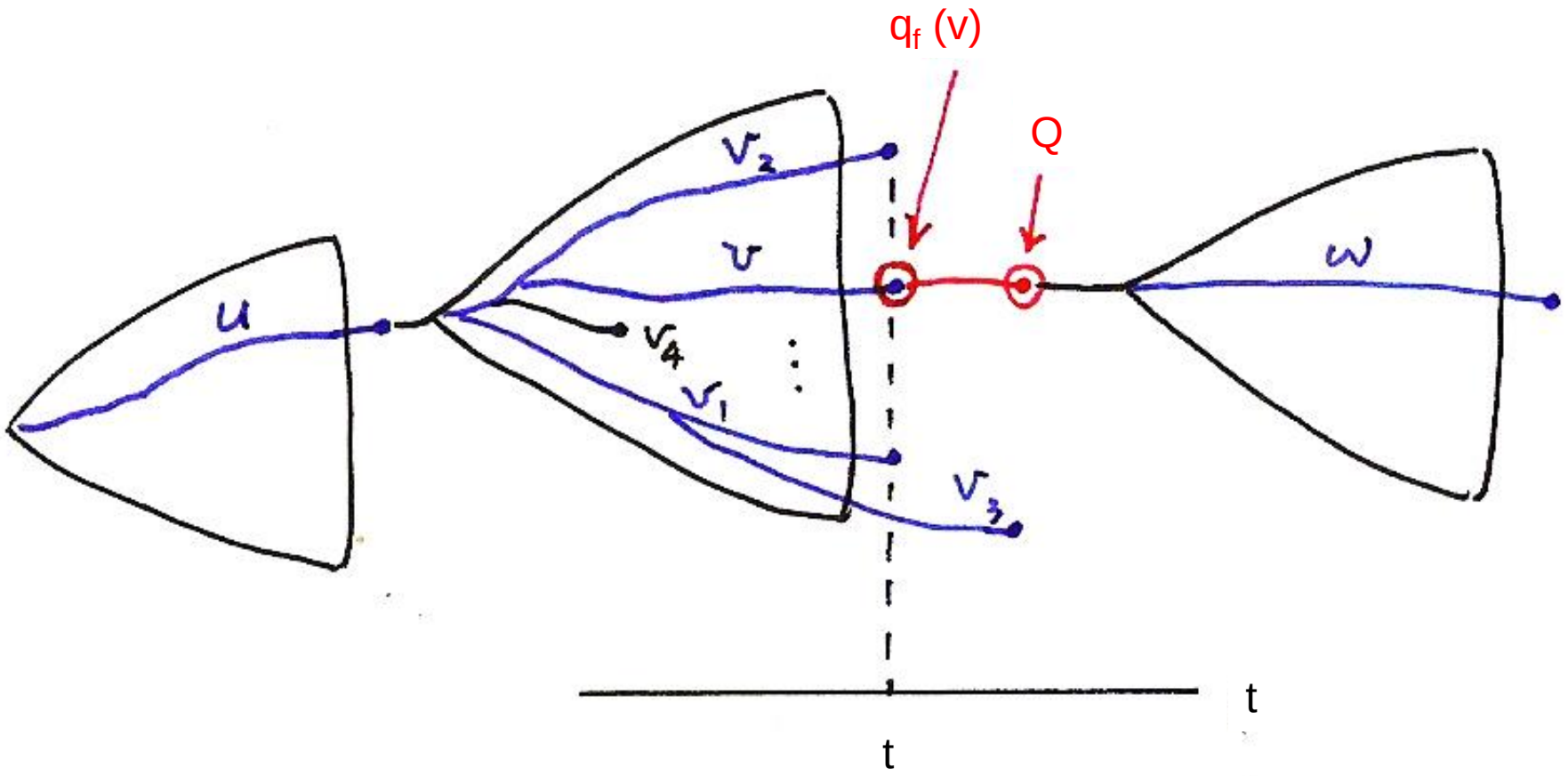
Viterbi Algorithm **(P.21 of 4.0)**

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



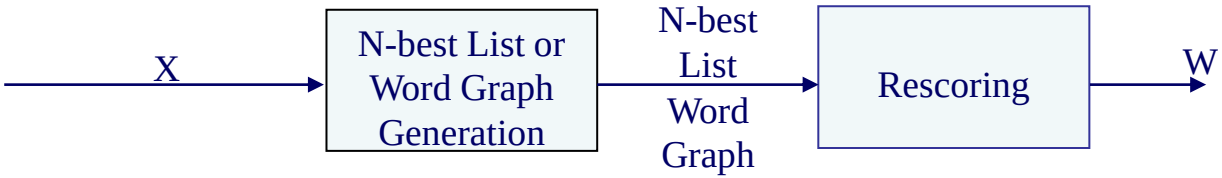


$$\delta_t(i) = \max_{q_1,q_2,\ldots q_{t-1}} P[q_1,q_2,\ldots q_{t-1}, q_t = i, o_1,o_2,\ldots,o_t \mid \lambda]$$


q_f (v)
Q
u
v
w
v_2
v_4
v_1
v_3
t
t

Time (Frame)- Synchronous Viterbi Search for Large-Vocabulary Continuous Speech Recognition

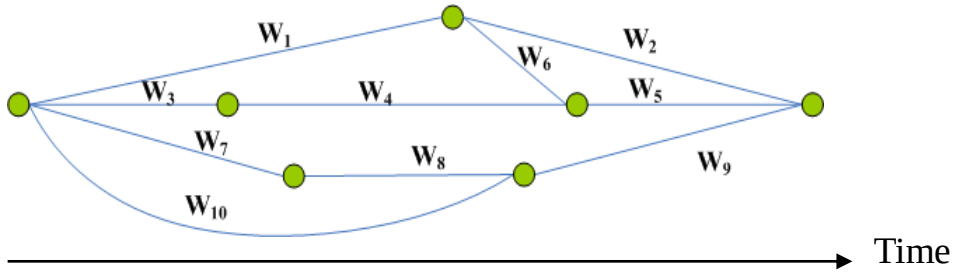
- **Beam Search**
  - at each time t only a subset of promising paths are kept
  - example 1: define a beam width L (i.e. keeping only L paths at each time)
    example 2: define a threshold Th (i.e. all paths with $D < D_{max,t}$-Th are deleted)
  - very helpful in reducing the search space
- **Two-pass Search (or Multi-pass Search)**



  - use less knowledge or less constraints (e.g. acoustic model with less context dependency or language model with lower order) in the first stage, while more knowledge or more constraints in rescoring in the second path
  - search space significantly reduced by decoupling the complicated search process into simpler processes

- **N-best List and Word Graph (Lattice)**

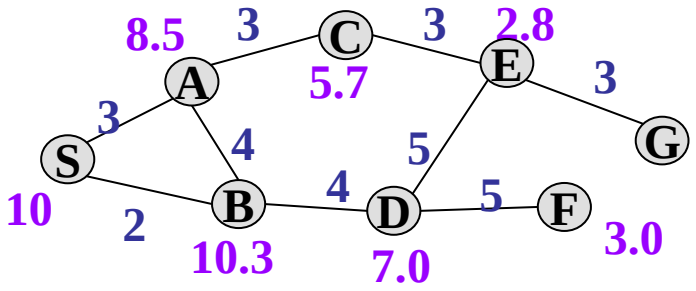
$W_1 W_2$
$W_1 W_6 W_5$
$W_3 W_4 W_5$
$W_7 W_8 W_9$
$W_{10} W_9$



  - similarly constructed with dynamic programming iterations

Some Search Algorithm Fundamentals

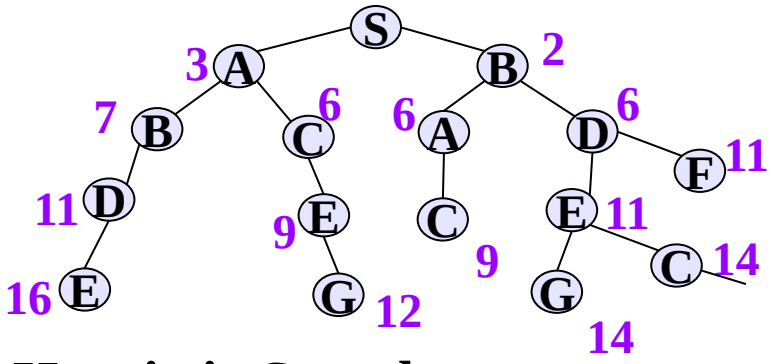
- **An Example – a city traveling problem**



S: starting
G: goal

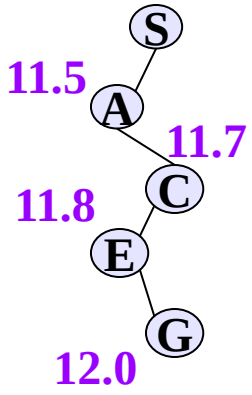
– to find the minimum distance path

- **Search Tree(Graph)**



- **Blind Search Algorithms**
  - Depth-first Search: pick up an arbitrary alternative and proceed
  - Breath-first Search: consider all nodes on the same level before going to the next level
  - no sense about where the goal is

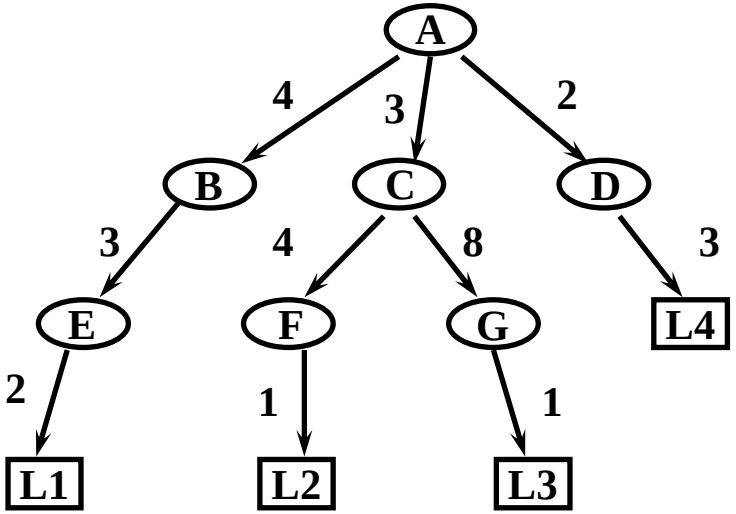
- **Heuristic Search**
  - Best-first Search
  - based on some knowledge, or "heuristic information"
    f(n) = g(n)+h*(n)
    g(n): distance up to node n
    h*(n): heuristic estimate for the remaining distance up to G
  - heuristic pruning



$h^*(n)$: straight-line distance

Heuristic Search: Another Example

- **Problem: Find a path with the highest score from root node "A" to some leaf node (one of "L1","L2","L3","L4")**



$f(n) = g(n) + h^*(n)$

$g(n)$: score from root node to node n

$h(n)$: exact score from node n to a specific leaf node

$h^*(n)$: estimated value for h(n)

Node	g(n)	h*(n)	f(n)
A	0	15	15
B	4	9	13
C	3	12	15
D	2	5	7
E	7	4	11
F	7	2	9
G	11	3	14
L1	9	0	9
L2	8	0	8
L3	12	0	12
L4	5	0	5

List of Candidate Steps

Top	Candidate List
A(15)	A(15)
C(15)	C(15), B(13), D(7)
G(14)	G (14), B(13), F(9), D(7)
B(13)	B(13), L3(12), F(9), D(7)
L3(12)	L3 (12), E(11), F(9), D(7)

A* Search and Speech Recognition

- **Admissibility**
  - a search algorithm is admissible if it is guaranteed that the first solution found is optimal, if one exists (for example, beam search is NOT admissible)
- **It can be shown**
  - the heuristic search is admissible if

    $h^*(n) \geq h(n)$ for all n with a highest-score problem
  - A^* search when the above is satisfied
- **Procedure**
  - Keep a list of next-step candidates, and proceed with the one with the highest f(n) (for a highest-score problem)
- **A^* search in Speech Recognition**
  - example 1: use of weak constraints in the first pass to generate heuristic estimates in multi-pass search
  - example 2: estimated average score per frame as the heuristic information

    $s_f = [\log P(\bar{o}_{i,j} | \bar{q}_{i,j})]/(j - i + 1)$

    $\bar{o}_{i,j}$: observations from frame i to j, $\bar{q}_{i,j}$: state sequences from frame i to j

    estimated with many (i, j) pairs from training data

    $h^*(n)$ obtained from Max $[s_f]$, Ave $[s_f]$, Min $[s_f]$ and (T - t)