Lecture 04: Modelling Sequences

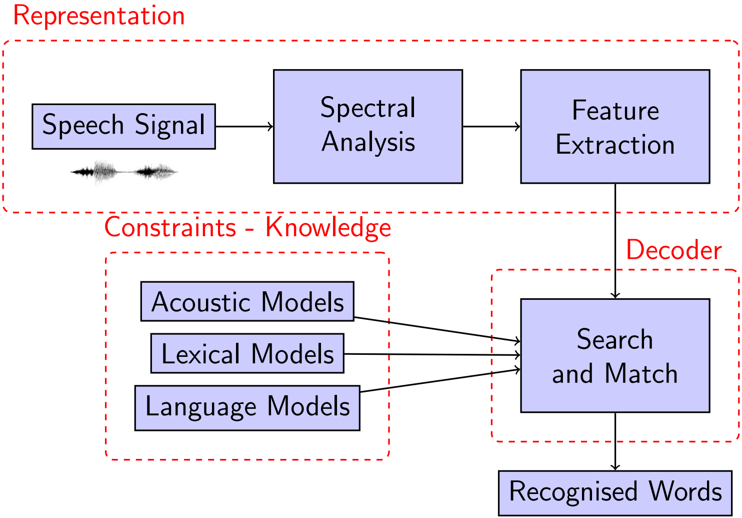
DT2118 Speech and Speaker Recognition

Giampiero Salvi

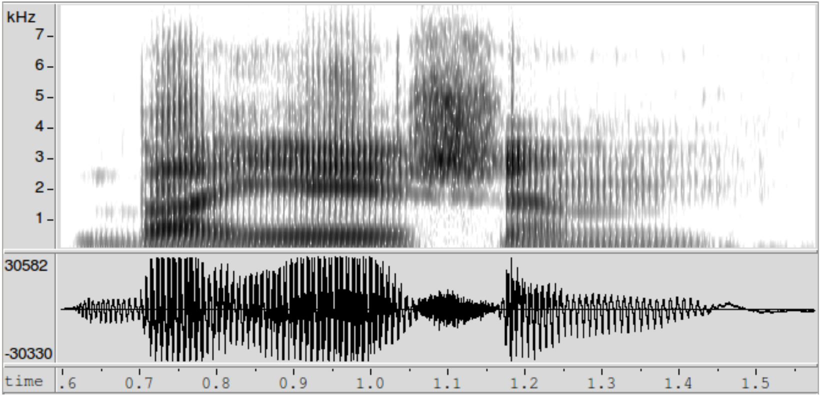
KTH/CSC/TMH giampi@kth.se

VT2016

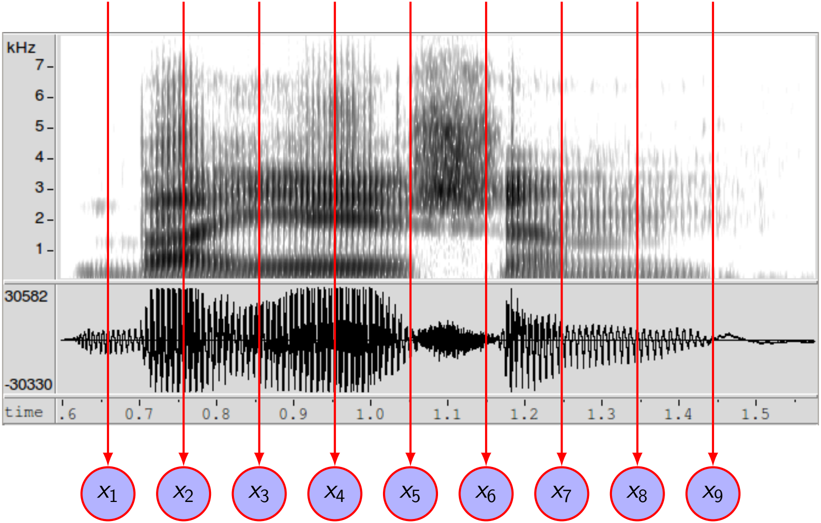
Components of ASR System



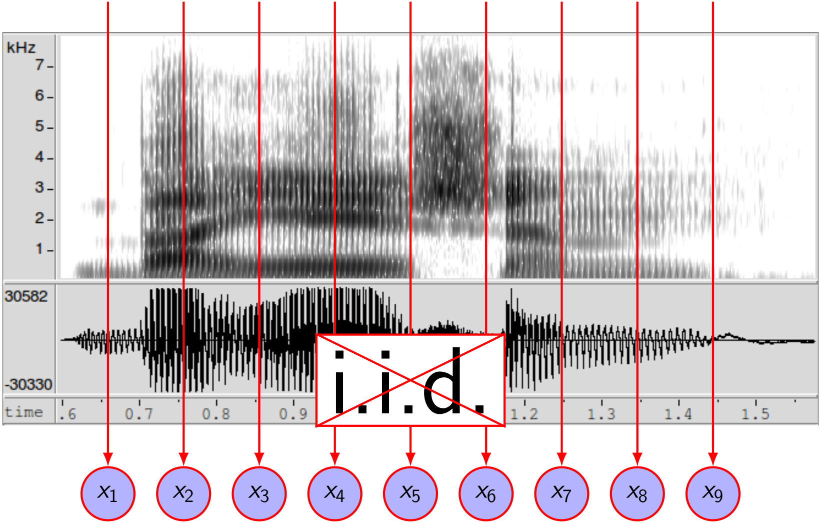
Sequences in Statistical Terms



Sequences in Statistical Terms



Sequences in Statistical Terms

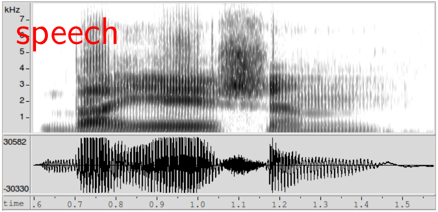


Sequential Data: Not Only Speech

Time sequences

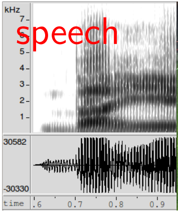
Sequential Data: Not Only Speech

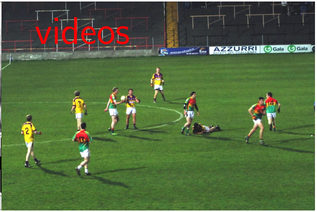
Time sequences



Sequential Data: Not Only Speech

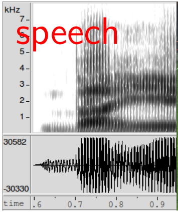
Time sequences

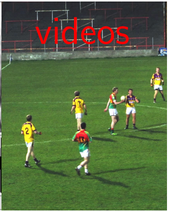




Sequential Data: Not Only Speech

Time sequences

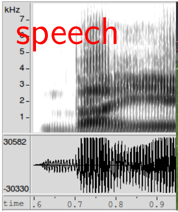

speech

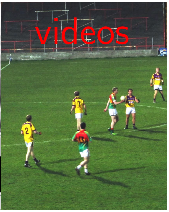

videos

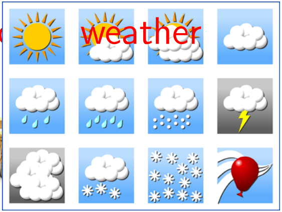
econometrics

Sequential Data: Not Only Speech

Time sequences

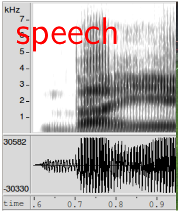






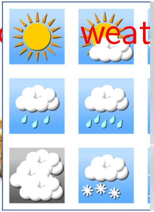
Sequential Data: Not Only Speech

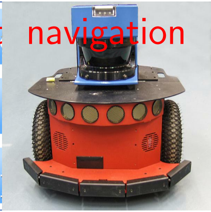
Time sequences


speech

videos

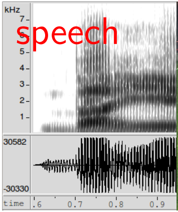
econo


weat

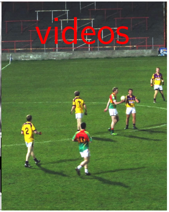

navigation

Sequential Data: Not Only Speech

Time sequences

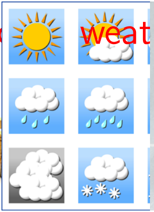


speech



videos

econ



weat

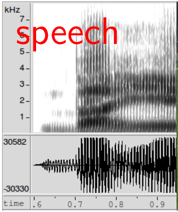
navigation

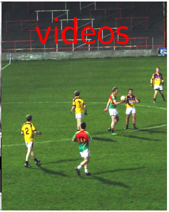
Timeless sequences

DNA

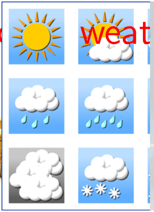
Sequential Data: Not Only Speech

Time sequences


speech

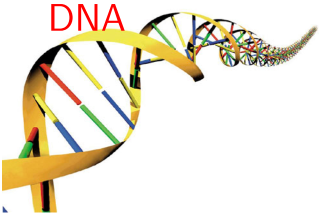

videos

econo


weat

navigation

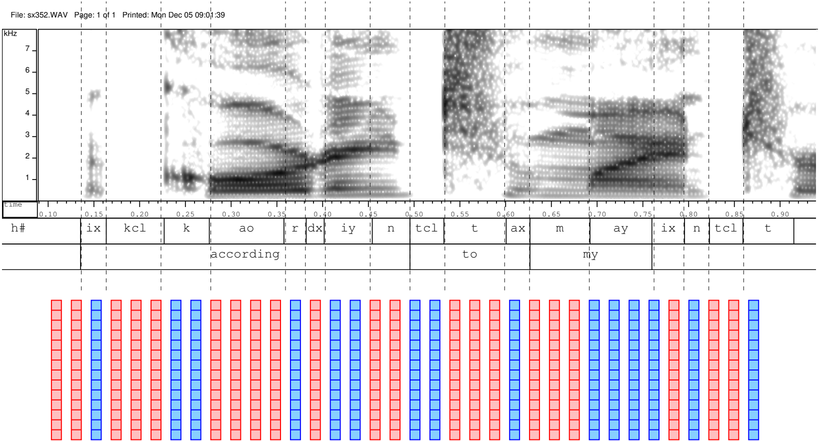
Timeless sequences


DNA

text

Nel mezzo del cammin di nostra vita
mi ritrovai per una selva oscura,
ché la diritta via era smarrita.
Ahi quanto a dir qual era è cosa dura
esta selva selvaggia e aspra e forte
che nel pensier rinova la paura!
Tant' è amara che poco è più morte;
ma per trattar del ben ch'i' vi trovai,
dirò de l'altre cose ch'i' v'ho scorte.
Io non so ben ridir com' i' v'intrai,
tant' era pien di sonno a quel punto
che la verace via abbandonai.

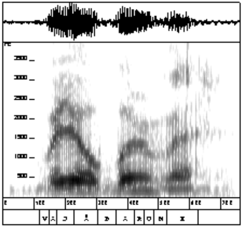
Frame-Based Processing



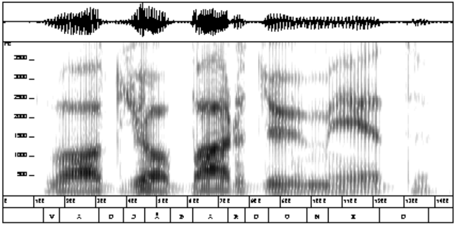
Frame-wise distance metrics

distance	$d(x, y)$
city block:	$\sum_i \lvert x_i - y_i \rvert$
Euclidean:	$\sqrt{\sum_i (x_i - y_i)^2}$
Mahalanobis:	$\sum_i (x_i - \mu_y)^2 / \sigma_y$
probability function:	$f(X = x \lvert \mu_y, \Sigma_y)$
artificial neural networks:	$f\left(\sum_i w_i x_i - \theta\right)$

Comparing Utterances



Va jobbaru me



Vad jobbar du med

“What is your occupation”
(“What work you with”)

Combining frame-wise scores into utterance scores

Template Matching

- oldest technique
- simple comparison of template patterns
- compensate for varying speech rate (Dynamic Programming)

Hidden Markov Models (HMMs)

- most used technique
- models of segmental structure of speech
- recognition by Viterbi search (Dynamic Programming)

Outline

Motivation

Template Matching

- Dynamic Programming

Statistical Approach

- Dynamic Bayesian Networks
- Markov assumption
- State Space Models

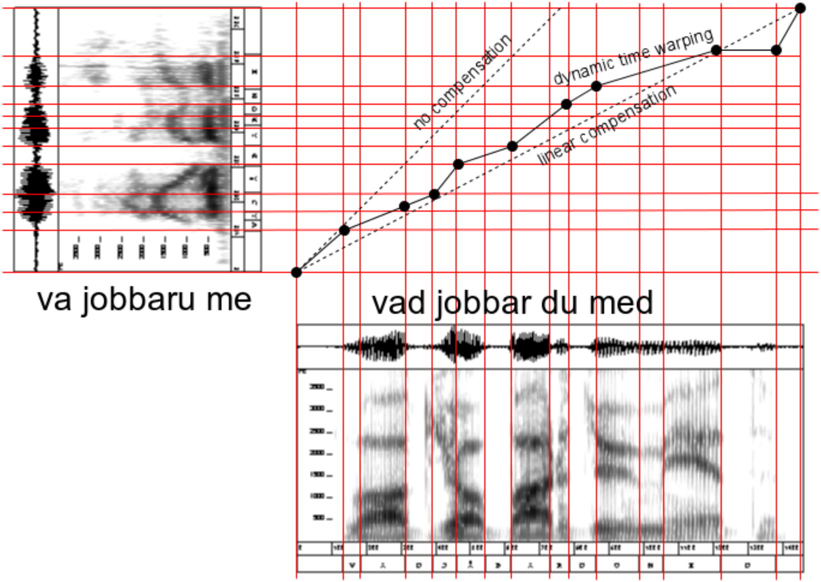
Hidden Markov Models

- Inference

Template Matching: Why?

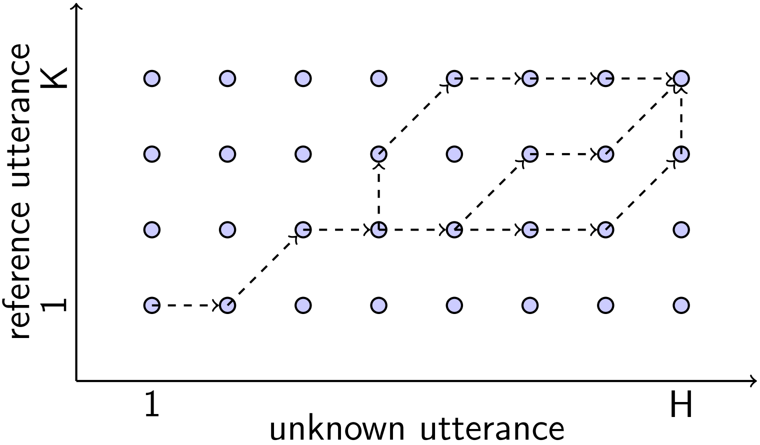
1. Historical: first technique used in ASR
2. Pedagogical: explain the problem and Dynamic Programming
3. Also called Dynamic Time Warping (DTW)

Template Matching



Dynamic Programming

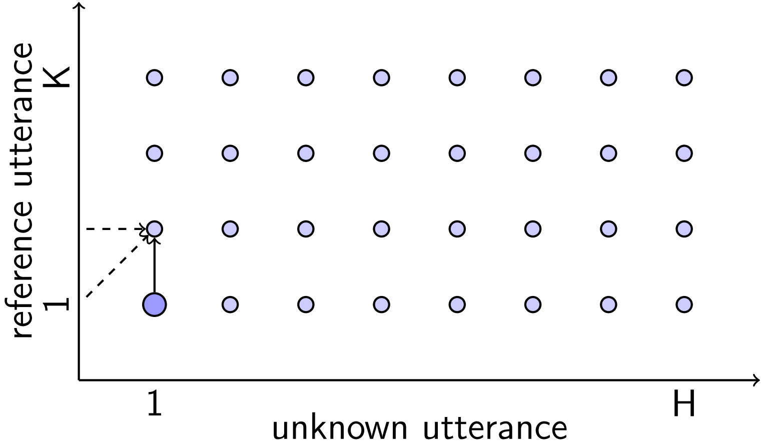
- compare any possible alignment
- problem: exponential with H and K!



Dynamic Programming

Dynamic Time Warping (DTW) algorithm

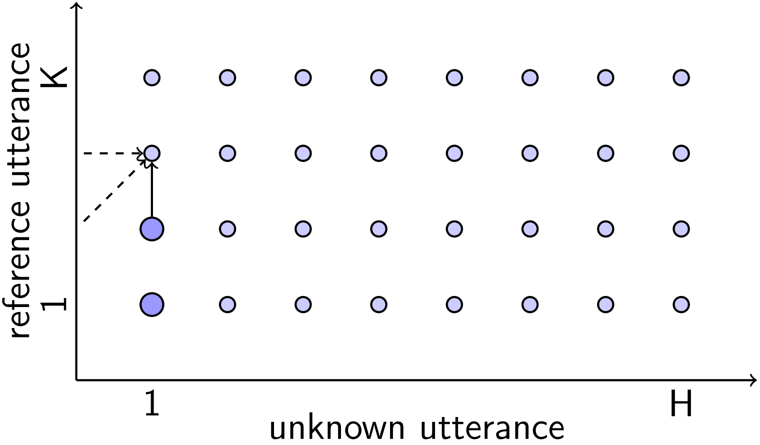
```
for $h = 1$ to $H$ do
    for $k = 1$ to $K$ do
        $AccD[h,k] = LocD[h,k] + \min(AccD[h-1,k], AccD[h-1,k-1], AccD[h,k-1])$
```



Dynamic Programming

Dynamic Time Warping (DTW) algorithm

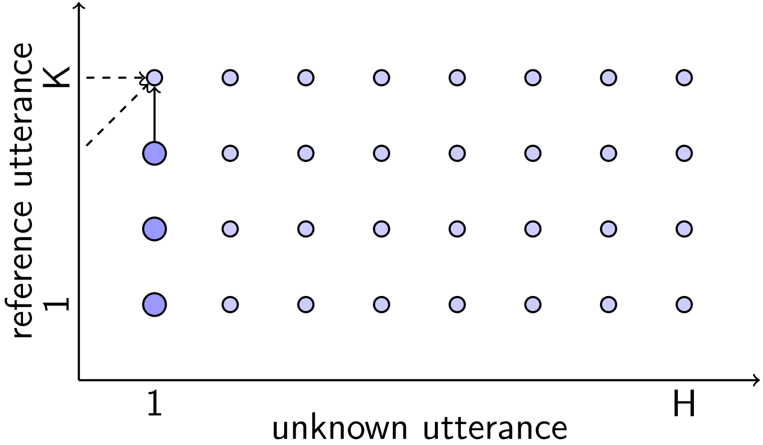
```
for $h = 1$ to $H$ do
    for $k = 1$ to $K$ do
        $AccD[h,k] = LocD[h,k] + \min(AccD[h-1,k], AccD[h-1,k-1], AccD[h,k-1])$
```



Dynamic Programming

Dynamic Time Warping (DTW) algorithm

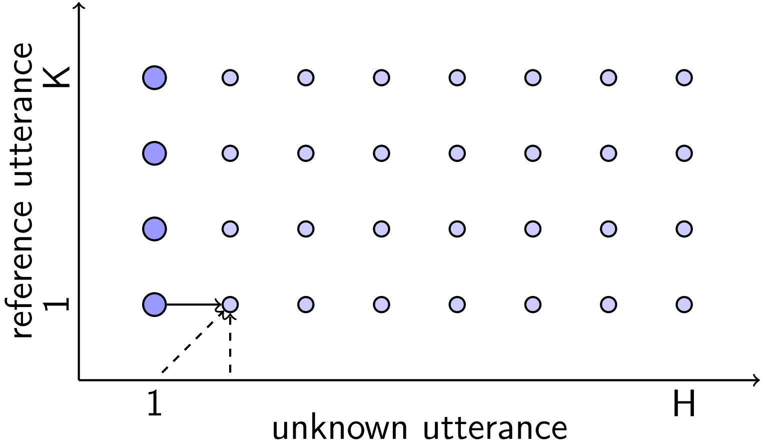
```
1: for $h = 1$ to $H$ do
2:     for $k = 1$ to $K$ do
3:         $AccD[h,k] = LocD[h,k] + \min(AccD[h-1,k], AccD[h-1,k-1], AccD[h,k-1])$
```



Dynamic Programming

Dynamic Time Warping (DTW) algorithm

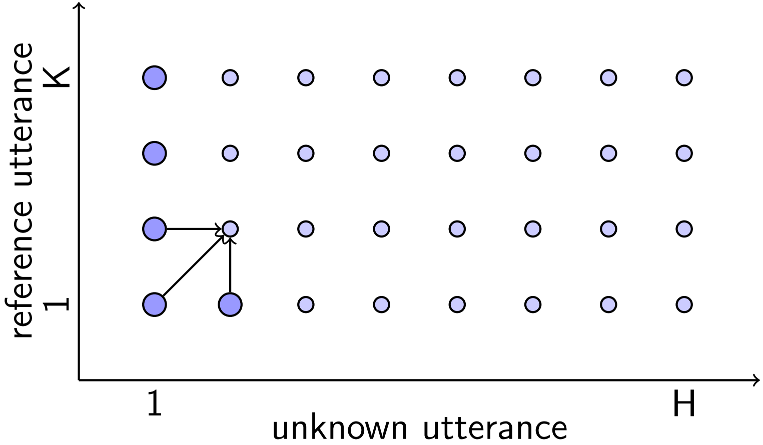
```
for $h = 1$ to $H$ do
    for $k = 1$ to $K$ do
        $AccD[h,k] = LocD[h,k] + \min(AccD[h-1,k], AccD[h-1,k-1], AccD[h,k-1])$
```



Dynamic Programming

Dynamic Time Warping (DTW) algorithm

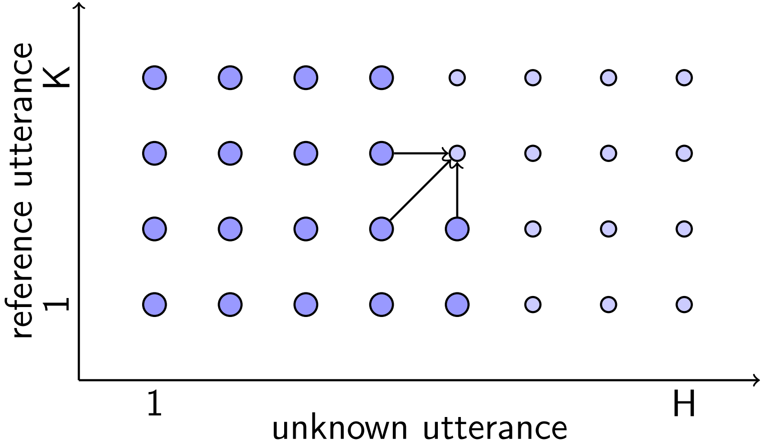
```
for $h = 1$ to $H$ do
    for $k = 1$ to $K$ do
        $AccD[h,k] = LocD[h,k] + \min(AccD[h-1,k], AccD[h-1,k-1], AccD[h,k-1])$
```



Dynamic Programming

Dynamic Time Warping (DTW) algorithm

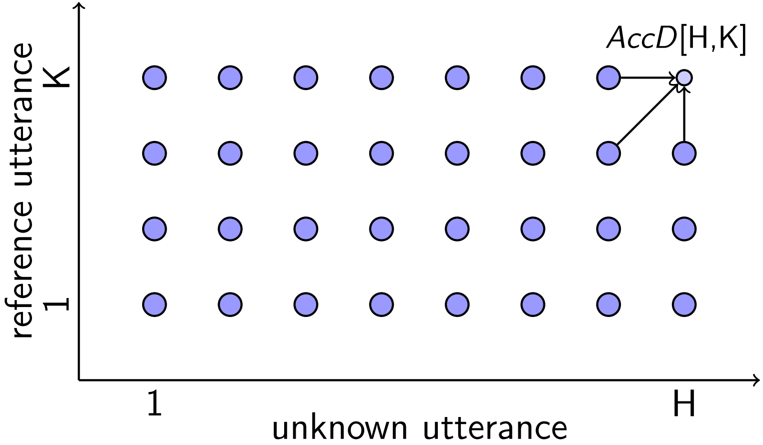
```
for $h = 1$ to $H$ do
    for $k = 1$ to $K$ do
        $AccD[h,k] = LocD[h,k] + \min(AccD[h-1,k], AccD[h-1,k-1], AccD[h,k-1])$
```



Dynamic Programming

Dynamic Time Warping (DTW) algorithm

```
for $h = 1$ to $H$ do
    for $k = 1$ to $K$ do
        $AccD[h,k] = LocD[h,k] + \min(AccD[h-1,k], AccD[h-1,k-1], AccD[h,k-1])$
```



DP Example: Spelling

- observations are letters
- local distance: 0 (same letter), 1 (different letter)
- Unknown utterance: ALLDRIG
- Reference1: ALDRIG
- Reference2: ALLTID
- Problem: find closest match

Distance char-by-char:

- ALLDRIG–ALDRIG $= 5$
- ALLDRIG–ALLTID $= 4$

DP Example: Solution

$LocD$[h,k]=

$$
\begin{array}{cccccccc}
\texttt{G} & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
\texttt{I} & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\
\texttt{R} & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\
\texttt{D} & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
\texttt{L} & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
\texttt{A} & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
 & \texttt{A} & \texttt{L} & \texttt{L} & \texttt{D} & \texttt{R} & \texttt{I} & \texttt{G}
\end{array}
$$

$AccD$[h,k]=

$$
\begin{array}{cccccccc}
\texttt{G} & 5 & 4 & 4 & 3 & 2 & 1 & 0 \\
\texttt{I} & 4 & 3 & 3 & 2 & 1 & 0 & 1 \\
\texttt{R} & 3 & 2 & 2 & 1 & 0 & 1 & 2 \\
\texttt{D} & 2 & 1 & 1 & 0 & 1 & 2 & 3 \\
\texttt{L} & 1 & 0 & 0 & 1 & 2 & 3 & 4 \\
\texttt{A} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
 & \texttt{A} & \texttt{L} & \texttt{L} & \texttt{D} & \texttt{R} & \texttt{I} & \texttt{G}
\end{array}
$$

Distance ALLDRIG–ALDRIG: $AccD$[H,K] $= 0$

DP Example: Solution

$LocD$[h,k]=

$$\begin{array}{cccccccc}
\mathtt{G} & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
\mathtt{I} & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\
\mathtt{R} & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\
\mathtt{D} & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
\mathtt{L} & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
\mathtt{A} & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
 & \mathtt{A} & \mathtt{L} & \mathtt{L} & \mathtt{D} & \mathtt{R} & \mathtt{I} & \mathtt{G}
\end{array}$$

$AccD$[h,k]=

$$\begin{array}{cccccccc}
\mathtt{G} & 5 & 4 & 4 & 3 & 2 & 1 & 0 \\
\mathtt{I} & 4 & 3 & 3 & 2 & 1 & 0 & 1 \\
\mathtt{R} & 3 & 2 & 2 & 1 & 0 & 1 & 2 \\
\mathtt{D} & 2 & 1 & 1 & 0 & 1 & 2 & 3 \\
\mathtt{L} & 1 & 0 & 0 & 1 & 2 & 3 & 4 \\
\mathtt{A} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
 & \mathtt{A} & \mathtt{L} & \mathtt{L} & \mathtt{D} & \mathtt{R} & \mathtt{I} & \mathtt{G}
\end{array}$$

Distance ALLDRIG–ALDRIG: $AccD$[H,K] $= 0$

Distance ALLDRIG–ALLTID? (5min)

DP Example: Solution

LocD[h,k]=

D	1	1	1	0	1	1	1
I	1	1	1	1	1	0	1
T	1	1	1	1	1	1	1
L	1	0	0	1	1	1	1
L	1	0	0	1	1	1	1
A	0	1	1	1	1	1	1
	A	L	L	D	R	I	G

AccD[h,k]=

D	5	3	3	2	3	3	3
I	4	2	2	2	2	2	3
T	3	1	1	1	2	3	4
L	2	0	0	1	2	3	4
L	1	0	0	1	2	3	4
A	0	1	2	3	4	5	6
	A	L	L	D	R	I	G

Distance ALLDRIG–ALDRIG: *AccD*[H,K] $= 0$

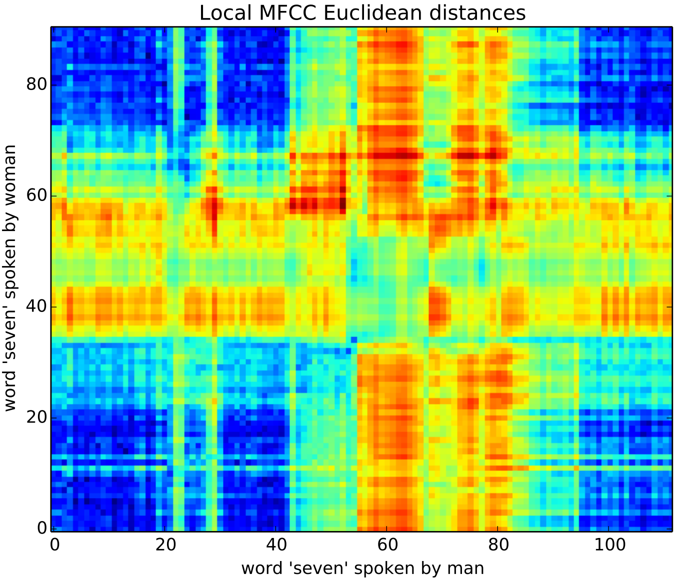
Distance ALLDRIG–ALLTID: *AccD*[H,K] $= 3$

Best path: Backtracking

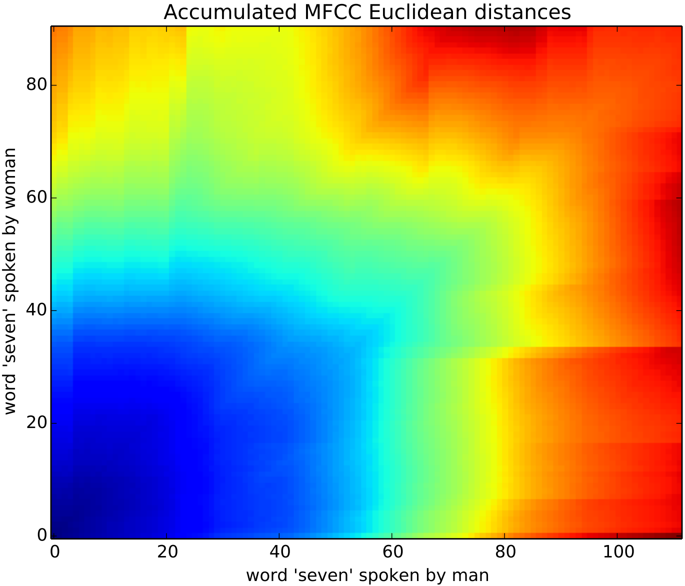
Sometimes we want to know the path

1. at each point [h,k] remember the minimum distance predecessor (back pointer)
2. at the end point [H,K] follow the back pointers until the start

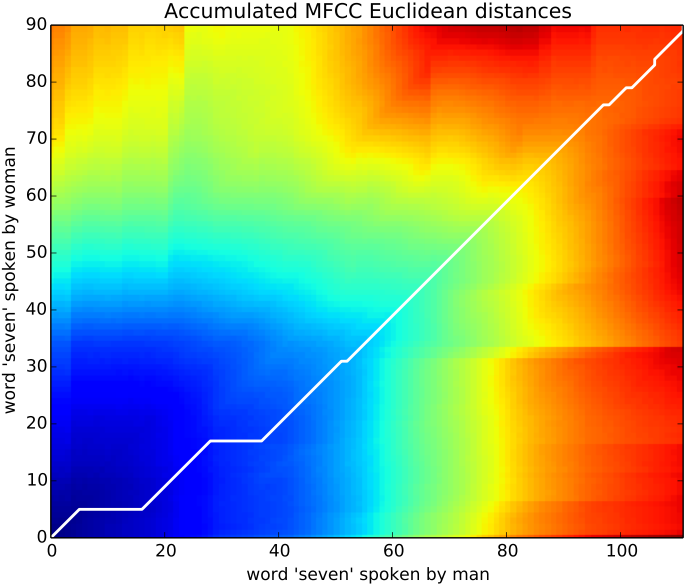
Real Example from Lab1



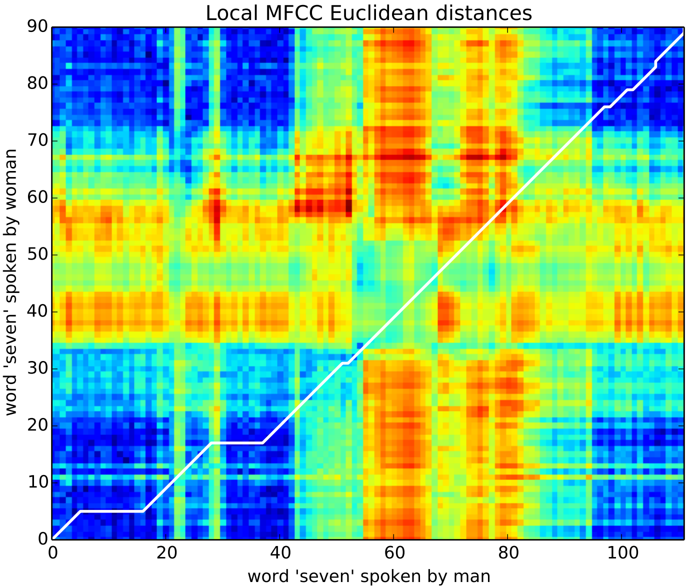
Real Example from Lab1



Real Example from Lab1



Real Example from Lab1



Properties of Template Matching

Pros:

- + No need for phonetic transcriptions
- + within-word co-articulation for free
- + high time resolution

Cons:

- – cross-word co-articulation not modelled
- – needs word segmentation (isolated words)
- – requires recordings of every word
- – not easy to model variation
- – does not scale up with vocabulary size

Outline

Motivation

Template Matching

- Dynamic Programming

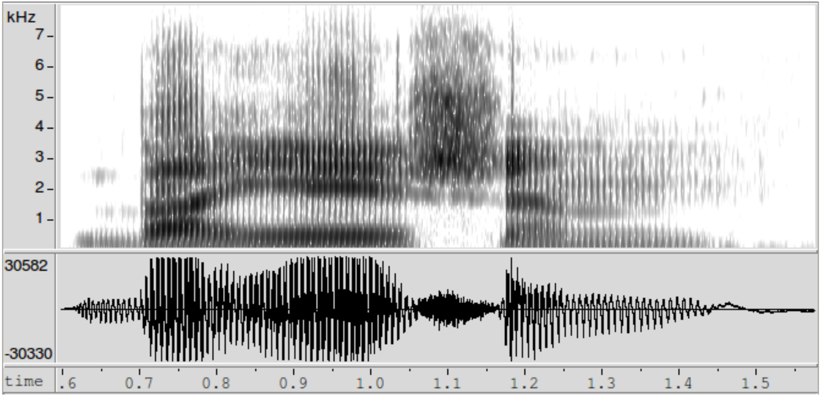
Statistical Approach

- Dynamic Bayesian Networks
- Markov assumption
- State Space Models

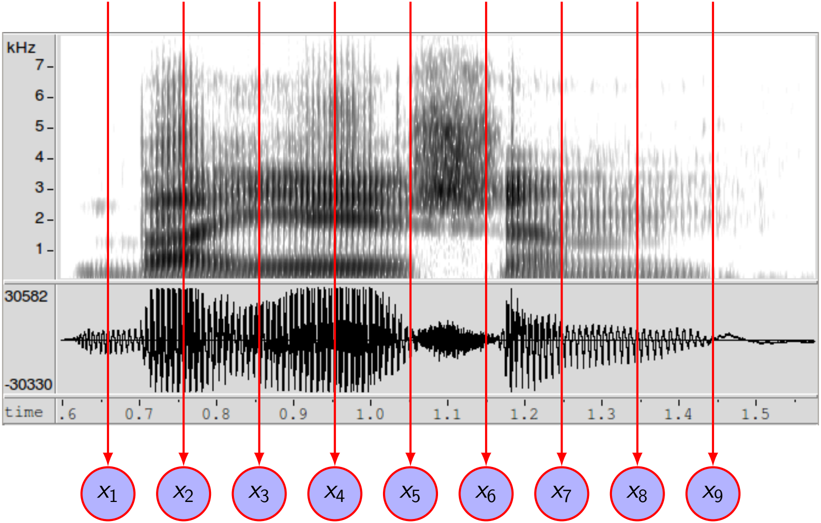
Hidden Markov Models

- Inference

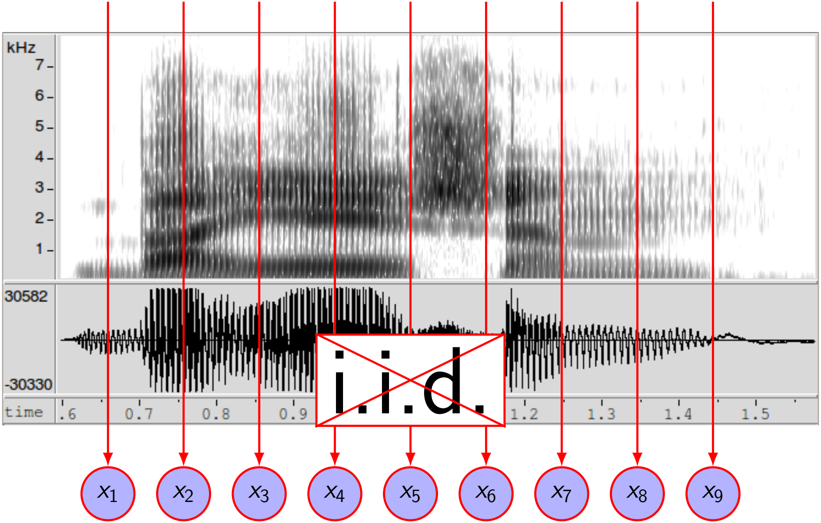
Statistical Approach



Statistical Approach



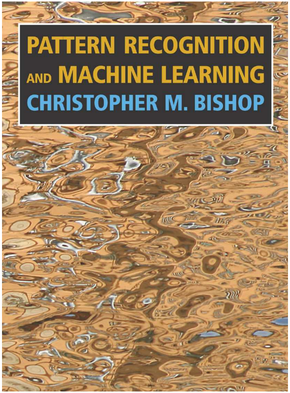
Statistical Approach



Historical Perspective

- Hidden Markov Models first studied in the '60s
- applied to ASR in the end of the '80s
- later seen as special case of Bayesian Networks

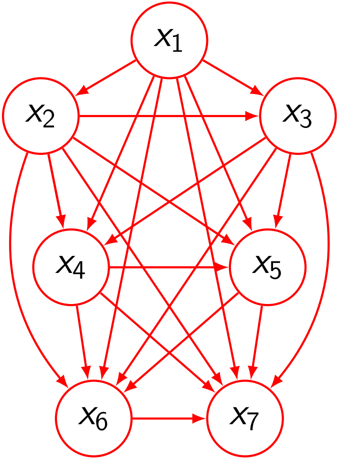
Extra Literature (Optional)



C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006

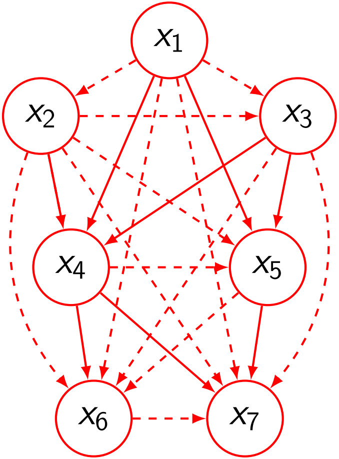
Bayesian Networks (reminder)

$$
\begin{aligned}
p(x_1, \ldots, x_7) = \\
&p(x_1) \\
&p(x_2 | x_1) \\
&p(x_3 | x_1, x_2) \\
&p(x_4 | x_1, x_2, x_3) \\
&p(x_5 | x_1, x_2, x_3, x_4) \\
&p(x_6 | x_1, x_2, x_3, x_4, x_5) \\
&p(x_7 | x_1, x_2, x_3, x_4, x_5, x_6)
\end{aligned}
$$



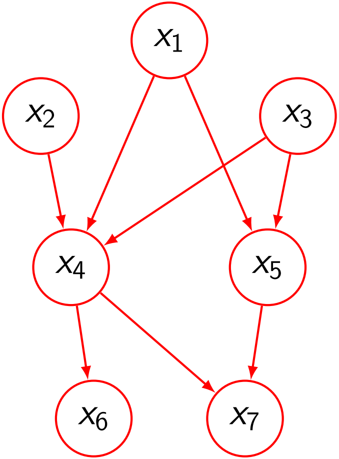
Bayesian Networks (reminder)

$$
\begin{aligned}
p(x_1, \ldots, x_7) = \\
& p(x_1) \\
& p(x_2 | x_1) \\
& p(x_3 | x_1, x_2) \\
& p(x_4 | x_1, x_2, x_3) \\
& p(x_5 | x_1, x_2, x_3, x_4) \\
& p(x_6 | x_1, x_2, x_3, x_4, x_5) \\
& p(x_7 | x_1, x_2, x_3, x_4, x_5, x_6)
\end{aligned}
$$



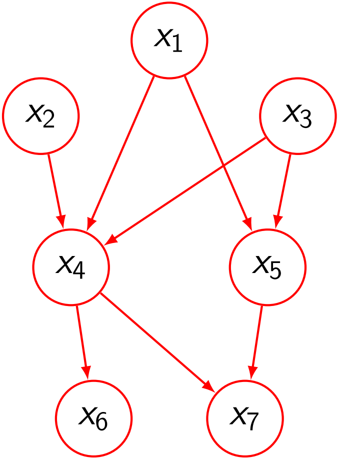
Bayesian Networks (reminder)

$$
\begin{aligned}
p(x_1, \ldots, x_7) = \\
&p(x_1) \\
&p(x_2) \\
&p(x_3) \\
&p(x_4 | x_1, x_2, x_3) \\
&p(x_5 | x_1, x_3) \\
&p(x_6 | x_4) \\
&p(x_7 | x_4, x_5)
\end{aligned}
$$



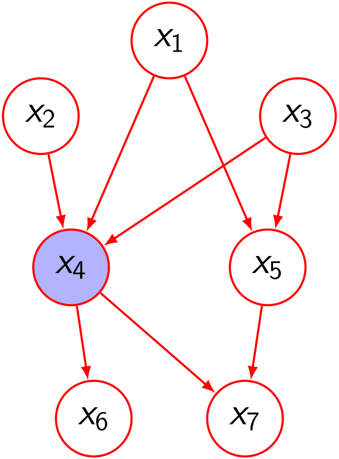
Bayesian Networks (reminder)

$$p(x_1, \ldots, x_7) = \prod_{k=1}^{K} p(x_k | \mathrm{pa}_k)$$



Bayesian Networks (reminder)

If we observe x_4…

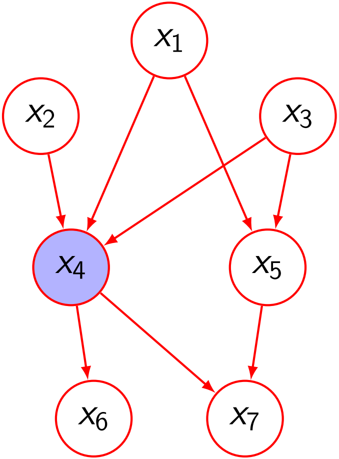


Bayesian Networks (reminder)

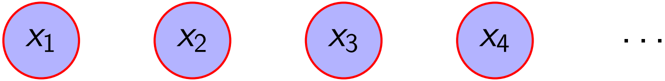
If we observe x_4…

d-separation:

- x_6 and x_7 cond. indep.
- x_1, x_2 and x_3 dependent (head-to-head)

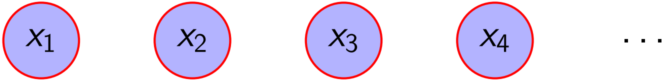


(Dynamic) Bayesian Networks



independence assumption (e.g. i.i.d) not satisfactory

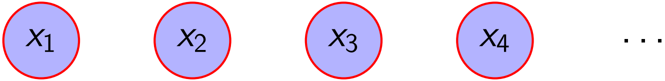
(Dynamic) Bayesian Networks



Most general case, applying product rule recursively $(p(a, b) = p(a)p(b|a))$

$$p(x_1, \dots, x_N) = p(x_1)p(x_2, \dots, x_N|x_1)$$

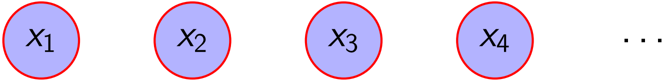
(Dynamic) Bayesian Networks



Most general case, applying product rule recursively $(p(a,b) = p(a)p(b|a))$

$$p(x_1,\dots,x_N) = p(x_1)p(x_2|x_1)p(x_3,\dots,x_N|x_1,x_2)$$

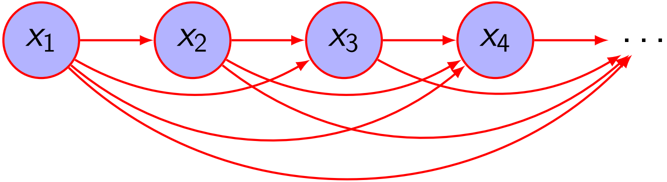
(Dynamic) Bayesian Networks



Most general case, applying product rule recursively ($p(a, b) = p(a)p(b|a)$)

$$\begin{aligned} p(x_1, \ldots, x_N) =& p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)\cdots \\ & \cdots p(x_N|x_1, \ldots, x_{N-1}) \end{aligned}$$

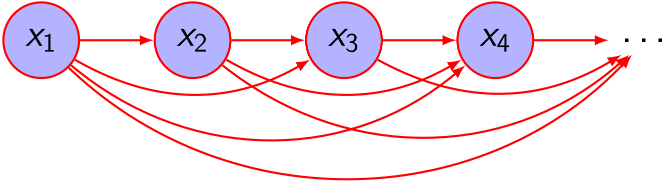
(Dynamic) Bayesian Networks



Most general case, applying product rule recursively $(p(a, b) = p(a)p(b|a))$

$$p(x_1, \dots, x_N) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)\cdots$$
$$\cdots p(x_N|x_1, \dots, x_{N-1})$$

(Dynamic) Bayesian Networks

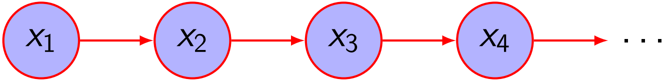


Most general case, applying product rule recursively ($p(a, b) = p(a)p(b|a)$)

$$p(x_1, \ldots, x_N) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \cdots \\ \cdots p(x_N|x_1, \ldots, x_{N-1})$$

Grows quadratically with sequence length (N)!!!

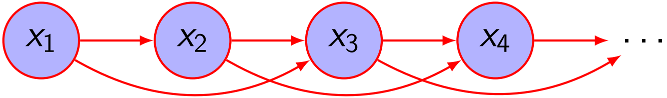
Markov assumption



First order Markov assumption:
$p(x_n|x_1, \dots, x_{n-1}) \approx p(x_n|x_{n-1})$

$$p(x_1, \dots, x_N) = p(x_1) \prod_{n=2}^{N} p(x_n|x_{n-1})$$

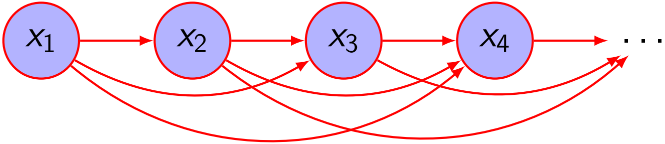
Markov assumption



Second order Markov assumption:

$$p(x_1, \ldots, x_N) = p(x_1)p(x_2|x_1)\prod_{n=3}^{N} p(x_n|x_{n-2}, x_{n-1})$$

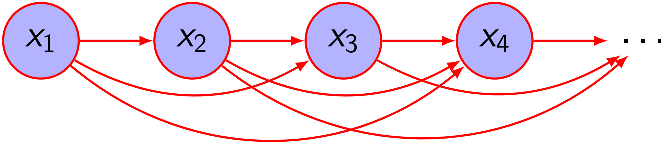
Markov assumption



Third order Markov assumption:

$$p(x_1, \ldots, x_N) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)$$
$$\prod_{n=4}^{N} p(x_n|x_{n-3}, x_{n-2}, x_{n-1})$$

Markov assumption



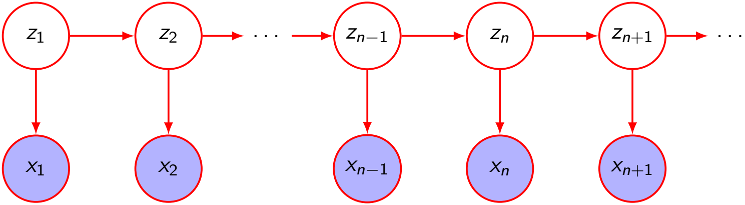
Third order Markov assumption:

$$\begin{aligned} p(x_1, \ldots, x_N) &= p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \\ & \prod_{n=4}^{N} p(x_n|x_{n-3}, x_{n-2}, x_{n-1}) \end{aligned}$$

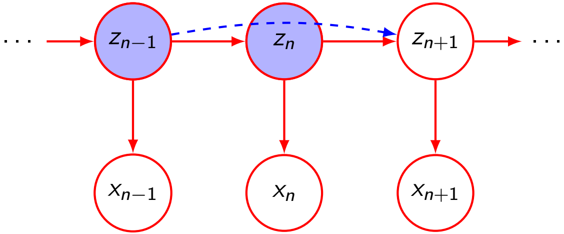
Grows quadratically with order!!!

State Space Models

Adding latent variables z_n

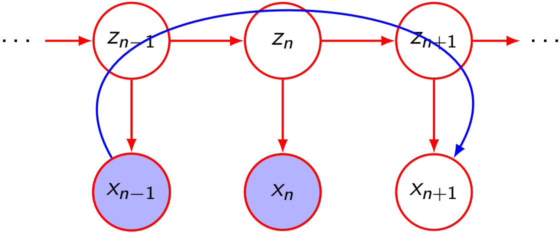


State Space Models: Properties



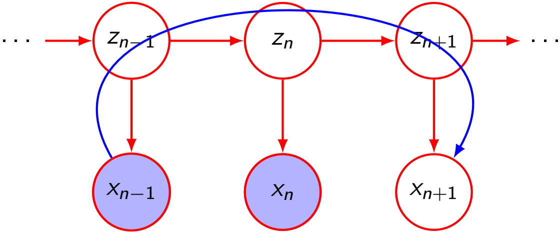
- given z_n, z_{n+1} is independent of $z_1, \ldots, z_{n-1}$
  $p(z_{n+1}|z_1, \ldots, z_n) = p(z_{n+1}|z_n)$

State Space Models: Properties



- given z_n, z_{n+1} is independent of $z_1, \ldots, z_{n-1}$
  $p(z_{n+1}|z_1, \ldots, z_n) = p(z_{n+1}|z_n)$
- $p(x_{n+1}|x_1, \ldots, x_n)$ does not simplify

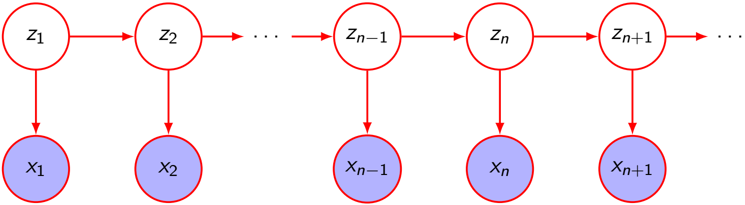
State Space Models: Properties



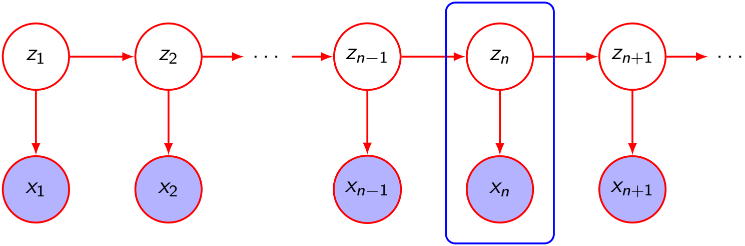
- given z_n, z_{n+1} is independent of $z_1, \ldots, z_{n-1}$
  $p(z_{n+1}|z_1, \ldots, z_n) = p(z_{n+1}|z_n)$
- $p(x_{n+1}|x_1, \ldots, x_n)$ does not simplify

We have modelled indefinitely long dependencies with a limited set of parameters!

Stationary State Space Models

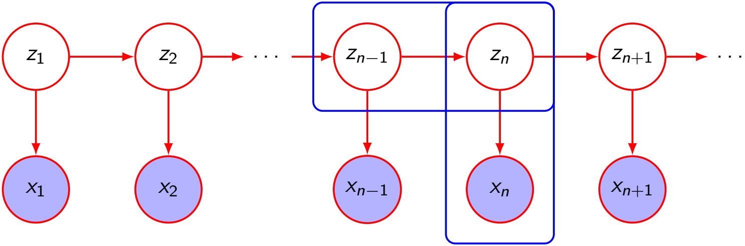


Stationary State Space Models



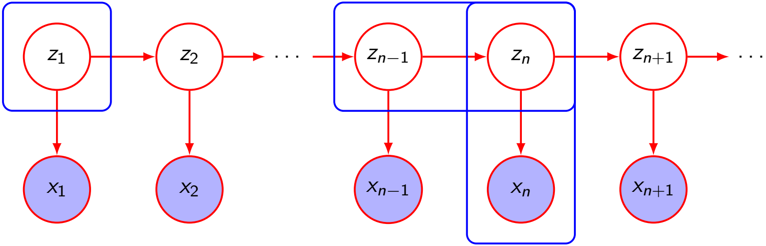
- Emission: $p(x_n|z_n)$

Stationary State Space Models



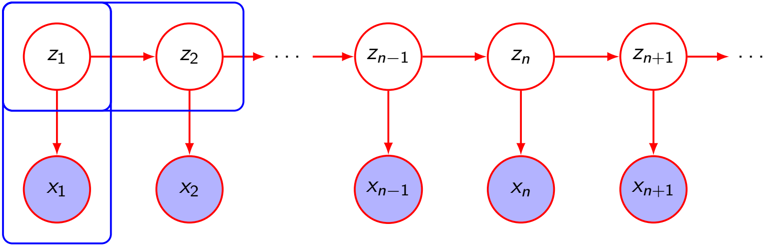
- Emission: $p(x_n|z_n)$
- Transition: $p(z_n|z_{n-1})$

Stationary State Space Models



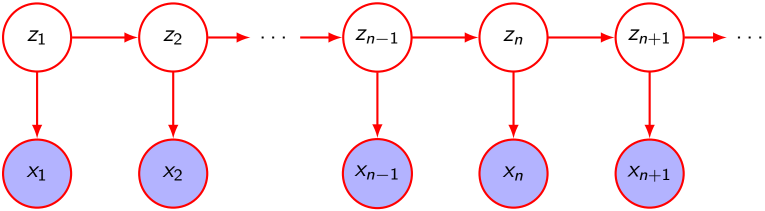
- Emission: $p(x_n|z_n)$
- Transition: $p(z_n|z_{n-1})$
- Initial: $p(z_1)$

Stationary State Space Models



- Emission: $p(x_n|z_n)$
- Transition: $p(z_n|z_{n-1})$
- Initial: $p(z_1)$

State Space Models Instances



- if z_n are discrete: Hidden Markov Models
- if z_n are continuous: Linear Dynamical Systems

Outline

Motivation

Template Matching

- Dynamic Programming

Statistical Approach

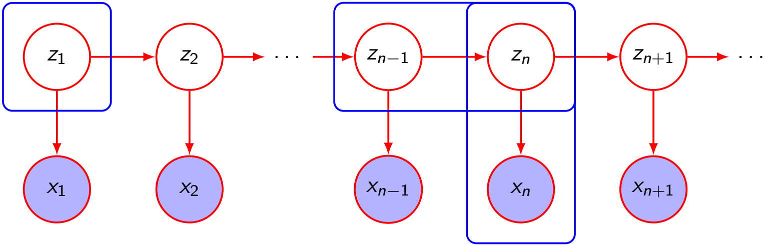
- Dynamic Bayesian Networks
- Markov assumption
- State Space Models

Hidden Markov Models

- Inference

Hidden Markov Models

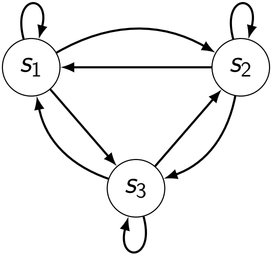
State space models with discrete z_n



- Emission: $p(x_n|z_n) = p(x_n|z_n, \phi)$
  equivalent to Mixture Model
- Transition: $p(z_n|z_{n-1}) = p(z_n|z_{n-1}, A)$
- Initial: $p(z_1) = p(z_1|\pi)$

Hidden Markov Models (HMMs)

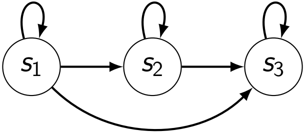
Ergodic HMM



Elements:	
set of states:	$S = \{s_1, s_2, s_3\}$
transition probabilities:	$A(s_a, s_b) = P(s_b, t \vert s_a, t-1)$
prior probabilities:	$\pi(s_a) = P(s_a, t_0)$
state to observation probs:	$\phi(o, s_a) = P(o \vert s_a)$

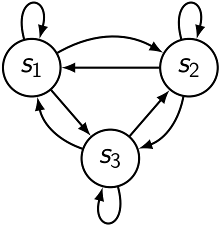
Hidden Markov Models (HMMs)

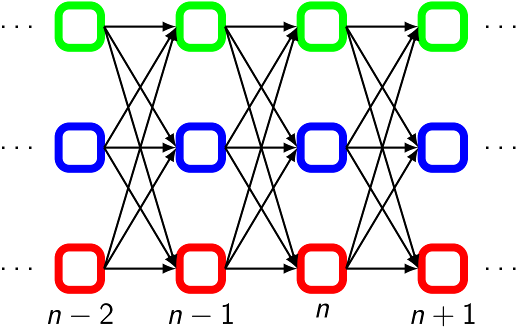
Left-to-right HMM



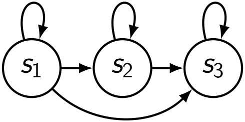
Elements:	
set of states:	$S = \{s_1, s_2, s_3\}$
transition probabilities:	$A(s_a, s_b) = P(s_b, t \mid s_a, t-1)$
prior probabilities:	$\pi(s_a) = P(s_a, t_0)$
state to observation probs:	$\phi(o, s_a) = P(o \mid s_a)$

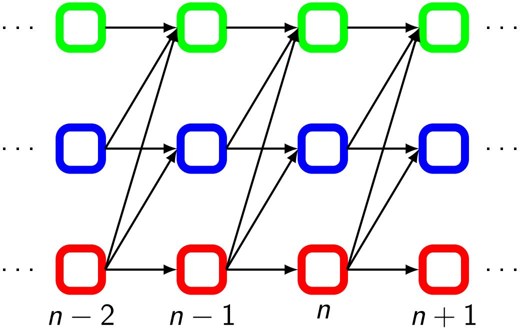
HMMs: Trellis (Lattice)





HMMs: Trellis (Lattice)





A probabilistic perspective: Bayes' rule

$$P(\text{words}|\text{sounds}) = \frac{P(\text{sounds}|\text{words})P(\text{words})}{P(\text{sounds})}$$

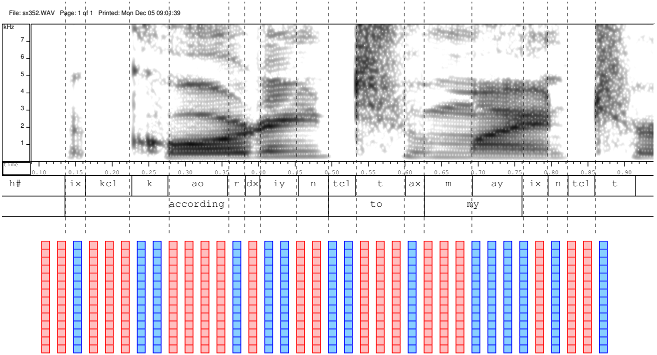
- $P(\text{sounds}|\text{words})$ can be estimated from training data and transcriptions
- $P(\text{words})$: *a priori* probability of the words (Language Model)
- $P(\text{sounds})$: *a priori* probability of the sounds (constant, can be ignored)

Probabilistic Modelling

Problem: How do we model $P(\text{sounds}|\text{words})$?

Probabilistic Modelling

Problem: How do we model $P(\text{sounds}|\text{words})$?

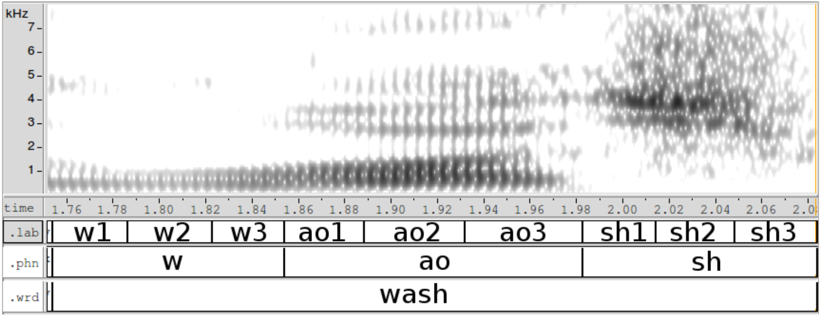


Every feature vector (observation at time t) is a continuous stochastic variable (e.g. MFCC)

Stationarity

Problem: speech is not stationary

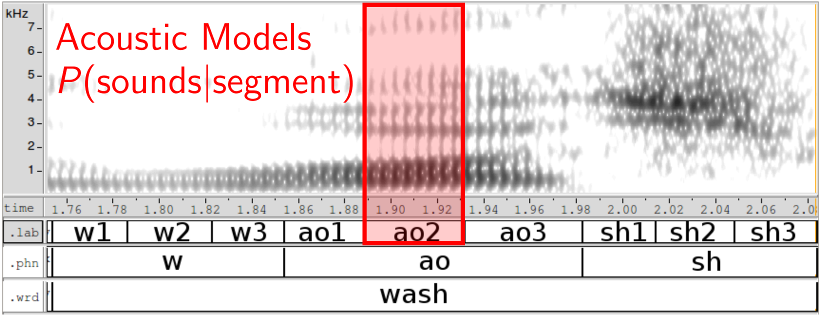
- we need to model short segments independently
- the fundamental unit can not be the word, but must be shorter
- usually we model three segments for each phoneme



Stationarity

Problem: speech is not stationary

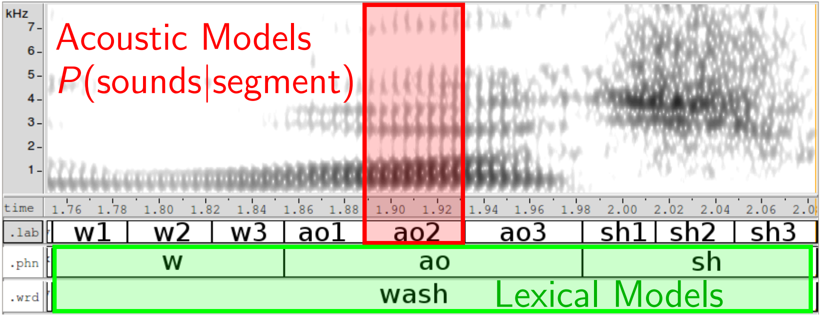
- we need to model short segments independently
- the fundamental unit can not be the word, but must be shorter
- usually we model three segments for each phoneme



Stationarity

Problem: speech is not stationary

- we need to model short segments independently
- the fundamental unit can not be the word, but must be shorter
- usually we model three segments for each phoneme



Local probabilities (frame-wise)

If segment sufficiently short

$$P(\text{sounds}|\text{segment})$$

can be modelled with standard probability distributions

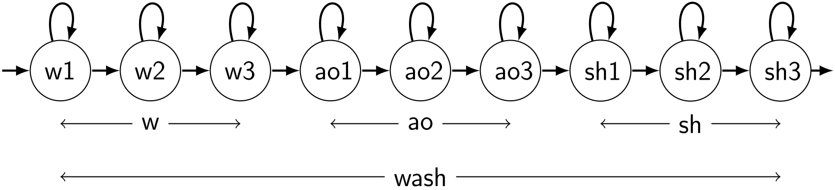
$$\phi(o, s_a) = P(o|s_a)$$

Usually Gaussian or Gaussian Mixture

Global Probabilities (utterance)

Problem: How do we combine the different $P(\text{sounds}|\text{segment})$ to form $P(\text{sounds}|\text{words})$?

Answer: Hidden Markov Model (HMM)



HMM-questions (Inference)

1. what is the probability that the model has generated the sequence of observations? (isolated word recognition)
2. what is the most likely state sequence given the observation sequence? (continuous speech recognition)
3. how can the model parameters be estimated from examples? (training)

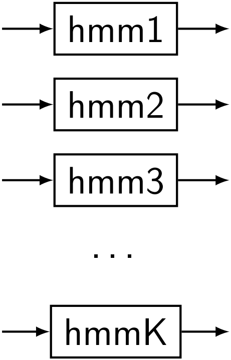
HMM-questions (Inference)

1. what is the probability that the model has generated the sequence of observations? (isolated word recognition) forward algorithm
2. what is the most likely state sequence given the observation sequence? (continuous speech recognition) Viterbi algorithm [3]
3. how can the model parameters be estimated from examples? (training) Baum-Welch[1]

[3] A. J. Viterbi. "Error Bounds for Convolutional Codes and an Asymtotically optimum decoding algorithm". In: *IEEE Trans. Inform. Theory* IT-13 (Apr. 1967), pp. 260–269

[1] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains". In: *Ann. Math. Statist.* 41.1 (1970), pp. 164–171

Isolated Words Recognition

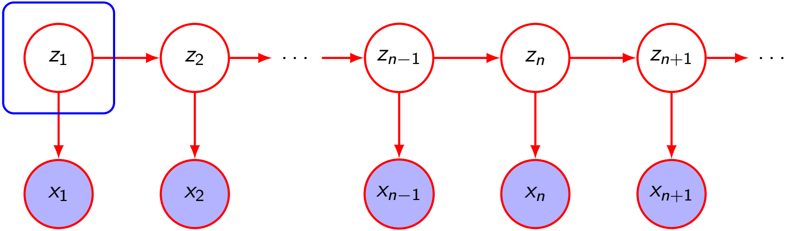


Compare Likelihoods (forward algorithm)

HMM Inference: Joint Distribution

$$X = \{x_1, \dots, x_N\}$$
$$Z = \{z_1, \dots, z_N\}$$

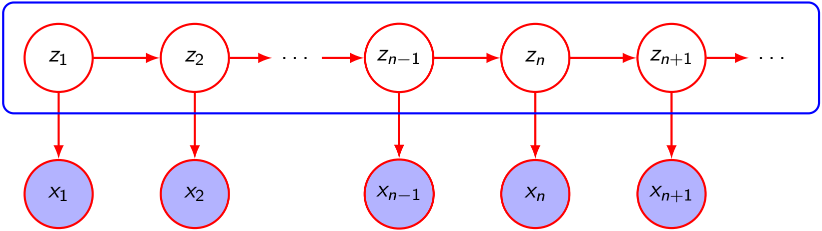
$$P(X, Z|\theta) = p(z_1|\pi) \left[\prod_{n=2}^{N} p(z_n|z_{n-1}, A) \right] \prod_{m=1}^{N} p(x_m|z_m, \phi)$$



HMM Inference: Joint Distribution

$$X = \{x_1, \ldots, x_N\}$$
$$Z = \{z_1, \ldots, z_N\}$$

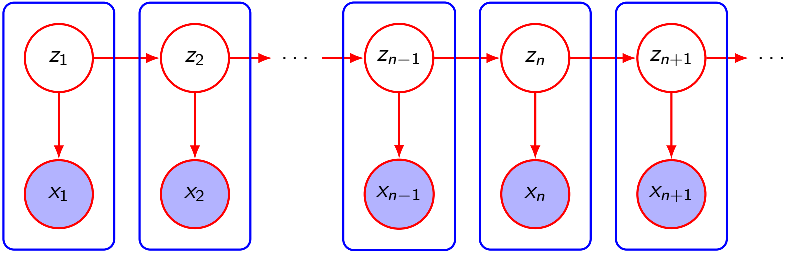
$$P(X, Z|\theta) = p(z_1|\pi) \left[\prod_{n=2}^{N} p(z_n|z_{n-1}, A) \right] \prod_{m=1}^{N} p(x_m|z_m, \phi)$$



HMM Inference: Joint Distribution

$$X = \{x_1, \ldots, x_N\}$$
$$Z = \{z_1, \ldots, z_N\}$$

$$P(X, Z|\theta) = p(z_1|\pi) \left[\prod_{n=2}^{N} p(z_n|z_{n-1}, A) \right] \prod_{m=1}^{N} p(x_m|z_m, \phi)$$

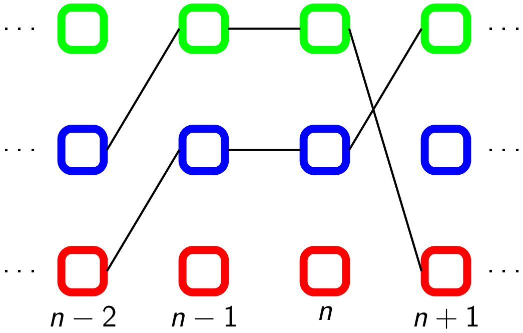


HMM Inference: Likelihood Function

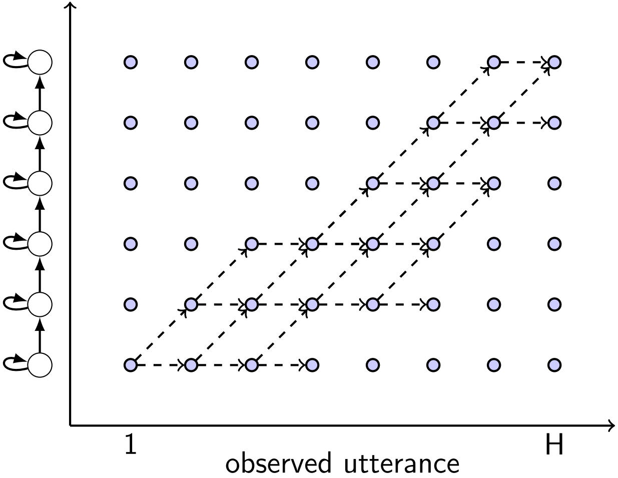
marginalise joint distribution over Z:

$$P(X|\theta) = \sum_{Z} p(X, Z|\theta)$$

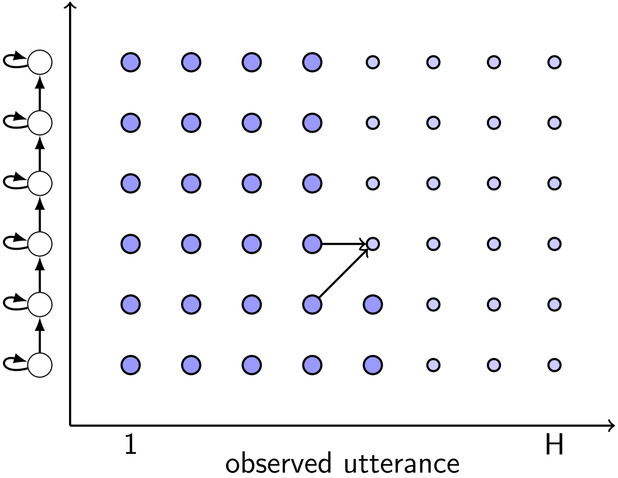
Problem: there are K^N possible sequences for Z



Hidden Markov Models (HMMs)



Hidden Markov Models (HMMs)



Solution: Forward algorithm

Instead of AccD[h,k] (Template Matching)

$$\alpha_n(j) \equiv p(x_1, \ldots, x_n, z_n = s_j)$$

At the end, instead of AccD[H,K]:

$$P(X|\theta) = \sum_{i=1}^{M} \alpha_N(i)$$

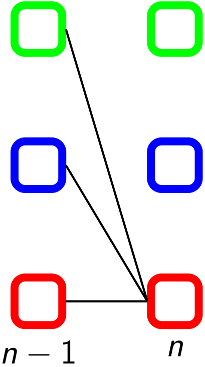
Forward Probability

Initialization:

$$\alpha_1(j) = \pi_j \phi_j(x_1)$$

Recursion:

$$\alpha_n(j) = \left[\sum_{i=1}^{M} \alpha_{n-1}(i) a_{ij} \right] \phi_j(x_n)$$



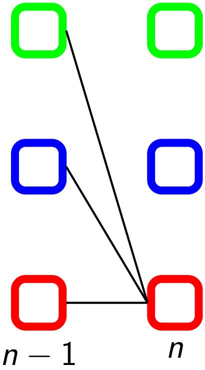
Forward Probability

Initialization:

$$\alpha_1(j) = \pi_j \phi_j(x_1)$$

Recursion:

$$\alpha_n(j) = \left[\sum_{i=1}^{M} \alpha_{n-1}(i) a_{ij} \right] \phi_j(x_n)$$



equivalent to sum-product in Bayesian Networks

Backward probability

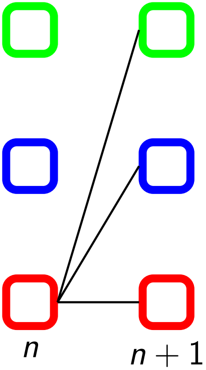
$$\beta_n(i) \equiv p(x_{n+1}, \ldots, x_N | z_n = s_i)$$

Initialization:

$$\beta_N(i) \equiv p(?|z_n = s_i) \equiv 1$$

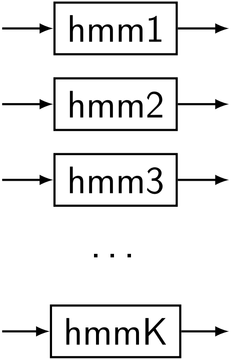
Recursion:

$$\beta_n(i) = \left[\sum_{j=1}^{M} a_{ij} \phi_j(x_{n+1}) \beta_{n+1}(j) \right]$$



Find best sequence of states: why?

Isolated Words



hmm1

hmm2

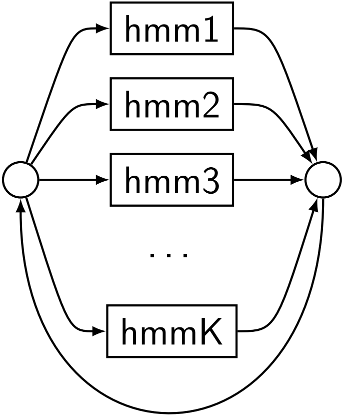
hmm3

. . .

hmmK

(Likelihood)

Continuous Speech



hmm1

hmm2

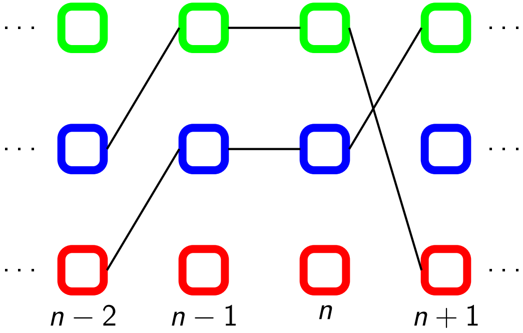
hmm3

. . .

hmmK

Find best sequence of states: how?

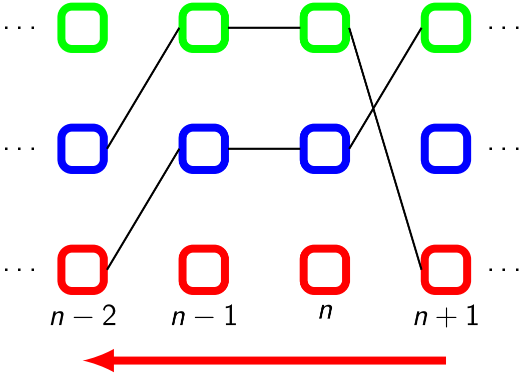
- Viterbi algorithm [3]
- equivalent to max-sum in Bayesian Networks



[3] A. J. Viterbi. "Error Bounds for Convolutional Codes and an Asymtotically optimum decoding algorithm". In: *IEEE Trans. Inform. Theory* IT-13 (Apr. 1967), pp. 260–269

Find best sequence of states: how?

- Viterbi algorithm [3]
- equivalent to max-sum in Bayesian Networks



[3] A. J. Viterbi. "Error Bounds for Convolutional Codes and an Asymtotically optimum decoding algorithm". In: *IEEE Trans. Inform. Theory* IT-13 (Apr. 1967), pp. 260–269

Summary: update rules

Forward algorithm (sum-product):

$$\alpha_n(j) = \left[\sum_{i=1}^{M} \alpha_{n-1}(i) a_{ij}\right] \phi_j(x_n)$$

Viterbi algorithm (max-sum):

$$V_n(j) = \max_{i=1}^{M} \left[V_{n-1}(i) a_{ij}\right] \phi_j(x_n)$$

$$B_n(j) = \arg\max_{i=1}^{M} \left[V_{n-1}(i) a_{ij}\right]$$

Practical Issues

Product of many probabilities: numerical problems
Solution: work in log domain

$$
\begin{aligned}
\alpha'_1(j) &= \pi'_j + \phi'_j(x_1) \\
\alpha'_n(j) &= \log \sum_{i=1}^{M} e^{\left(\alpha'_{n-1}(i) + a'_{ij}\right)} + \phi'_j(x_n) \\
\beta'_N(i) &= 0 \\
\beta'_n(i) &= \log \sum_{j=1}^{M} e^{\left(a'_{ij} + \phi'_j(x_{n+1}) + \beta'_{n+1}(j)\right)}
\end{aligned}
$$

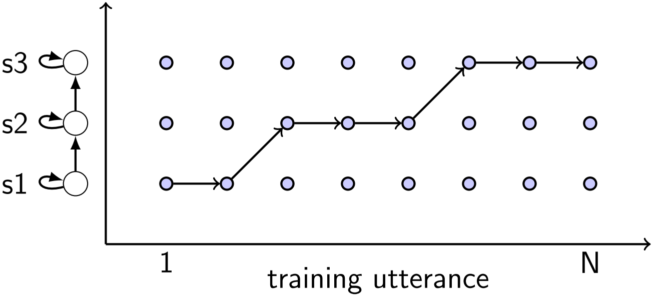
HMM Inference: Learning

- Given observations X update model parameters

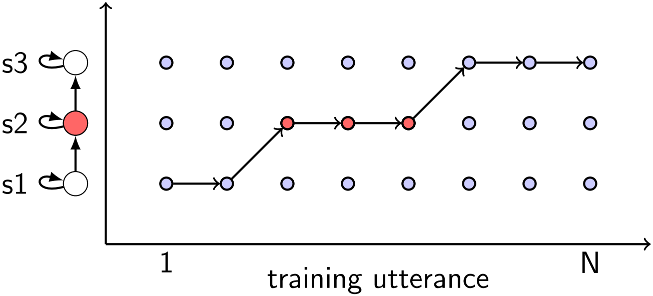
$$\theta = \{\pi, A, \phi\}$$

- to maximise either:
    - model fit to data (e.g. likelihood, posterior)
    - classification performance (discriminative training)
- Incomplete data: state sequence Z

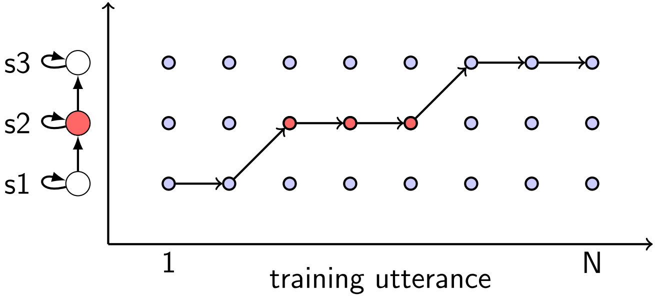
Viterbi training (simple approach)



Viterbi training (simple approach)



Viterbi training (simple approach)



problem: sensitive to misalignments

HMM Inference: Learning

Latent variables → Expectation Maximisation

- locally maximise the likelihood
- close form and efficient solution with froward-backward or Baum-Welch algorithm [1]
- general idea: sum over all possible paths weighted by probability of the path
- also: every observation vector contributes to all parameter updates

[1] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains". In: *Ann. Math. Statist.* 41.1 (1970), pp. 164–171

Example: Transition Probability

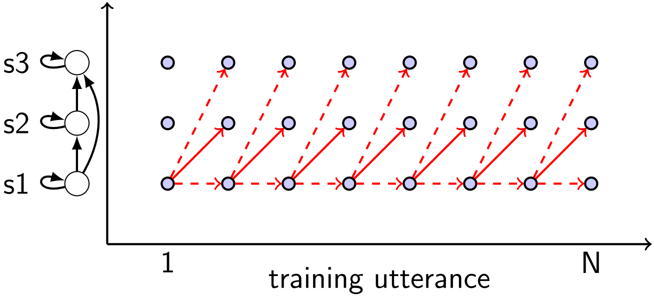
We want to update

$$a_{ij} = P(z_n = s_j | z_{n-1} = s_i)$$

Define $\gamma_n(i, j)$ as the probability of making a transition from s_i to s_j at time step n given the current model parameters θ and the current observation sequence X

$$\gamma_n(i, j) = P(z_{n-1} = s_i, z_n = s_j | X, \theta)$$

Example: Transition Probability



$$a_{12}^{\text{new}} = \frac{E\left[s_1 \rightarrow s_2 | X, \theta\right]}{E\left[s_1 \rightarrow s_{\text{any}} | X, \theta\right]} = \frac{\sum_{n=1}^{N} \gamma_n(1, 2)}{\sum_{n=1}^{N} \sum_{k=1}^{3} \gamma_n(1, k)}$$

Example: Transition Probability

- $\sum_{n=1}^{N} \gamma_n(i,j)$ is the expected number of transitions between state s_i and s_j (given X and θ)
- we never take a hard decision on when the transition happened

Calculate Gamma (DP)

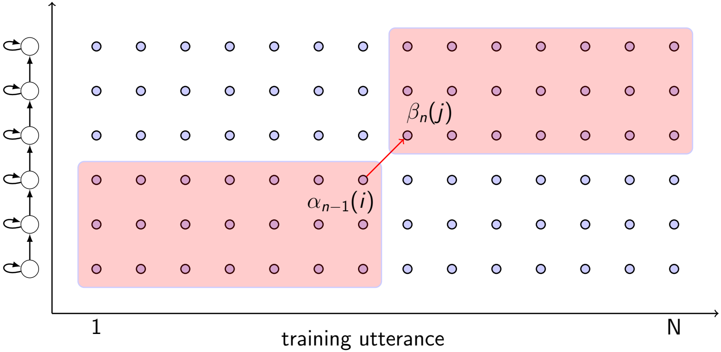
$$\gamma_n(i,j) \quad = \quad P(z_n = s_j, z_{n-1} = s_i | X, \theta)$$

Calculate Gamma (DP)

$$\begin{aligned}\gamma_n(i,j) &= P(z_n = s_j, z_{n-1} = s_i | X, \theta) \\ \text{(Bayes)} &= \frac{P(z_n = s_j, z_{n-1} = s_i, X | \theta)}{P(X|\theta)}\end{aligned}$$

Calculate Gamma (DP)

$$
\begin{aligned}
\gamma_n(i,j) &= P(z_n = s_j, z_{n-1} = s_i | X, \theta) \\
\text{(Bayes)} &= \frac{P(z_n = s_j, z_{n-1} = s_i, X | \theta)}{P(X|\theta)} \\
&= \frac{\alpha_{n-1}(i)\ a_{ij}\ \phi_j(x_n)\ \beta_n(j)}{\sum_{k=1}^{M} \alpha_N(k)}
\end{aligned}
$$



Baum-Welch: Properties

instance of Expectation Maximisation:

- iterative procedure
- guaranteed to convert to local maximum of the likelihood $P(X|\theta^{\text{new}})$
- sensitive to initialisation
- update formulae depend on state to output probability model $\phi_j(x_n)$