Hidden Markov Models

Hauptseminar Machine Learning

18.11.2003

Referent: Nikolas Dörfler

Overview

Markov Models

Hidden Markov Models

Types of Hidden Markov Models

Applications using HMMs

Three central problems:

Evaluation: Forward algorithm

Backward algorithm

Decoding: Viterbi algorithm

Learning: Forward-Backward Algorithm

Speech recogition with HMM

Isolated word recognizer

Measured performance

Conclusion

Markov Models

- A System of n states : the system is at time t in state □(t)

- Changes of state are nondeterministic but depending on the previous states

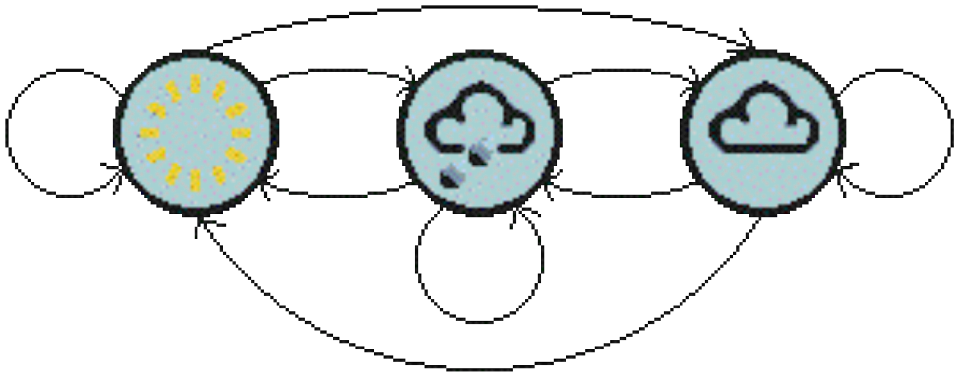
  In a First order, ..., n-order model changes depend on the previous 1, ..., n states

- Transition probabilities: Often shown as state transition matrix

- Vector of initial probabilities π

Example: Weather system

Three states: sunny, cloudy, rainy



Transition Matrix A

$$A = \begin{bmatrix} 0,5 & 0,25 & 0,25 \\ 0,375 & 0,125 & 0,375 \\ 0,125 & 0,625 & 0,375 \end{bmatrix}$$

State vector

$\pi = (1,0,0)^T$

Hidden Markov Models

- n invisible states

- every state emits at time t a visible symbol/state $v(t)$

- System generates a sequence of symbols (states) $V^T = \{ v(1), v(2), v(3), ..., v(n)\}$

- Transition probability: $P(\omega(t+1)=j \mid \omega(t)=i) = a_{ij}$

  = Transition matrix A

- Probability of emitting $v(t)$ in state $\omega(t)=j$: $P(v(t) \mid \omega(t)=j) = b_j(k)$

  = Confusion matrix B

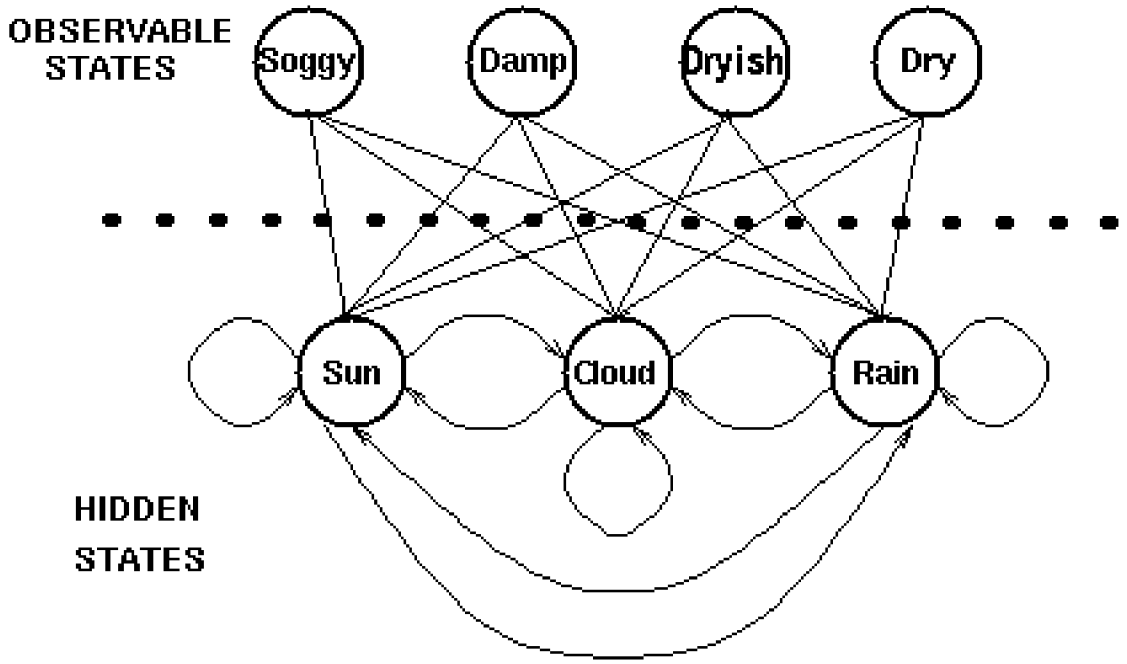
- Vector of initial probabilities π

- Normalization conditions

$$\sum_j a_{ij} = 1 \quad \text{for all i} \qquad \sum_j b_j(k) = 1 \quad \text{for all k}$$

Example of a hidden Markov model:

Indirect observation of the weather using a piece of seaweed



Transition matrix A

	sun	clouds	rain
sun	0,5	0,25	0,25
clouds	0,375	0,125	0,375
rain	0,125	0,625	0,375

Confusion matrix B

	Dry	Dryish	Damp	Soggy
sun	0,6	0,2	0,15	0,05
clouds	0,25	0,25	0,25	0,25
rain	0,05	0,1	0,35	0,5

State vector $\pi = (1,0,0)^T$

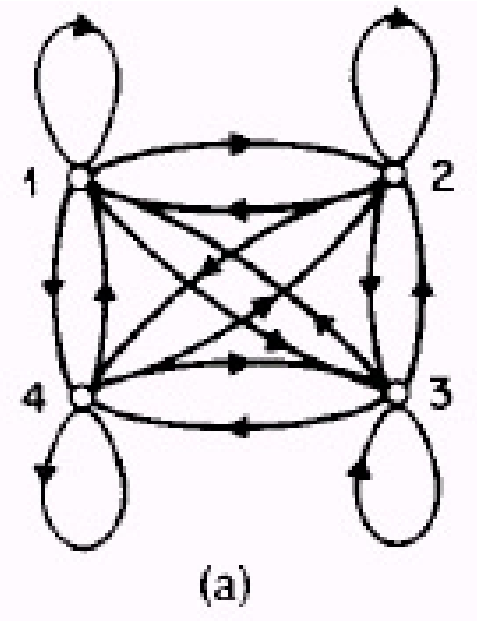
Types of Hidden Markov Models

Ergodic

All states can be reached within one step from everywhere

Transition matrix A entries are never zero

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$



Left-right Models

No backleading transitions

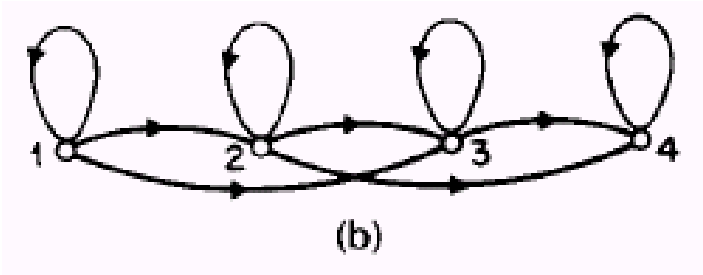
$a_{ij} = 0 \quad j < i$

Additional constraints:

$a_{ij} = 0 \quad j > i + \Delta$

(f.e. $\Delta = 2$ => no jumps of more than 2 states)

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}$$



(b)

Applications using HMM

Speech recognition

Language modelling

Protein sequence analysis

Recognition of hand writings

Financial/Economic Models

Three central problems

1. Evaluation

Given a HMM (A,B,π) :

Find the probability that a sequence of visible states V^T was generated by that model

2. Decoding

Given a HMM (A,B,π) and a set of observations:

Find the most probable sequence of hidden states that led to those observations

3. Learning

Given the number of visible and hidden states and some sequences of training observations:

Find the parameters a_{ij} and $b_j(k)$

Evaluation:

Probability that the system M produces a sequence V^T

$$P(V^T) = \sum_{r=1}^{r_{max}} P(V^T | \omega_r^T) P(\omega_r^T)$$

For all possible sequences $\omega_r^T = \{\omega(1), \omega(2), ..., \omega(T)\}$

That means: Take all sequences of hidden states, calculate the probability that they calculated V^T and add them

$$P(V^T) = \sum_{r=1}^{N} \prod_{t=1}^{T} P(v(t) | \omega(t)) P(\omega(t) | \omega(t-1))$$

N is the number of states , T is the number of visible symbols / steps to go

Problem: Complexity is $o(N^T T)$

The Forward alogithm:

Calculate the Problem recursively

$$\alpha_j(t) = \begin{cases} \pi_j b_j(v(0)) & \text{t=0 and j = initial state} \\ \sum_{i=1}^{N} \alpha_i(t-1) a_{ij} b_j(v(t)) & \text{else} \end{cases}$$

$b_j(v(t))$ means the probability to emit the state selected by v(t)

$\alpha_j(t)$ is the probability that the model is in state j and has produced the first t elements of V^T

Forward algorithm

initialize $\alpha_j(0) = \pi(j) b_j(v(0))$, t=0 a_{ij}, b_j, visible sequence V^T

for t <= t + 1

$$\alpha_j(t) = \sum_{i=1}^{N} \alpha_i(t-1) a_{ij} b_j(v(t)) \quad \text{for all } j \in N$$

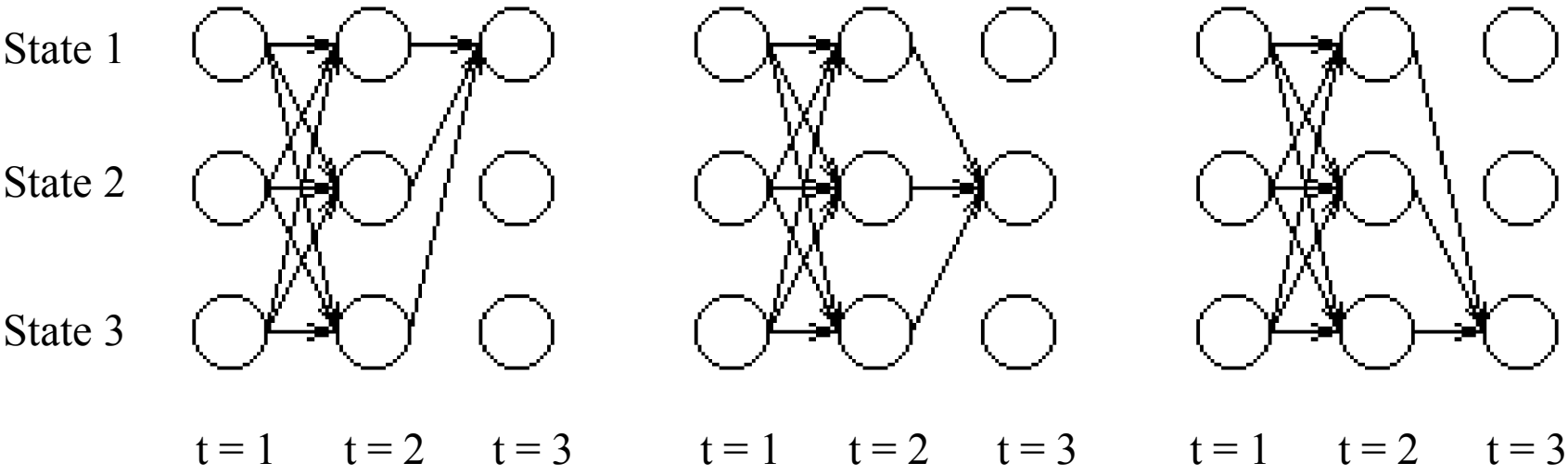
until t = T

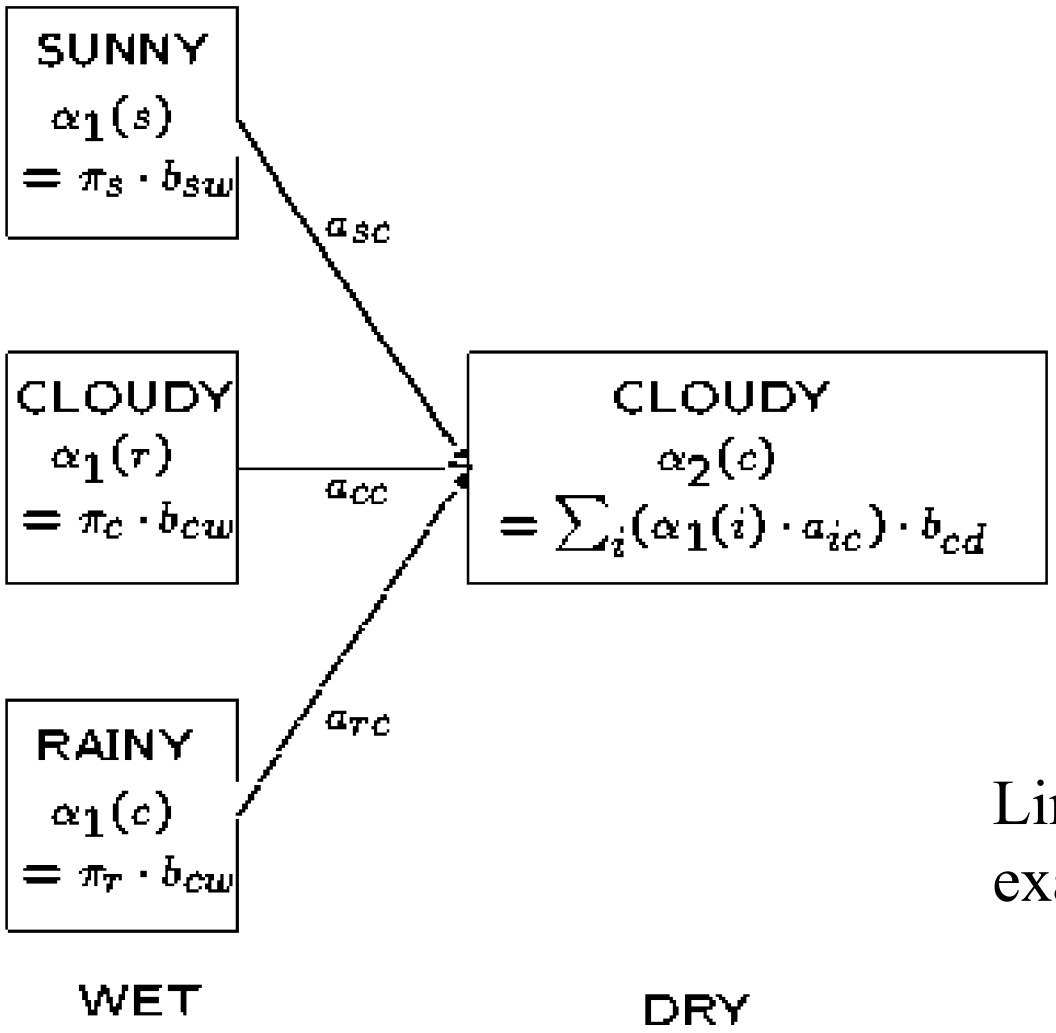
return $P(V^T) = \alpha_{final}(T)$

end

Complexity of this algorithm: $o(N^2T)$

Example: Forward algorithm





Link to Java applet example

Backward algorithm

initialize $\beta_i(T) = 1$, $t = T$ a_{ij}, b_{jk}, visible sequence V^T

for $t \leq t - 1$

$$\beta_j(t) \leftarrow \sum_{i=1}^{N} \beta_i(t+1) a_{ij} b_j(v(t+1)) \quad \text{for all } j < N$$

until $t = 1$

return $P(V^T) = \beta_i(0)$

end

$\beta_i(t)$ is the probability that the model is in state j and will produce the last T - t elements of V^T

Decoding (Viterbi Algorithm)

Finds the sequence of hidden states in a N-state model M that most probable generated a sequence $V^T = \{v(0),v(1),...,v(t)\}$ of visible states.

Can be recursively calculated with the Viterbi algorithm:

$\delta_i(t) = \max P(\Phi(1),..., \Phi(t)=i, v(1),...,v(t) \mid M)$ over all pathes Φ

recursively: $\delta_j(t) = \max_i [\delta_i(t-1)a_{ij}] b_j(v(t))$

with $\delta_j(0) = \pi(i)\, b_i(v(0))$

$\delta_i(t)$ is the maximal probability along a path $\Phi(1),..., \Phi(t) = i$ to generate the sequence v(1),...,v(t) (partial best path)

to keep track of the path maximizing $\delta_i(t)$ the array $\psi_j(t)$ is used

Sequence calculation:

initialize $\delta_i(0) = \pi(i) b_i(v(0))$, $\psi_i(0)=0$, t=0 für alle i

for t <= t + 1

for all states j

$\delta_j(t) = \max_{1 \le i \le N} [\delta_i(t-1) a_{ij}]\ b_j(v(t))$

$\psi_j(t) = \arg \max_{1 \le i \le N} [\delta_i(t-1) a_{ij}]$

until t = T

Sequence Termination:

$\pi(T) = \arg \max_{1 \le i \le N} [\delta_i(t-1) a_{ij}]$

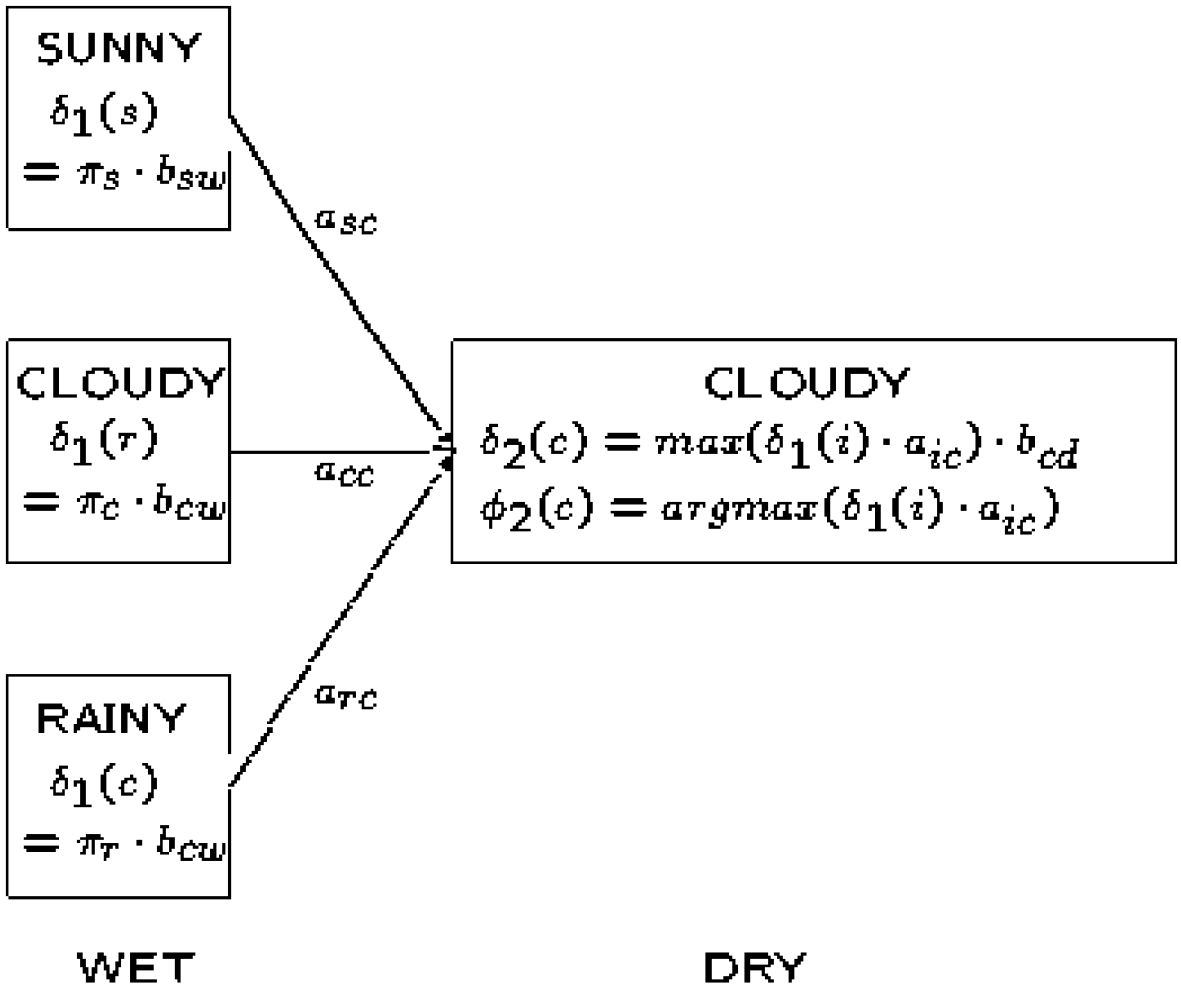
Sequence backtracking:

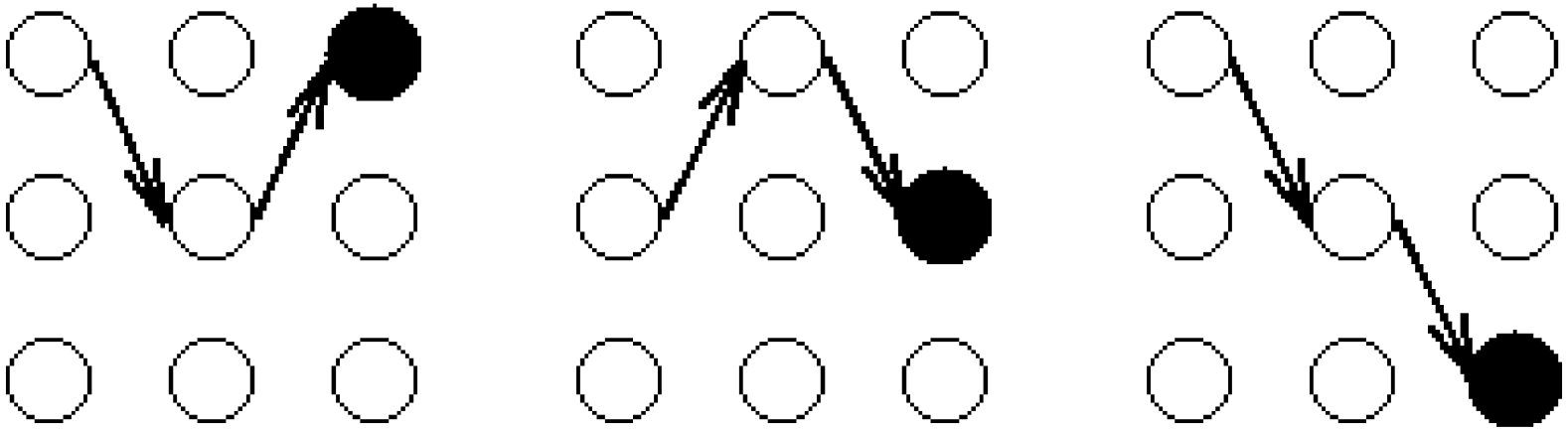
for t = T-1, t <= t - 1

$\pi(t) = \psi_{\pi(t+1)}(t+1)$

until t = 0

Example:


SUNNY
$\delta_1(s)$
$= \pi_s \cdot b_{sw}$
CLOUDY
$\delta_1(r)$
$= \pi_c \cdot b_{cw}$
RAINY
$\delta_1(c)$
$= \pi_r \cdot b_{cw}$
a_{sc}
a_{cc}
a_{rc}
CLOUDY
$\delta_2(c) = max(\delta_1(i) \cdot a_{ic}) \cdot b_{cd}$
$\phi_2(c) = argmax(\delta_1(i) \cdot a_{ic})$
WET
DRY



Link to Java applet
example

Positive aspects of the Viterbi Algorithm

Reduction of computational complexity by recursion

Takes the entire context to find a optimal solution, therefore lower error rates with noisy data

Learning (Forward Backward algorithm)

determine the N-state model parameters a_{ij} and b_{jk} based on a training sequence by iteratively calculating better values

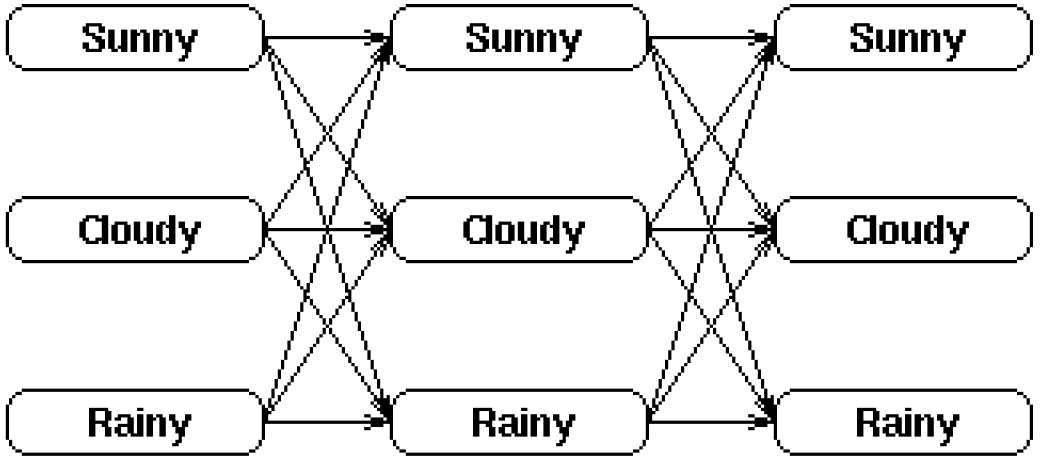
Definition: Probability of a transition $\omega_i(t)$ to $\omega_j(t+1)$ given the model M generated V^T by any path

$$\gamma_{ij}(t) = P(\omega_i(t), \omega_j(t+1) \mid V^T, M)$$

$$\gamma_{ij}(t) = \frac{\alpha_i(t) a_{ij} b_j(v(t+1)) \beta_j(t+1)}{P(V^T \mid M)}$$

$$\gamma_{ij}(t) = \frac{\alpha_i(t) a_{ij} b_j(v(t+1)) \beta_j(t+1)}{\sum_{k=1}^{N} \sum_{l=1}^{N} \alpha_{ki}(t) a_{kl} b_l(v(t+1)) \beta_l(t+1)}$$

$$\gamma_i(t) = \sum_{j=1}^{N} \gamma_{ij}(t)$$ Probability of being in state ω_i at time t



Observations : **dry** **damp** **soggy**

$\sum_{t=1}^{T-1} \gamma_i(t)$ Expected number of transitions from ω_i to any other state

$\sum_{t=1}^{T} \gamma_i(t)$ Expected number of times in ω_i

$\sum_{t=1}^{T-1} \gamma_{ij}(t)$ Expected number of transitions from ω_i to ω_j

which gives us better values for a_{ij}

$$a'_{ij} = \frac{\sum_{t=1}^{T-1} \gamma_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)} = \frac{\sum_{t=1}^{T-1} \gamma_{it}(t) a_{ij} b_i(v(t+1)) \gamma_j(t+1)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \gamma_k(t) a_{ik} b_k(v(t+1)) \gamma_k(t+1)}$$

Calculation of the b_j'

$$b'_j(v_k) = \frac{\sum_{t=1}^{T} \gamma_j(t) b_t(v_k)}{\sum_{t=1}^{T} \gamma_j(t)}$$

Expected number of times in θ_i emitting v_k

Expected number of times in θ_i

$\pi'(i) = \gamma_1(0)$ Probability of being in state i at time 0

Positive Aspect

arbitrary precision of estimation

Problems

How to choose the initial parameters a_{ij} and b_j?

For π and a_{ij} either random or uniform

for the b_j better initial estimates are very useful for fast convergence

estimate these parameters by:

Manual segmentation

Maximum Likelihood segmentation

What is the appropriate model

Must be decided on the kind of signal modeled

In speech recognition often a left-right model is used to model the advancing of time

f.e. every sylable get a state and a final silent state

Scaling

Problem: The α_i and β_i are always smaller than 1 so the calculations converge against zero

This exceeds the precision range even in double precision

Solution: Multiply the α_i and β_i by a scaling coefficient c_t that is independent from i but depends on t

For example:

$$c_t = \frac{1}{\sum_{i=1}^{N} \alpha_i(t)}$$

These parameters fall out in the calculation of a_{ij}‘ and b_j‘

Speech recognizers using HMMs

Isolated Word recognizer

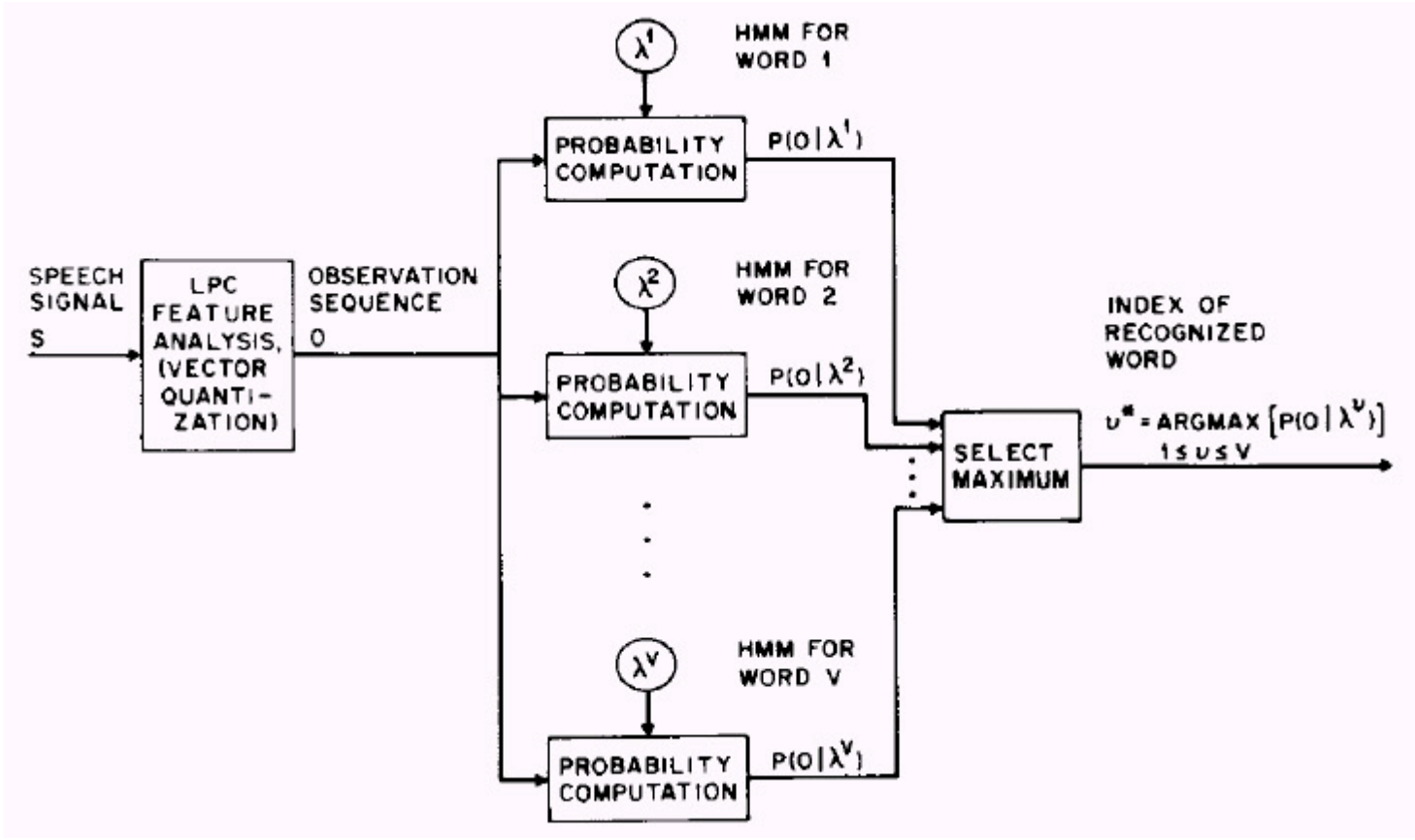
Build a HMM for each word in a vocabulary and calculate the (A,B,π) parameters (train the model)

For each word to be recognized:

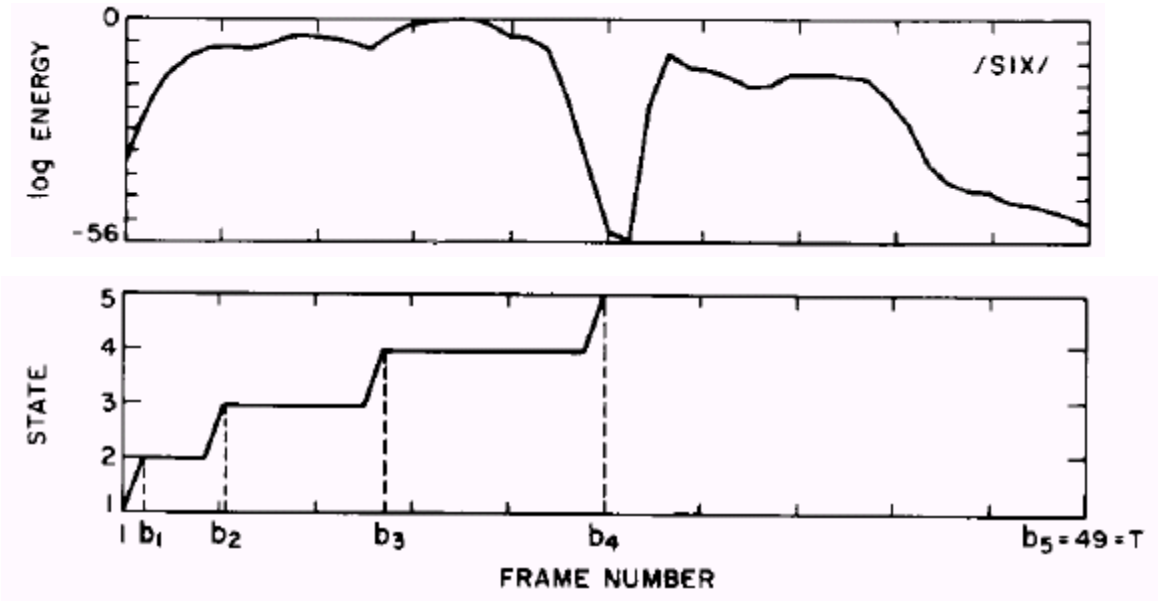
- Feature analysis (Vector quantization)

  Generates Observation Vectors from the signal

- Run Viterbi on all models to find the most probable model for that observation sequence



Block diagram of an isolated word recognizer



a) log energy and b) state assignment for the word ‚six‘

Measured performance of an isolated word recognizer

Table 1 Average Digit Error Rates for Several Recognizers and Evaluation Sets

Recognizer Type	Evaluation Set			
	Original Training	TS2	TS3	TS4
LPC/DTW	0.1	0.2	2.0	1.1
LPC/DTW/VQ	—	3.5	—	—
HMM/VQ	—	3.7	—	—
HMM/CD	0	0.2	1.3	1.8

100 digits by 100 talkers(50 female / 50 male)

Original Training: the original training set was used

TS2: the original speakers as in the training

TS3: Complete new set of speakers

TS4: Another new set of speakers

Conclusion

There are various processes where the real activity is invisible and only a generated pattern can be observed. These can be modeled by HMM

Limitations of HMM :

needs enough training data

The Markov assumption that each state only depends on the previous state is not always true

Advantages

Acceptable calculation complexity

Low error rates

HMM are the predominant method for current automatic speech recognition and play a great role in other recognition systems

Bibliography

Rabiner, L.R.
A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition;
Proceedings of the IEEE, Vol.77, Iss.2, Feb 1989; Pages:257-286

Richard O. Duda, Peter E. Hart, David G. Stork
Pattern Classification chapter 3.10

Eric Keller (Editor)
Fundamentals of Speech synthesis and Speech recognition
Chapter 8 by *Kari Torkkola*

Used Websites

http://www.comp.leeds.ac.uk/roger/HiddenMarkovModels/html_dev/main.html
Introduction to Hidden Markov Models, University of Leeds

http://www.cnel.ufl.edu/~yadu/report1.html
Speech recognition using Hidden Markov Models,
Yadunandana Nagaraja Rao , University of Florida