

Introduction to Graphical Models, Hidden Markov Models and Bayesian Networks

Markus Stengel

Address: Department of Information and Computer Sciences
Toyohashi University of Technology
Toyohashi, 441-8580 Japan

Date: March 7th, 2003

I TABLE OF CONTENTS

| | |
|--|----|
| I Table of Contents..... | 1 |
| II HMM, GM, BN..... | 1 |
| 1 Hidden Markov Model..... | 2 |
| 1.1 Introduction..... | 2 |
| 1.2 Markov Processes..... | 2 |
| 1.2.1 Discrete Time Markov Processes..... | 2 |
| 1.2.2 Limitations of Discrete Time Markov Processes..... | 4 |
| 1.3 Definition of Hidden Markov Models..... | 4 |
| 1.4 Problems for HMMs..... | 5 |
| 1.4.1 Problems..... | 5 |
| 1.4.1.1 Probability Evaluation..... | 5 |
| 1.4.1.2 Optimal State Sequence..... | 6 |
| 1.4.1.3 Parameter Estimation..... | 6 |
| 1.4.2 Solutions..... | 6 |
| 1.4.2.1 Probability Evaluation..... | 6 |
| 1.4.2.2 Optimal State Sequence..... | 8 |
| 1.4.2.3 Parameter Estimation..... | 10 |
| 1.5 Continuous Observation Densities in HMMs..... | 13 |
| 1.6 Types and Classes of HMMs..... | 15 |
| 1.6.1 Types..... | 15 |
| 1.6.1.1 Ergodic HMM..... | 15 |
| 1.6.1.2 Left-Right HMM..... | 16 |
| 1.6.1.3 Variations..... | 17 |
| 1.6.2 Classes..... | 17 |
| 1.6.2.1 Factorial HMMs..... | 17 |
| 1.6.2.2 Tree structured HMMs..... | 18 |
| 1.6.2.3 Autoregressive HMMs..... | 18 |
| 1.6.2.4 Vector Linear Predictor HMMs..... | 19 |
| 1.7 Optimization Criteria..... | 20 |
| 1.7.1 Maximum Likelihood (ML)..... | 20 |
| 1.7.2 Minimum Discrimination Information (MDI)..... | 20 |
| 1.7.3 Maximum Mutual Information (MMI)..... | 21 |
| 2 Graphical Models (GM)..... | 23 |
| 2.1 Introduction..... | 23 |
| 2.2 Undirected Graphical Models..... | 23 |
| 2.2.1 Introduction..... | 23 |
| 2.2.2 Independence..... | 23 |
| 2.3 Directed Graphical Models..... | 23 |
| 2.3.1 Introduction..... | 23 |
| 2.3.2 Definitions..... | 24 |

| | |
|---|----|
| 2.3.3 Independence..... | 24 |
| 2.3.4 Conditional Probability Distribution (CPD)..... | 25 |
| 3 Bayesian Networks (BN)..... | 26 |
| 3.1 Introduction..... | 26 |
| 3.2 Why “Bayesian”?..... | 26 |
| 3.3 Conditional Independence and Representation..... | 26 |
| 3.4 Example..... | 27 |
| 3.5 Inference..... | 28 |
| 3.6 Explaining away..... | 29 |
| 3.7 Reasoning..... | 29 |
| 3.8 Causality or correlation?..... | 29 |
| 3.9 Singly connected and multiply connected networks..... | 29 |
| 3.10 Evidence and Belief Propagation..... | 30 |
| 3.11 BN with discrete and continuous nodes..... | 31 |
| 3.12 Dynamic Bayesian Networks (DBN)..... | 31 |
| 3.12.1 Introduction..... | 31 |
| 3.12.2 Strengths and Advantages of DBN..... | 32 |
| 3.12.3 Hidden Markov Models as DBN..... | 32 |
| 3.12.4 Inference..... | 34 |
| 3.12.5 Learning..... | 34 |
| 4 Comments..... | 35 |
| III Index..... | i |
| IV Bibliography..... | iv |
| V Illustrations..... | vi |

II HMM, GM, BN

Introduction

This essay is an excerpt from a report I wrote which covers some of the various topics I studied. Since I felt it might be useful to others as well, I decided to make this part publicly available. Due to its origin there maybe some references to the original report within the text, so don't let this confuse you.

This essay starts with an introduction to *Hidden Markov Models* and continues with a brief explanation of *Graphical Models*. Thereafter, *Bayesian Networks* and their relationship to various other models, such as the Hidden Markov Models, is outlined.

The illustration below might aid in understanding the relationship between hidden Markov models, Graphical Models and Bayesian networks.

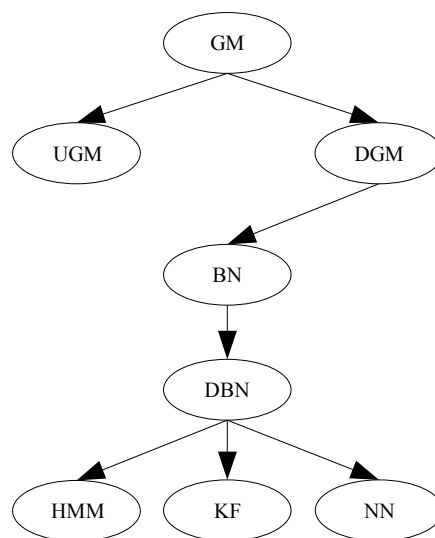


Illustration 1 the relationships between the various models in this essay

where the abbreviations have the following meaning: **GM** Graphical Model, **UGM** Undirected Graphical Model, **DGM** Directed Graphical Model, **BN** Bayesian Network, **DBN** Dynamic Bayesian Network, **HMM** Hidden Markov Model, **KF** Kalman Filter and **NN** Neural Network¹.

¹ though the Kalman Filter and Neural Networks are not covered by this essay, they are put in there to ease the understanding of the relationships between models in the dynamic Bayesian networks framwork

1 Hidden Markov Model

1.1 Introduction

A traditional approach to speech recognition systems is the use of grammars and templates.

The grammar approach one is a very strict approach and only allows limited flexibility of the recognition system, i.e. only paths that are predefined in the grammar can be taken. It explicitly models all possible combinations of patterns, such as words or utterances. Therefore, it can be used alone only with speech that is always pronounced in exactly the same way (same speed, same pitch, etc.). Hence it is rather useful with speech that has been preprocessed and converted to a machine-readable format.

The template approach is based on the idea of deriving typical sequences of speech frame for a pattern, which can be, but is not limited to a e.g. Word, via some averaging procedure [1]. Then a distance measure function can be used to compare patterns and temporally align them to account for e.g. differences in speaking rates across talkers or repetitions of the word by the same talker. The methodology of the template approach is well developed and provides good recognition performance for a variety of practical applications.

The inherent problem of both approaches is lack of flexibility. To date, it has not been possible for anyone to adequately model natural language to the extent that it can be used in *automatic speech recognition systems* (ASR). This already puts a stop on the grammar approach. Furthermore, though the template approach is a simplified, non-parametric method which already contains some inherent statistical signal characterization, due to the way the templates have been collected, it is not sufficient. Particularly, it neglects the very important *second-order statistics*, i.e. *covariances*.

1.2 Markov Processes

1.2.1 Discrete Time Markov Processes

Consider a system that can be in one of a set of N distinct states indexed by $\{1, 2, \dots, N\}$. At regularly spaced, discrete times the system undergoes a change of state (that also includes the same state, a repetition) according to a set of probabilities associated with the state. The time instants associated with state changes are denoted as $t = 1, 2, \dots$. The actual state at time t be denoted as q_t . In general, a full probabilistic description of the system would require specification of the current state at time t , as well as all predecessor states.

Here, however, we limit ourselves to the special case of a discrete time, first order Markov chain. Therefore the probabilistic dependence can be truncated to just the preceding state²:

$$P(q_t | q_{t-1} = i, q_{t-2} = k, \dots, q_1 = l) = P(q_t | q_{t-1} = i) \quad (1)$$

Furthermore, we only consider those processes in which the right-hand side of (1) is independent of time. Thus we get the state-transition probabilities a_{ij} :

$$a_{ij} = P(q_t | q_{t-1} = i), \quad 1 \leq i, j \leq N \quad (2)$$

² **Note:** If we wish to model a second, third, ... order Markov process, we need to take the corresponding amount of preceding states into account (2, 3, ...). n order Markov properties means that the current state q_t is dependent on only the last n preceding states

Since the a_{ij} obey stochastic constraints, they have the following properties:

$$a_{ij} \geq 0, \quad \forall j, i \quad (3)$$

and

$$\sum_{j=1}^N a_{ij} = 1, \quad \forall i \quad (4)$$

Because the output of the process is the set of states at each instant of time and, therefore, each state corresponds to an observable event, this stochastic process could be called an *observable Markov model*.

A simple example of this model is the following three-state Markov model of the weather (Illustration 2). We assume that once a day the weather is observed as being one of the following:

- State 1: rain or snow
- State 2: cloudy
- State 3: sunny

We demand that the weather on day t is characterized by a single one of the three states above, and that the matrix A of state-transition probabilities a_{ij} is:

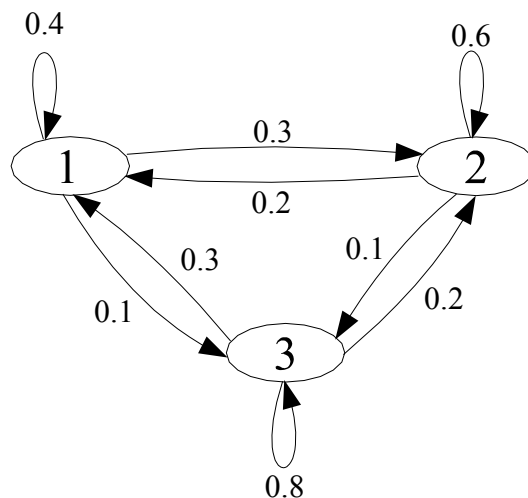


Illustration 2: Markov model of the weather

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

Given the model of Illustration 2, we can now answer questions about the behavior of weather patterns over time. For example, “What is the probability that the weather for eight consecutive days is sun-sun-sun-rain-rain-sun-cloudy-sun?”:

We define the *observation sequence* O as $O = (3, 3, 3, 1, 1, 3, 2, 3)$ and the probability distribution over the initial state π as

$$\pi_i = P(q_1 = i), \quad 1 \leq i \leq N \quad (5)$$

Then we can directly determine $P(O|Model)$, assuming we assign $\pi = (0, 0, 1)$:

$$\begin{aligned} & P(O|Model) \\ &= P(3, 3, 3, 1, 1, 3, 2, 3|Model) \\ &= P(3) P(3|3) P(3|3) P(1|3) P(1|1) P(3|1) P(2|3) P(3|2) \\ &= \pi_3 * (a_{33})^2 * a_{31} * a_{11} * a_{13} * a_{32} * a_{23} \\ &= 1.536 * 10^{-4} \end{aligned} \quad (6)$$

1.2.2 Limitations of Discrete Time Markov Processes

Markov models, in which each state corresponds to a deterministically observable event, cannot model random output since the output is tied to the state. These models are far too restrictive too many problems of interest, such as speech recognition.

1.3 Definition of Hidden Markov Models

Hidden Markov Models (HMMs) are a frequently used tool for modeling time series data. They are used in numerous applications such as image recognition, pattern recognition, data compression and speech recognition. It can represent probability distributions over sequences of observations:

Since discrete time Markov processes are too limited and too restrictive, we extend the concept of Markov models in the way that the observation is a probabilistic function of the state. The resulting model is a doubly embedded stochastic process with an underlying stochastic process that is **not** directly observable; it is *hidden*. The hidden states are the true states of the described system. They are assumed to be *discrete*³ and can only be observed through another set of stochastic processes that produce the sequence of observations. It is important to note that the number of observed states may differ from the number of hidden states. Furthermore, we assume that the state of this hidden process satisfies the Markov property⁴. The outputs also satisfy a Markov property with respect to the states: Given q_t , o_t is independent of the states and observations at all other time indices.

Taken together, these Markov properties mean that the joint distribution of a sequence of states and observations can be factored in the following way⁵ [2]:

$$P(Q_{1:t}, O_{1:t}) = P(q_1) P(o_1|q_1) \prod_{i=2}^T P(q_i|q_{i-1}) P(o_i|q_i) \quad (7)$$

3 Q_t can take on N values which will be denoted by the integers $\{1, \dots, N\}$

4 usually the first-order Markov property, but higher orders can also be used

5 for notational convenience $O_{1:t}$ denotes o_1, \dots, o_t

To define a probability distribution over sequences of observations, all that is left to specify is the probability distribution over the initial state π as in (5), the $N \times N$ state transition matrix A defining $P(q_t|q_{t-1})$ and the output model defining $P(o_t|q_t)$. It is usually assumed that the state transition matrices as the output models are – except for the initial state – *time invariant*⁶. Therefore, if the observables are discrete symbols which can take on one of M values, the output model can be fully specified by a $N \times M$ *observation matrix* B , which is also called the *emission matrix*. If the observation vectors are real-valued, $P(o_t|q_t)$ can be modeled in many different forms, e.g. *Gaussian*⁷, *mixture of Gaussians*, *neural network* etc.

Therefore, an HMM model can be fully specified by the following parameter set:

$$\lambda = (N, M, A, B, \pi) \quad (8)$$

For convenience, a more frequently used compact notation is

$$\lambda = (A, B, \pi) \quad (9)$$

In fact, this parameter set defines a probability measure for O , i.e. $P(O|\lambda)$. So when we use the terminology HMM we are referring to the parameter set λ and its associated probability measure.

1.4 Problems for HMMs

1.4.1 Problems

Once a system can be described by an HMM, three problems need to be addressed:

1.4.1.1 Probability Evaluation

Abstract problem

We have a number of HMMs⁸ – a set of (A, B, π) triples – which describe different systems, and a sequence of observations $O_{1:T}$. How can we find out efficiently which HMM most probably generated the sequence?

Example

For example, let's use the weather example as given in section 1.2.1. We extend it in the following way: Depending on the season, the weather transition differs. So we have four hidden states, one for each season: spring, summer, autumn, winter. We now want to determine the season based on the observation of the weather.

Occurrence

This sort of problem occurs in speech recognition when a large number of Markov models is used, i.e. each modeling a particular word. So after the HMM best fitting to an observation sequence formed from a spoken word is identified, the word is identified as well.

⁶ not dependent on t

⁷ for high-dimensional real-valued observations, a very useful model can be obtained by replacing the Gaussian by a factor analyzer (FA) [3]

⁸ a set of λ

1.4.1.2 Optimal State Sequence

Abstract problem

Given the observation sequence $O_{1:T}$ and the HMM λ , how do we choose a corresponding state sequence $Q_{1:T}$ that best explains the observations?

Example

Consider the extended weather example from above. We want to know sequence of the seasons (e.g.: is it winter, autumn, spring, summer?)

Occurrence

This is the probably most important part of a speech recognition system, since this is where the actual recognition takes place. It can also be used, as in e.g. in Natural Language Processing, to tag words with their syntactic class. Then the words in the sentences are the observables, and the syntactic classes are their hidden states.

1.4.1.3 Parameter Estimation

Abstract problem

Given the observation sequence $O_{1:T}$ and the HMM λ , how do we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$?

Example

As before, the extended weather example from above shall be used: After using our system for some time we realized, that it is not as precise as we want it to be. So we want to readjust the probabilities of our weather prediction system.

Occurrence

This is most common when training the speech recognition system.

1.4.2 Solutions

1.4.2.1 Probability Evaluation

Exhaustive search for solution

The most straightforward way to calculate the probability of the observation sequence $O_{1:T}$ given the model $\lambda = (A, B, \pi)$ is through enumeration of every possible state sequence of length T , the number of observations. Since there are N possible transitions from each state to another, there are N^T such state sequences.

If given one such fixed state sequence $S_{1:T}$, where s_1 is the initial state, assuming statistical independence of observations the probability of the state sequence is

$$P(O|S, \lambda) = \prod_{t=1}^T P(o_t | s_t, \lambda) \quad (10)$$

$$P(O|S, \lambda) = b_{s_1}(o_1) * b_{s_2}(o_2) * \dots * b_{s_t}(o_t) \quad (11)$$

The probability of such a state sequence S can be written as

$$P(S|\lambda) = \pi_{s_1} * a_{s_1 s_2} * a_{s_2 s_3} * \dots * a_{s_{T-1} s_T} \quad (12)$$

with the joint probability of O and q

$$P(O, S|\lambda) = P(O|s, \lambda) P(S|\lambda) \quad (13)$$

Hence the probability of O is obtained by summing this joint probability over all possible state sequences S , giving

$$P(O|\lambda) = \sum_{all S} P(O|S, \lambda) P(S|\lambda) \quad (14)$$

This way of calculating is computationally expensive, if not even infeasible. Therefore a more effective algorithm is required.

The Forward Algorithm

We can use the time invariance of the probabilities to reduce the complexity of the problem. Consider the forward variable $\alpha_t(i)$ defined as

$$\alpha_t(i) = P(o_1 o_2 \dots o_t, q_t = i | \lambda) \quad (15)$$

$\alpha_t(i)$ is the probability of the *partial observation sequence* $O_{1:t}$ and state i at time t , given the model λ . In other words, these *partial probabilities* represent the probability of getting to a particular state q at time t . This way we can reduce the complexity of calculating (14).

$\alpha_t(i)$ can be solved inductively as follows:

1. Initialization

$$\alpha_1(i) = \pi_1 b_i(o_1), \quad 1 \leq i \leq N \quad (16)$$

2. Induction

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(o_{t+1}), \quad 1 \leq t \leq T-1, 1 \leq j \leq N \quad (17)$$

3. Termination

$$P(O_{1:T}|\lambda) = \sum_{i=1}^N \alpha_T(i) \quad (18)$$

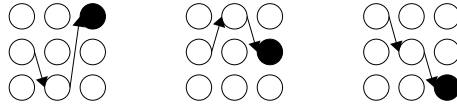


Illustration 3 partial best paths in a simplified 3-state model for the first 3 observations

The basic idea of the forward algorithm, also called the *forward procedure* [1], is to exploit recursion in the calculations to avoid the necessity for exhaustive calculation of all paths. The probability of the observation sequence is found by calculating the partial probabilities at time $t=1, 2, \dots, T$, and adding all α 's at $t=T$.

Using the forward algorithm, it is straightforward to determine e.g. which of a number of HMMs best describes a given observation sequence - the forward algorithm is evaluated for each, and that giving the highest probability selected.

1.4.2.2 Optimal State Sequence

Unlike the probability evaluation in 1.4.2.1, there are several ways of solving this problem. Finding the optimal state sequence first requires definition of what an “optimal state sequence” is.

For example, one could maximize the number of states that are **individually** most likely to occur at each time t . This optimality criterion maximizes the expected number of correct individual states. Though this seems a viable solution at first glance, given some thought it quickly becomes obvious that there could be some problem with the resulting state sequence. For example, if the HMM has state transition which have probabilities of zero, i.e. those transitions are not possible, the resulting state sequence might not even be a valid one. Naturally, this problem can be eased through various ways, e.g. one could solve for the state sequence that maximizes the expected number of correct tuples of states. However, the most widely used criterion is to find the **single** best state sequence Q , which means maximizing $P(Q|O,\lambda)=P(Q,O|\lambda)$. For this task the Viterbi algorithm [1][4] is used.

Viterbi Algorithm

We first define the *partial probability* δ , which is the probability of reaching a particular intermediate state in the lattice of observations and states:

$$\delta_t(i) = \max_{q_1, q_2, \dots, q_{t-1}} \{P(q_1 q_2 \dots q_{t-1}, q_t=i, o_1 o_2 \dots o_t | \lambda)\} \quad (19)$$

Thus δ is the maximum probability of all sequences ending at state i at time t . Furthermore, the partial best path is the sequence which achieves this maximum probability. In a simplified 3-state model for the first 3 observations this would like this:

By induction we have

$$\delta_{t+1}(j) = (\max_i \{\delta_t(i) a_{ij}\}) * b_j(o_{t+1}) \quad (20)$$

In order to retrieve the state sequences, we need to keep track of the argument that maximized (20), for each t and j . This is achieved by using an array $\psi_t(j)$. The complete Viterbi algorithm is listed below:

1. Initialization

$$\begin{aligned} \delta_1(i) &= \pi_i b_i(o_1), & 1 \leq i \leq N \\ \psi_1(i) &= 0 \end{aligned} \quad (21)$$

2. Recursion

$$\begin{aligned} \delta_t(j) &= (\max_{1 \leq i \leq N} \{\delta_{t-1}(i) a_{ij}\}) * b_j(o_t), & 2 \leq t \leq T, 1 \leq j \leq N \\ \psi_t(j) &= \arg \max_{1 \leq i \leq N} \{\delta_{t-1}(i) a_{ij}\}, & 2 \leq t \leq T, 1 \leq j \leq N \end{aligned} \quad (22)$$

3. Termination

$$\begin{aligned} P^* &= \max_{1 \leq i \leq N} \{\delta_T(i)\} \\ q_T^* &= \arg \max_{1 \leq i \leq N} \{\delta_T(i)\} \end{aligned} \quad (23)$$

4. Path (state sequence) backtracking

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \dots, 1 \quad (24)$$

Except for the backtracking step, the Viterbi algorithm is similar in implementation to the forward calculation of (16)-(18). The main difference is the maximization over previous states in (22) that is used instead of the summation in (17).

Alternative Viterbi Implementation

By taking logarithms of the model parameters, the above Viterbi algorithm can be implemented without the need for any multiplications:

0. Preprocessing

$$\begin{aligned} \tilde{\pi}_i &= \log(\pi_i), & 1 \leq i \leq N \\ \tilde{b}_i(o_t) &= \log(b_i(o_t)), & 1 \leq i \leq N, 1 \leq t \leq T \\ \tilde{a}_{ij} &= \log(a_{ij}), & 1 \leq i, j \leq N \end{aligned} \quad (25)$$

1. Initialization

$$\begin{aligned} \tilde{\delta}_1(i) &= \log(\delta_1(i)) = \tilde{\pi}_i + \tilde{b}_i(o_1), & 1 \leq i \leq N \\ \psi_1(i) &= 0, & 1 \leq i \leq N \end{aligned} \quad (26)$$

2. Recursion

$$\begin{aligned}\tilde{\delta}_t(j) &= \log(\delta_t(j)) = \max_{1 \leq i \leq N} \{\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij}\} + \tilde{b}_j(o_t) \\ \psi_t(j) &= \arg \max_{1 \leq i \leq N} \{\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij}\}, \quad 2 \leq t \leq T, 1 \leq j \leq N\end{aligned}\quad (27)$$

3. Termination

$$\begin{aligned}\tilde{P}^* &= \max_{1 \leq i \leq N} \{\tilde{\delta}_T(i)\} \\ q_T^* &= \arg \max_{1 \leq i \leq N} \{\tilde{\delta}_T(i)\}\end{aligned}\quad (28)$$

4. Backtracking

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \dots, 1 \quad (29)$$

1.4.2.3 Parameter Estimation

Also called the *learning problem*, this is the most difficult problem. It stems from the fact that there are many circumstances in practical problems where the model parameters cannot be directly estimated. Though there is no known way to analytically determine the model parameter set that maximizes the probability of the observation sequence in a closed form [1], we can choose $\lambda = (A, B, \pi)$ such that its likelihood $P(O|\lambda)$ is locally maximized using an iterative procedure such as the *Baum-Welch method*⁹ [5] or using *gradient techniques* [6].

Here the iterative EM algorithm will be explained. First though, we need to define the *backward procedure*:

Backward Procedure

Similar to the forward procedure in 1.4.2.1, we define the *backward variable* $\beta_t(i)$, which is the probability of the partial observation sequence from $t+1$ to the end, given state i at time t and the model λ :

$$\beta_t = P(o_{t+1} o_{t+2} \dots o_T | q_t = i, \lambda) \quad (30)$$

$\beta_t(i)$ can then be solved inductively¹⁰:

1. Initialization

$$\beta_T(i) = 1, \quad 1 \leq i \leq N \quad (31)$$

2. Induction

$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \quad (32)$$

⁹ also known as the *expectation-maximization method (EM)*

¹⁰ in the initialization step, $\beta_T(i)$ is initialized with an arbitrary value

Further required definitions

We define the probability of being in state i at time t , and in state j at time $t+1$, given O and λ :

$$\xi_t(i, j) = P(q_t = i, q_{t+1} = j | O, \lambda) \quad (33)$$

Using the definition of the forward and the backward variables, we can write $\xi_t(i, j)$ in the form

$$\begin{aligned} \xi_t(i, j) &= \frac{P(q_t = i, q_{t+1} = j, O | \lambda)}{P(O | \lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^N \sum_{j=1}^N \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)} \end{aligned} \quad (34)$$

Given $\xi_t(i, j)$, we can define $\gamma_t(i)$, the probability of being in state i at time t , given the entire observation sequence and the model:

$$\gamma_t(i) = \sum_{j=1}^N \xi_t(i, j) \quad (35)$$

By summing over $\gamma_t(i)$ and $\xi_t(i, j)$, we can get the two quantities:

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{expected number of transitions from state } i \text{ in } O \quad (36)$$

$$\sum_{t=1}^{T-1} \xi_t(i, j) = \text{expected number of transitions from state } i \text{ to state } j \text{ in } O \quad (37)$$

Now we are ready to define the *reestimation formulas* which are known as

The Forward-Backward Algorithm

If we define the model $\lambda = (A, B, \pi)$, we can compute the reestimated model $\lambda^* = (A^*, B^*, \pi^*)$ using following reestimation formulas:

$$\pi^* = \gamma_1(i) \quad (38)$$

$$a_{ij}^* = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \quad (39)$$

$$b_j^*(k) = \frac{\sum_{t=1}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)}, \quad v_k = \text{observed symbol} \quad (40)$$

The final result of this reestimation procedure is a *Maximum Likelihood (ML)* estimate of the HMM. It is important to realize that the forward-backward algorithm leads to *local maxima* only. Since the likelihood function is often very complex and can have many local maxima, this needs to be taken into account.

Baum's Auxiliary Function

Instead of deriving from the forward and backward procedure, one can also find the reestimation formulas (38)-(40) by maximizing *Baum's auxiliary function* over λ :

$$Q(\lambda^*, \lambda) = \sum_q P(O, q | \lambda^*) * \log(P(O, q | \lambda)) \quad (41)$$

Baum's auxiliary function's advantage to gradient techniques is that it provides **monotonic** improvement in the likelihood, whilst gradient techniques are not guaranteed to do so [1].

Stochastic constraints

An important property of the reestimation procedure is that the stochastic constraints of the HMM parameters

$$\sum_{i=1}^N \pi_i^* = 1 \quad (42)$$

$$\sum_{j=1}^N a_{ij}^* = 1, \quad 1 \leq i \leq N \quad (43)$$

$$\sum_{k=1}^M b_j^*(k) = 1, \quad 1 \leq j \leq N \quad (44)$$

are automatically incorporated at each iteration.

1.5 Continuous Observation Densities in HMMs

Since to this point the observations were characterized as discrete symbols chosen from a finite alphabet, a discrete probability density within each could be used. However, sometimes the observations are *continuous* signals or vectors. Using *vector quantization codebooks* and other methods, one can convert, such observations into a sequence of discrete symbols. However, an improper or not so well fit discretization might lead to degradation. Therefore it would be advantageous to model continuous signal representations directly.

In order to use a continuous observation density, the form of the model *probability density function (pdf)* needs to be restricted to ensure that the parameters of the pdf can be reestimated in a consistent way. The most general representation of the pdf, for which a reestimation procedure has been formulated [1], is a finite *mixture* of the form

$$b_j(\langle o \rangle) = \sum_{k=1}^M c_{jk} N^*(\langle o \rangle, \mu_{jk}, U_{jk}), \quad 1 \leq j \leq N \quad (45)$$

where $\langle o \rangle$ is the *observation vector*¹¹ that is modeled, c_{jk} are the *mixture coefficients*, or *mixture gains*, for the k th mixture in state j , and N^* ¹² is any log-concave or elliptically symmetric density [6]. The most frequently used density is *Gaussian*¹³. To ease the understanding, first the definition of the one-dimensional case [7] is given:

$$\hat{N}(x, a, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-a)^2}{2\sigma^2}}, \quad x \in \mathbb{R}, a \in \mathbb{R}, \sigma^2 \in \mathbb{R}_+^* \quad (46)$$

Here, x is a real-valued *random variable*, a is the *mean value* and σ^2 is the *variance*. In the case of $a=0$ and $\sigma^2=1$, this is called the *standard normal distribution* or, equivalently, the *standard Gaussian distribution*. In the multidimensional case, which is the usual used one in ASR systems, the *multivariate Gaussian distribution* is defined as

$$\hat{N}(X, A, \Sigma^2) = \frac{1}{\prod_{i=1}^n \sqrt{2\pi\sigma_i^2}} e^{-\sum_{i=1}^n \frac{(x_i - a_i)^2}{2\sigma_i^2}}, \quad X \in \mathbb{R}^n, A \in \mathbb{R}^n, \Sigma^2 \in \mathbb{R}_+^{*n} \quad (47)$$

For clarification: $X=(x_1, x_2, \dots, x_n)$, $A=(a_1, a_2, \dots, a_n)$ and $\Sigma^2=(\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2)$.

In the following, it is assumed without loss of generality that N^* in (45) is Gaussian with mean vector μ_{jk} and covariance matrix U_{jk} for the k th mixture component in state j . Since the mixture gains c_{jk} satisfy the stochastic constraint

11 The **observation vector** differs from the **observation sequence** in the way that the entries in the observation vector all represent observations **at the same time**, whilst entries in the observation sequence vector represent **consecutive** observations. Thus an observation sequence O can have observation vectors as its components.

12 not to be confused with N

13 also known as *normal distribution*

$$\begin{aligned} \sum_{k=1}^M c_{jk} &= 1, \quad 1 \leq j \leq N \\ c_{jk} &\geq 0, \quad 1 \leq j \leq N, 1 \leq k \leq M \end{aligned} \quad (48)$$

the pdf is properly normalized:

$$\int_{-\infty}^{\infty} b_j(\langle o \rangle) do = 1 \quad (49)$$

To denote the reestimation formulas for the pdf, we first need to define γ_t^{jk} , the probability of being in state j at time t with the k th mixture component accounting for $\langle o \rangle_t$, the observation vector at time t :

$$\gamma_t(j, k) = \frac{\alpha_t(j) \beta_t(j)}{\sum_{j=1}^N \alpha_t(j) \beta_t(j)} \left[\frac{c_{jk} \dot{N}(\langle o \rangle_t, \mu_{jk}, U_{jk})}{\sum_{m=1}^M c_{jm} \dot{N}(\langle o \rangle_t, \mu_{jm}, U_{jm})} \right] \quad (50)$$

Then the reestimation formulas for the coefficients of the mixture density are of the form

$$c_{jk}^* = \frac{\sum_{t=1}^T \gamma_t(j, k)}{\sum_{t=1}^T \sum_{k=1}^M \gamma_t(j, k)} \quad (51)$$

$$\mu_{jk}^* = \frac{\sum_{t=1}^T \gamma_t(j, k) \langle o \rangle_t}{\sum_{t=1}^T \gamma_t(j, k)} \quad (52)$$

14 in the case of a simple mixture, γ_t generalizes to (35)

$$U_{jk}^* = \frac{\sum_{t=1}^T y_t(j, k) (\langle o \rangle_t - \mu_{jk}) (\langle o \rangle_t - \mu_{jk})'}{\sum_{t=1}^T y_t(j, k)} \quad (53)$$

It has been shown in [8] that an HMM state with a mixture density is equivalent to a multistate single-mixture density model. The basic idea is to create substates, where the transition coefficients from each state to its substates are equal to the mixture coefficients. Then the distribution of the composite set of substates (each with a single density) is mathematically equivalent to the composite mixture density within a single state. This is illustrated by illustration 4 below:

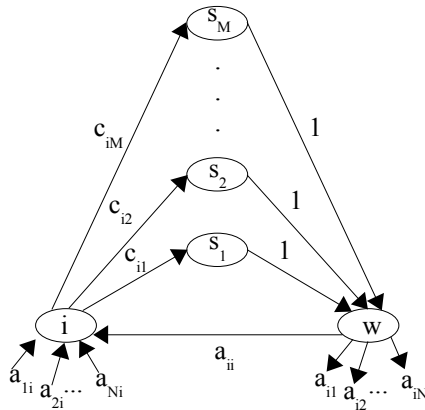


Illustration 4 Equivalence of a state with a mixture density to a multistate single-density distribution. *i* is the current state, *w* a wait state and *s* the substates

1.6 Types and Classes of HMMs

1.6.1 Types

1.6.1.1 Ergodic HMM

Ergodic HMM, or *fully connected HMM*, is the type of HMM we have considered until now: Every state of the model could be reached in a single step from every other state of the model. For example, a three state ergodic HMM would look like illustration 5 and is defined as in (54):

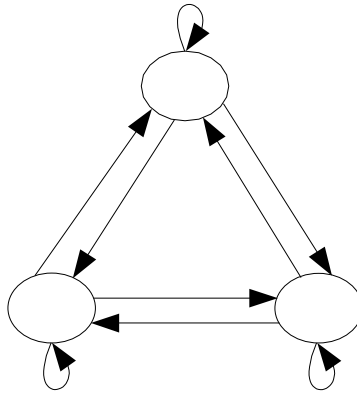


Illustration 5 three state ergodic HMM

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, \quad a_{ij} > 0, 1 \leq i, j \leq 3 \quad (54)$$

1.6.1.2 Left-Right HMM

The fundamental property of a *left-right HMM*, or *Bakis model*, is that as time increases, the state index increases (or stays the same), thus the system proceeds from left to right. It can easily model signals whose properties change over time in a successive manner, i.e. speech.

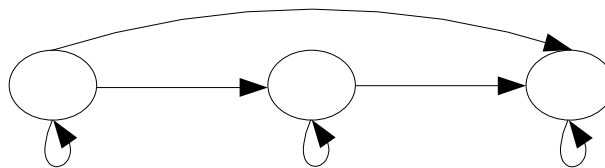


Illustration 6 three state left-right HMM

The entries of the transition matrix are

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad (55)$$

$$\begin{aligned} a_{ij} &> 0, & 1 \leq i \leq j \leq 3 \\ a_{ij} &= 0, & 1 \leq j < i \leq 3 \end{aligned}$$

giving us

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{bmatrix}, \quad a_{ij} > 0, 1 \leq i \leq j \leq 3 \quad (56)$$

1.6.1.3 Variations

Though there are other models, the ergodic and the left-right HMM are the most important ones. Besides, many HMMs are just variations from the previous two, where they vary in the sense that there are

- cross-links between paths
- parallel paths (i.e. the *parallel path left-right HMM*)
- limitations as to how many states can be skipped

1.6.2 Classes

1.6.2.1 Factorial HMMs

We generalize the HMM by representing the state S_t using a collection of discrete state variables

$$S_t = S_t^{(1)}, S_t^{(2)}, \dots, S_t^{(m)}, \dots, S_t^{(M)} \quad (57)$$

If each can take on K values¹⁵, the state space of the factorial HMM consists of all K^M combinations of the $S_t^{(m)}$ variables, and the transition structure results in a $K^M \times K^M$ transition matrix. Since both the time complexity and sample complexity of estimation algorithm are exponential in M , and interesting structures cannot be discovered due to the arbitrary interaction of all variables, we want to constrain the underlying state transitions. For example, we can let each state variable evolve to its own dynamics and uncouple it from the other state variables:

$$P(S_t | S_{t-1}) = \prod_{m=1}^M P(S_t^{(m)} | S_{t-1}^{(m)}) \quad (58)$$

¹⁵ naturally this can be extended, so that each state variable $S_t^{(m)}$ can take on $K^{(m)}$ variables

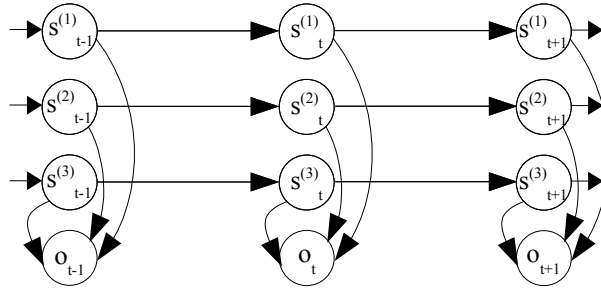


Illustration 7 three state factorial HMM

This is further described in [2].

1.6.2.2 Tree structured HMMs

In factorial HMMs, the state variables at one time step are assumed to be **a priori** independent given the state variables at the previous time step. If we couple the state variables in a single time step [9], we can relax this assumption.

For example, we can order them, such that $S_t^{(m)}$ depends on $S_t^{(n)}$ for $1 \leq n < m$. Furthermore, we can make the state variables and the output dependent on an observable input variable X_t :

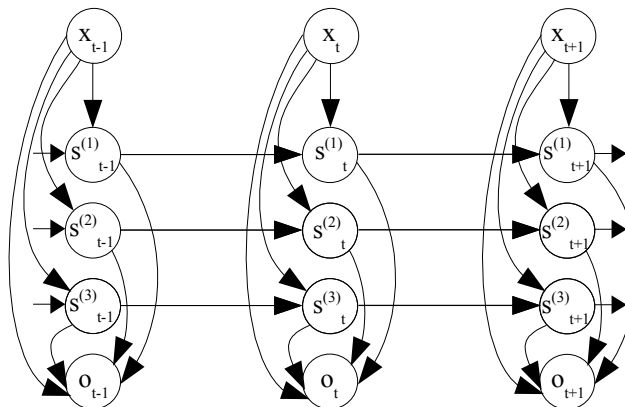


Illustration 8 three state tree structured HMM

Tree structured HMMs are useful for modeling time series with both temporal and spatial structure at multiple resolutions.

1.6.2.3 Autoregressive HMMs

The class of *autoregressive HMMs*¹⁶ is particularly applicable to speech processing. The observation vectors are drawn from an *autoregression process*. The components of the observation vector $O_{0:K-1}$ are assumed to be from an autoregressive Gaussian source, satisfying the

¹⁶ an autoregressive model is also known as *infinite impulse response filter (IIR)*, *all pole filter* and, in physics applications, as a *maximum entropy model*

relationship

$$o_k = - \sum_{i=1}^p a_i o_{k-i} + e_k \quad (59)$$

where $e_{0 \leq k < K}$ are Gaussian, independent, identically distributed random variables with zero mean and variance σ_e^2 , and $a_{1 \leq i \leq p}$ are the *autoregression* or *predictor coefficients*.

Verbally, since there is “memory” or “feedback”, the system can generate internal dynamics, and the current observation can be estimated by a linear weighted sum of previous observations, where the weights are the autoregression coefficients.

The problem in *autoregression (AR)* is to derive the best values a_i from a given series $O_{0:K-1}$. Most methods assume the series is linear and stationary. The main two categories are *least squares* and *Burg method* [10]. The most common least squares method is based upon the *Yule-Walker equations*. More specific information can be found in [1].

1.6.2.4 Vector Linear Predictor HMMs

When using HMM, it is assumed that successive frames from a given output state are *independent and identically distributed (IID)*. This piece-wise constant assumption about the temporal evolution of speech signals is known not to be a valid assumption [11], in fact, the speech signals are highly temporally correlated.

A possible solution to this is to extend the standard HMM to allow explicit modeling of temporal information by using *Vector Linear Predictors (VLP)*. VLP extended models can capture temporal dynamics whilst maintaining the efficient training and decoding algorithms of the standard HMMs. VLP-HMMs deliver better results than standard HMMs, but poorer than standard HMMs with *Dynamic Parameters*. Furthermore, VLP-HMMs do not give a better discrimination since the model is not the true speech model, and the selection of the predictor model offsets are arbitrary.

VLP-HMMs are in all aspects the same as standard HMMs except for the definition of the *output probability density*. For VLP-HMMs, the output probability density for state j of a particular Gaussian model is defined as

$$b_j(\langle o \rangle_t) = \frac{1}{\sqrt{(2\pi)^d |C_j|}} e^{-\frac{1}{2}(\langle o \rangle_t C_j^{-1} \langle o \rangle_t)} \quad (60)$$

where C_j is the covariance matrix of the prediction error $e_j(t)$ for state j at time t , which is defined as

$$e_j(t) = \langle o \rangle_t - \left(\mu_j + \sum_p A_{j_p} (\langle o \rangle_{t+q_p} - \mu_{j_p}) \right) \quad (61)$$

where A_{j_p} is the p th predictor for state j , q_p is the “offset” associated with the p th predictor and μ_{j_p} is the mean value for vectors at offset q_p . The predictors can have arbitrary offsets, and either a

full or a diagonal matrix can be used as predictor.

The reestimation formulas for maximum likelihood and *maximum mutual information (MMI)* can be derived using (60) and (61), though it should be noted that the MMI formulas require a constant D which represents the step size and ensures convergence. This constant is also required for the calculation of the inter-frame covariance matrix between observations from offset q_x and q_y (the state index j is dropped for clarity):

$$C_{xy}^* = \frac{\sum_t \psi(t) (\langle o \rangle_{t-q_x} - \mu_x^* (O_{t-q_y} - \mu_y^*)) + DC_{xy}}{\sum_t \psi(t) + D} \quad (62)$$

In the case of ML, $D=0$ and $\psi_j(t)$ is $\gamma_t(j)$ (cf. (35)). In the case of MMI D is chosen to ensure convergence and $\psi_j(t) = \gamma_t(j) - \gamma_t(j)_{gen}$, where $\gamma_t(j)_{gen}$ is the occupation probability for frame t of sentence r from the confusion lattice.

More can be found in [11]

1.7 Optimization Criteria

Though to this point only the maximum likelihood criterion has been used, there are other criteria such as the *maximum mutual information (MMI)* and *minimum discrimination information (MDI)*. Brief formal definition of the latter two along with ML are given below:

1.7.1 Maximum Likelihood (ML)

The standard ML criterion is to use a training sequence of observations O to derive the set of model parameters λ , yielding

$$\lambda_{ML} = \arg \max_{\lambda} P(O|\lambda) \quad (63)$$

All previously discussed reestimation algorithms, as for example the forward-backward algorithm in 1.4.2.3, provide a solution to this problem.

1.7.2 Minimum Discrimination Information (MDI)

The basic idea about statistical models is that if the model parameters are correctly chosen, the signal or the observation sequence can be modeled. As already mentioned 1.6.2.4, the assumed model is sometimes inadequate to model the observed signal; no matter how carefully the model parameters are chosen, the modeling accuracy is limited. One can try to overcome this *model mismatch situation* with MDI¹⁷.

We associate the observed signal $O_{1:T}$ with a sequence of constraints $R_{1:T}$ (e.g. r_t be the autocorrelation matrix that characterizes the observation o_t). Obviously, depending on R , O is only one of possibly uncountably many observation sequences that satisfy R . Hence there exists a set of probability distributions of the observation sequences that would also satisfy R , which we denote $\Omega(R)$. The MDI is a measure of closeness between two probability measures under the

¹⁷ also with MMI, cf. 1.7.3

given constraint R and is defined by

$$v(R, P_\lambda) \equiv \inf_{Q \in \Omega(R)} \{I(Q: P_\lambda)\} \quad (64)$$

$$I(Q: P_\lambda) = \int q(O) \log \frac{q(O)}{p(O|\lambda)} dO \quad (65)$$

where P_λ is the probability measure in HMM form, and $I(Q: P_\lambda)$ is the *discrimination information* – calculated based on the given training set of observations – between distributions Q and P_λ [1], $q(*)$ and $p(*|\lambda)$ are the pdf corresponding to Q and P_λ respectively.

The MDI criterion tries to choose a model parameter set λ such that $v(R, P_\lambda)$ is minimized. The MDI can be interpreted that the model parameter λ is chosen so that the model $P(O|\lambda)$ is as close as possible to a member of the set $\Omega(R)$. This is the motivation behind the MDI criterion: Since the closeness is always measured in terms of the discrimination information evaluated on the given observation, the intrinsic characteristics of the training sequences strongly influence the parameter selection, and by emphasizing the measure discrimination, the model estimation is no longer limited by the assumed model form. However, the MDI optimization is not as straightforward as the ML optimization problem, and no simple robust implementation of the procedure is known [1].

1.7.3 Maximum Mutual Information (MMI)

As the MDI criterion, the *maximum mutual information criterion (MMI)* tries to maximize the discrimination of each model¹⁸.

We define $P(v)$ to be the a priori probability for word v , whereas v is taken from a vocabulary of V words. Each v is represented by an HMM, with parameter set λ^v , $v=1, 2, \dots, V$. Together with the a priori probabilities the set of HMMs $A=\{\lambda^v\}$ thus defines a probability measure for an arbitrary observation sequence O

$$P_A(O) = \sum_{v=1}^V P(O|\lambda^v, v) P(v) \quad (66)$$

where $P(O|\lambda^v, v)$ indicates that it is a probability conditioned on the word v . Now, to train these models utterances of known (labeled) words are used. These labeled training sequences are denoted by O^v , with superscript v denoting that O^v is a rendition of word v . Then the mutual information is defined as

$$I_A(O^v, v) = \log P(O^v|\lambda^v) - \log \sum_{w=1}^V P(O^v|\lambda^w, w) P(w) \quad (67)$$

¹⁸ the ability to distinguish between observation sequences generated by the model that describes the correct word and

The MMI criterion to find the entire model set Λ such that the mutual information is maximized:

$$(\Lambda)_{MMI} = \max_{\Lambda} \left\{ \sum_{v=1}^V I_{\Lambda}(O^v, v) \right\} \quad (68)$$

Though both ML and MMI are *cross-entropy* approaches, they differ in the following way:

- ML:
an HMM for the distribution of the **data given the word** is matched to the empirical distribution
- MMI:
an HMM for the distribution of the **word given the data** is matched to the empirical distribution

The problem of MMI is though, that $(\Lambda)_{MMI}$ is not as straightforward to obtain as $(\Lambda)_{ML}$. Furthermore one often has to use general optimization procedures like the *descent algorithm* to solve (68) which lead to numerical problems in implementations.

2 Graphical Models (GM)

2.1 Introduction

Graphical models (GM) combine *probability theory* and *graph theory*. They provide a tool for dealing with the problems of **uncertainty** and **complexity**. They play an increasingly important role in the design and analysis of machine learning algorithms. The basic idea of graphical models is the notion of **modularity**: a complex system is built by combining simpler parts. Where these various parts are combined, probability theory ensures that the system as a whole is consistent and provides ways to **interface** models to data. Due to its graph theoretic side, humans can easily and intuitively model highly-interacting sets of variables. Furthermore, the resulting data structure not only lends itself naturally to the design of efficient general-purpose algorithms, but also can be used with all the well-developed and highly efficient graph algorithms suited for the specific network topology.

Probabilistic graphical models are *graphs* in which nodes represent random variables. *Arcs*, or the lack of arcs, represent *conditional independence* assumptions. Therefore they provide a compact representation of *joint probability distributions* [12]. There are two types of graphical models: *Undirected graphical models* and *directed graphical models*.

2.2 Undirected Graphical Models

2.2.1 Introduction

As “undirected” already implies, in undirected graphical models, if there is an arc between a node A and a node B , one can always go from A to B and from node B (back) to node A with **equal probability**, that is, there is no “one-way street”.

Undirected graphical models are an important tool for representing probability distributions. They are most popular with the physics and vision communities, and are of little importance for speech recognition. Their set of semantics differs from those of directed graphical models and can be looked up in [13] and [14]. To illustrate the differences between undirected and directed graphical models, the definition of independence is given:

2.2.2 Independence

Undirected graphical models¹⁹ have a simple definition of *independence* [12]:

Two sets of nodes A and B are conditionally independent given a third set, C , if all paths between the nodes in A and B are separated by a node in C .

2.3 Directed Graphical Models

2.3.1 Introduction

As opposed to undirected graphical models, the arcs of directed graphical models are “one-way streets”. If there is an arc from a node A to a node B , one can only go from B to A when there is an explicit extra arc from B to A ²⁰.

Directed graphical models are especially useful for AI and statistics applications and, hence, for speech recognition. They can encode deterministic relationships and are easier to “learn” than undirected graphical models, that is fit to data.

¹⁹ also called *Markov Random Fields (MRFs)* or *Markov networks* [12]

²⁰ Of course, for brevity one may define undirected arcs in a directed graphical model to represent two arcs with opposing directions but equal probability.

2.3.2 Definitions

Before we continue, we need to state some basic definitions from graph theory. Consider illustration 9 below:

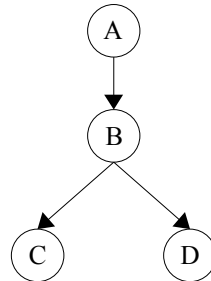


Illustration 9 a simple graph

The node X is a *parent* of another node Y if there is a directed arc from X to Y . If so, Y is a *child* of X . The *descendants* of a node are its children, its children's children, etc. For example, in illustration 9 B , C and D are A 's descendants. By contrast, the *ancestors* of a node are its parents, its parents' parents etc.

A *directed path* from a node X to node Y is a sequence of nodes starting from X and ending in Y such that each node in the sequence is a parent of the following node in the sequence, e.g. (A, B, D) is a directed path whilst (C, B, D) is not.

An *undirected path* from X to Y is a sequence of nodes starting from X and ending in Y such that each node in the sequence is a parent **or** a child of the following node. Hence a directed path is always also an undirected path, but not vice-versa.

2.3.3 Independence

Directed graphical models²¹ have a more complicated notion of independence which takes the directionality of the arcs into account [2]:

Each node is *conditionally independent* from its non-descendants given its parents. More generally, two disjoint sets of nodes A and B are conditionally independent given C , if C *d-separates* them.

d-separation [2][15] means that if along every undirected path between a node in A and a node in B there is a node D such that one of the following applies:

1. D has *converging arrows*²², and neither D nor its descendants are in C
2. D does not have converging arrows and D is in C

Put differently [2], A is conditionally independent from B given C if

$$P(A, B|C) = P(A|C)P(B|C), \quad \forall A, B \text{ where } P(C) \neq 0 \quad (69)$$

²¹ Also called Bayesian *networks* or *belief networks* (BNs) (cf page 26), it is important to note that despite the name, Bayesian methods don't need to be used at all. Rather they are so called because they use *Bayes' rule* for inference

²² that is, D is a child of both the previous and the following nodes in the path [12]

Though this definition is more complicated, directed graphical models have several advantages. Their most important one is that one can regard an *arc* from A to B as indicating that A **causes** B , which aids in constructing the graph structure.

2.3.4 Conditional Probability Distribution (CPD)

In addition to the graph structure we need to specify the parameters of the model. For a directed model, we need to specify the *conditional probability distribution (CPD)* at **each node**. If the variables are **discrete**, the CPD can be represented as a table, the *Conditional Probability Table (CPT)*, which lists the probability that each child node takes on each of its different values for each combination of values of its parents [12].

3 Bayesian Networks (BN)

3.1 Introduction

A *Bayesian Network (BN)* [15] has the ability to represent a probability distribution over arbitrary sets of random variables [16]. It is possible to factor these distributions in arbitrary ways, and to make arbitrary conditional independence assumptions. This makes it possible to vastly reduce the number of parameters required to represent a probability distribution and, therefore, allows the model parameters to be estimated with greater accuracy from a limited amount of data [17][18].

3.2 Why “Bayesian”?

Bayesian networks are named “Bayesian” because they make use of *Bayes’ Rule*, also called *Bayes’ Theorem*, for inference [12], which is

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (70)$$

where $P(B|A)$ is the *a posteriori* probability of B given A , and $P(A)$ is the *a priori* probability of A .

3.3 Conditional Independence and Representation

A BN is a directed graphical model, as introduced in 2.3. It is a graphical way to represent a particular factorization of a joint distribution. A directed arc is drawn from a node A to a node B if B is **conditioned on** A in the factorization of the joint distribution. For example, consider the BN

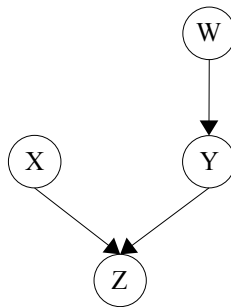


Illustration 10 a directed acyclic graph (DAG) consistent with the conditional independence relations in $P(W, X, Y, Z)$

with the independence relations

$$P(W, X, Y, Z) = P(W)P(X)P(Y|W)P(Z|X, Y) \quad (71)$$

It is easy to see, that e.g. Y is conditioned on W . Furthermore, given the values of X and Y we can use the conditional independence definition from 2.3.3 and show that W and Z are independent:

$$\begin{aligned}
 & P(W, Z|X, Y) \\
 &= \frac{P(W, X, Y, Z)}{P(X, Y)} \\
 &= \frac{P(W) P(X) P(Y|W) P(Z|X, Y)}{\int P(W) P(X) P(Y|W) P(Z|X, Y) dW dZ} \\
 &= \frac{P(W) P(Y|W) P(Z|X, Y)}{P(Y)} \\
 &= P(W|Y) P(Z|X, Y)
 \end{aligned} \tag{72}$$

The absence of arcs in a BN implies conditional independence relations which can be exploited to obtain efficient algorithms for computing marginal and conditional probabilities [2]. For example, the simplest conditional independence relationship encoded in a Bayesian network can be stated as follows:

A node is independent of its ancestors given its parents, where the ancestor/parent relationship is with respect to some fixed topological ordering of the nodes [12].

3.4 Example

Before we continue, a brief example shall be given:

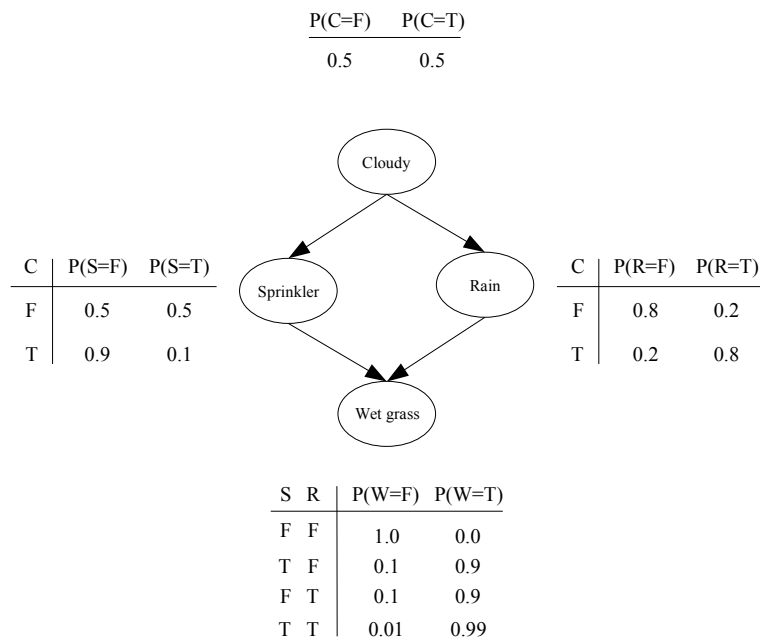


Illustration 11 a directed graph modeling the weather

All nodes are binary, i.e. have two possible values: *True* (T) and *false* (F). We see that the event “grass is wet” ($W=T$) has two possible causes: Either it is raining ($R=T$) or the water sprinkler is on ($S=T$). It is shown in the corresponding tables how strongly this relationship is, e.g. we see that $P(W=T|S=T, R=F)=0.9$ and therefore $P(W=F|S=T, R=F)=1-0.9=0.1$. Since the C node has

no parents, its CPT specifies the prior probability that it is cloudy (0.5 in this case).

By the *joint rule of probability*, the joint probability of all the nodes in the graph above²³ is

$$P(C, S, R, W) = P(C) P(S|C) P(R|C, S) P(W|C, S, R) \quad (73)$$

By using conditional independence relationships, we can rewrite this as

$$P(C, S, R, W) = P(C) P(S|C) P(R|C) P(W|S, R) \quad (74)$$

We can simplify $P(R|C, S)$ because R is independent of S given its parent C , and we can simplify $P(W|C, S, R)$ because W is independent of C given its parents S and R .

3.5 Inference

The most common task we wish to solve using BN is probabilistic inference [12]. Consider the example given in 3.4 above. Suppose we observe that the grass is wet. There are two possible causes for this: Either it is raining ($R=T$) or the sprinkler is on ($S=T$). We can use Bayes' Rule to compute the posterior probability of each explanation:

$$P(W=T) = \sum_{c, r, s} P(C=c, S=s, R=r, W=T) = 0.6471, \quad c, s, r \in \{T, F\} \quad (75)$$

$$\begin{aligned} & \frac{P(R=T|W=T)}{P(W=T)} \\ &= \frac{P(R=T, W=T)}{P(W=T)} \\ &= \frac{\sum_{c, s} P(C=c, S=s, R=T, W=T)}{P(W=T)} \\ &= \frac{0.4581}{0.6471} \\ &= 0.7079 \end{aligned} \quad (76)$$

$$c, s \in \{T, F\}$$

²³ compare to the example in 3.3

$$\begin{aligned}
& P(S=T|W=T) \\
&= \frac{P(S=T, W=T)}{P(W=T)} \\
&= \frac{\sum_{c,r} P(C=c, S=T, R=r, W=T)}{P(W=T)} \\
&= \frac{0.2781}{0.6471} \\
&= 0.4298
\end{aligned} \tag{77}$$

$c, r \in \{T, F\}$

So we see that it is more likely that the grass is wet, because it is raining: The *likelihood ratio* is $0.7079/0.4298 = 1.647$.

3.6 Explaining away

In the above example, two causes “compete” to “explain” the observed data. Hence even though S and R are marginally independent, they become conditionally independent when their common child W is observed. For example, when the grass is wet and we know that it is raining, the (posterior) probability of the sprinkler being on goes down: $P(S=T|W=T, R=T) = 0.1945$.

This is called *explaining away*, also known as *Berkenson's paradox* or *selection bias* [12].

3.7 Reasoning

There are two ways to infer in a Bayesian network [12]: *bottom-up* and *top-down*:

Bottom-up, or *diagnostic*, is to infer the most likely cause from an evidence of an effect. In the above example, we observed “grass is wet” and inferred that the most likely reason is that it is raining.

Top-down, or *generative*, is *causal reasoning*. For example, again using the above example, we can compute the probability that the grass will be wet given that it is cloudy.

3.8 Causality or correlation?

Bayesian networks enable us to put discussions on causality on a solid mathematical basis (cf 2.3.3 and 3.3). However, there may be the problem of distinguishing **causality** from mere **correlation**. We can “sometimes” [12] solve this, but we need to measure the relationship between **at least three variables**, where one acts as a “virtual control” for the relationship between the other two, so we don't always need to do experiments to infer causality.

3.9 Singly connected and multiply connected networks

If there is no undirected path with loops in the underlying graph of a BN, e.g. as in illustration 10, the network is called a *singly connected* network, and a general algorithm called *belief propagation* [15][19]²⁴ exists.

For *multiply connected networks*, in which there can be more than one undirected path between any two nodes, there exists a more general algorithm known as *junction tree algorithm* [20][21].

²⁴ cf 3.10

3.10 Evidence and Belief Propagation

The observed value of some variables in the network is called *evidence*. The goal of belief propagation is to update the marginal probabilities of all the variables in the network to incorporate this new evidence. This is achieved by *local message passing*:

Each node n sends a message to its parents and to its children. Since the graph is singly connected (cf 3.9), n separates the graph, and therefore the evidence, into two mutually exclusive sets:

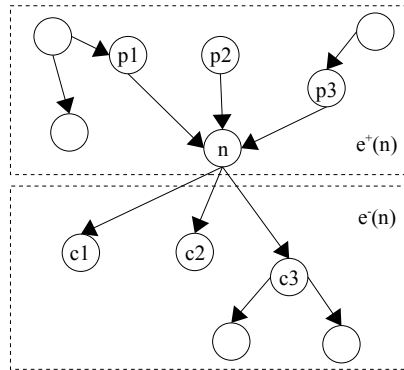


Illustration 12 separation of evidence in singly connected graphs

- $e^+(n)$: consists of n itself, the parents of n and the nodes connected to n through n 's parents²⁵
- $e^-(n)$: consists of the children of n and the nodes connected to n through n 's children²⁶

The *message* from n to each of its children is the probability of each setting of n given the evidence observed in the set $e^+(n)$. If n is a K -valued *discrete* variable, then this message is a K -dimensional vector. For **real-valued** variables, the message is the probability density over the domain of values n can take.

The message from n to each of its parents is the probability, given every setting of the parent, of the evidence observed in the set $e^-(n)$. The marginal probability of a node is proportional to the product of the messages obtained from its parents, weighted by the conditional probability of the node given its parents, and the message obtained from its children. If the parents of n are $\{p_1, \dots, p_k\}$ and the children of n are $\{c_1, \dots, c_l\}$, then

$$P(n|e) \propto \left[\sum_{p_1, \dots, p_k} P(n|p_1, \dots, p_k) \prod_{i=1}^k P(p_i|e^+(p_i)) \right] \prod_{j=1}^l P(c_j, e^-(c_j)|n) \quad (78)$$

where the summation (or, more generally, the integral) extends over all settings of $\{p_1, \dots, p_k\}$. For example, for the BN in illustration 10, given the evidence $e = \{X=x, Z=z\}$,

²⁵ the nodes for which the undirected path to n goes through a parent of n

²⁶ the nodes for which the undirected path to n goes through a child of n

$$\begin{aligned}
& P(Y|X=x, Z=z) \\
& \propto \left[\int P(Y|W)P(W)dW \right] P(Z=z, X=x|Y) \\
& \propto P(Y)P(Z=z|X=x, Y)P(X=x)
\end{aligned} \tag{79}$$

where $P(W)$ is the message, passed from W to Y since $e^+(W)=\{\}$ ²⁷, and $P(Z=z, X=x|Y)$ is the message passed from Z to Y . Variables in the evidence set are referred to as *observable variables*, while those not in the evidence set are referred to as *hidden variables*.

3.11 BN with discrete and continuous nodes

Though the previous examples used nodes with categorical values, as has already been implied in 3.10 it is also possible to create Bayesian networks with *continuous valued nodes*. The most common distribution for such variables is the Gaussian. Its implementation is similar to the HMM one (cf 1.5).

3.12 Dynamic Bayesian Networks (DBN)

3.12.1 Introduction

*Dynamic Bayesian Networks*²⁸ (DBN) are directed graphical models of stochastic processes. Put differently, they are simply BN for modeling time series data. Since we assume in time series modeling that an event can cause another event in the future, but no event in the future can cause one in the past, directed arcs flow forward in time²⁹. Hence the design of the BN is greatly simplified. For example, consider the example above:



Illustration 13 a Bayesian network representing a first-order Markov process

Illustration 13 is an example of a first-order Markov process, where each of the variables O_i in the observed sequence $O_{1:T}$ is influenced only by the previous variable O_{i-1} :

$$P(O_{1:T}) = P(O_1)P(O_2|O_1)\dots P(O_T|O_{T-1}) \tag{80}$$

As already mentioned in 1.2.1, one way to extend Markov models is to allow higher interactions between variables, e.g. use a n th-order Markov process where $n > 1$. But we can go much further than that, in fact, we can implement HMMs and other models as a DBN, which can have significant advantages.

²⁷ empty set

²⁸ *temporal Bayesian network* would be a better suited and descriptive name as long as one assumes the usual case that the structure of the BN does not change (*time-invariant*), but DBN has become entrenched [12]

²⁹ of course, they may also point to the same node to model repetitions

3.12.2 Strengths and Advantages of DBN

Dynamic Bayesian networks are ideally suited for modeling temporal processes. They have the following advantages over other models such as *Kalman filters* [12], *Neural networks* [16] and HMMs [16]:

1. **nonlinearity:**

By using a tabular representation of conditional probabilities³⁰, it is quite easy to represent arbitrary nonlinear phenomena. Moreover, it is possible to do efficient computation with DBNs even when the variables are continuous and the conditional probabilities are represented by Gaussians [22].

2. **interpretability:**

Each variable represents a specific concept.

3. **factorization:**

The joint distribution is factorized as much as possible. This leads to:

1. **statistical efficiency:**

Compared to an unfactored HMM with an equal number of possible states, a DBN with a factored state representation and sparse connections between variables will require **exponentially fewer** parameters.

2. **computational efficiency:**

Depending on the exact graph topology, the reduction in model parameters may be reflected in a reduction in running time.

4. **extensibility:**

DBNs can handle large numbers of variables, provided the graph structure is sparse.

5. **semantics:**

DBNs have a precise and well-understood probabilistic semantics, i.e. each variable can model a specific concept.

3.12.3 Hidden Markov Models as DBN

Hidden Markov models are a sub-class of DBN in which we assume that observations are dependent on a *hidden variable (state)*. The following illustration shows the simplest kind of an HMM with one discrete hidden node and one discrete observed node per slice:

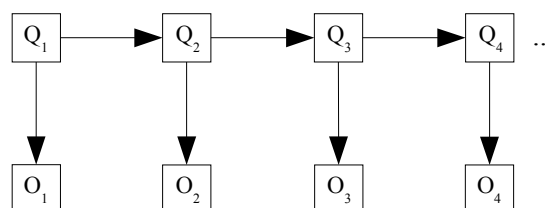


Illustration 14 HMM as DBN with one discrete hidden node and one discrete observed node, unrolled for 4 time slices

30 CPT, cf 2.3.4 and the example 3.4

Unrolled for four time slices, $O_{1:4}$ are the observed nodes and $Q_{1:4}$ are the hidden nodes of the above HMM. The structure and parameters are assumed to repeat as the model is unrolled further. Hence to specify a DBN we need to define

- the *intra-slice topology*: topology within a slice
- the *inter-slice topology*: topology between two slices
- the parameters for the first two slices³¹

Though it is straightforward to create a DBN from an HMM by use of a single hidden variable (which can take as many values as there are states in the HMM) in each time slice, there is no known way to convert an existing HMM to the **minimal**³² equivalent DBN [16].

However, there is a simple procedure for constructing an HMM from a DBN. What we want to specify are the five quantities³³ of an HMM:

1. the number of states
2. the number of observation symbols
3. transition probabilities between states
4. emission probabilities for observation symbols
5. probability distribution on the initial states

Given a DBN, these quantities can be derived for an equivalent HMM in the following way:

1. The number of HMM states is equal to the number of ways the DBN states can be instantiated, so if we have e.g. k **binary** DBN state nodes, there are 2^k HMM states.
2. The number of HMM observation symbols is equal to the number of ways the DBN observation nodes can be instantiated, so if we have e.g. k **binary** DBN observation nodes, there are 2^k HMM observation symbols.
3. To calculate the HMM transition probability from state i to state j :
 1. instantiate the DBN state nodes in a time slice t in the configuration that corresponds to HMM state i
 2. instantiate the state nodes in slice $t+1$ in the configuration that corresponds to j .
 3. the transition probability is the product of the probabilities of the state nodes in slice $t+1$ given their parents
4. Emission probabilities are calculated in a similar manner as the transition probabilities, but only the state and observation nodes in a single time slice are instantiated.
5. To calculate the probability of being initially in state i :
 1. instantiate the DBN state nodes in time slice 1 in the configuration that corresponds to HMM state i
 2. the product of the probabilities of the state nodes in time slice 1 given their parents³⁴ is the desired prior

3.12.4 Inference

For inference in singly connected networks the *belief propagation* algorithm has already been introduced in 3.10. However, besides the junction-tree algorithm mentioned in 3.9, for multiply

31 such a two-slice temporal BN is often called *2TBN*

32 minimal in terms of the number of parameters or computational requirements

33 as defined in (8) in 1.3

34 if there are no parents, they are not required

connected DBN *cutset conditioning* [21] may be the simplest algorithm [16].

3.12.5 Learning

In general, the learning techniques for Bayesian networks are analogous to the learning techniques for HMMs. In both cases, EM³⁵ is used. However, under some circumstances it is necessary to resort to *gradient descent*:

- If the conditional probabilities are represented by distributions in the *exponential family*, EM is used. Since the exponential family includes the Gaussian distribution EM is usually applicable.
- If the derivative of the data likelihood with respect to the conditional probabilities can be computed, gradient descent [23] is applicable. For example, if a functional representation of conditional probabilities is used in a BN, it may not be possible to derive EM update equations. Also, when optimization criteria other than maximum likelihood (ML)³⁶ are used in HMMs, such as maximum mutual information (MMI) or minimum discrimination information (MDI), EM is not applicable.

35 cf 1.4.2.3 and [5]

36 cf 1.7.1-1.7.3 and [1]

4 Comments

Due to lack of time and for the sake of brevity many important algorithms and methods, such as e.g. *deleted interpolation* and *corrective training* [1], *inference in clique trees* and *inference in general graphs* [16], or comparisons to other interesting models such as the *Kalman filter*, *neural networks* or *dynamic Bayesian multinets* [24] have not found their way into this essay. They can be looked up in the literature list, where especially recommended readings are [25], [2], [1], [12] and [16].

III INDEX

| | | | |
|--|-------------------|---|--------|
| 2TBN..... | 33 | Burg method..... | 19 |
| σ^2 | 3pp., 7p., 10, 13 | causality..... | 29 |
| A..... | 5 | causal reasoning..... | 29 |
| advantage of Baum's auxiliary function.... | 12 | child..... | 24 |
| advantages of DBN..... | 32 | cjk..... | 13 |
| advantages of directed graphical models |25 | clique trees..... | 35 |
| aij..... | 2 | conditional independence..... | 23 |
| all pole filter..... | 18 | conditionally independent..... | 24 |
| Alternative Viterbi Implementation..... | 9 | conditional probability distribution..... | 25 |
| ancestor..... | 24 | Conditional Probability Table..... | 25 |
| ancestors..... | 24 | continuous..... | 13, 31 |
| a posteriori..... | 26 | Continuous Observation Densities..... | 13 |
| AR..... | 19 | continuous valued nodes..... | 31 |
| arc..... | 25 | converging arrows..... | 24 |
| arcs..... | 23 | corrective training..... | 35 |
| ASR..... | 2 | correlation..... | 29 |
| automatic speech recognition systems .2 | | covariances..... | 2 |
| Autoregression..... | 19 | CPD..... | 25 |
| autoregression coefficients..... | 19 | CPT..... | 25 |
| autoregression process..... | 18 | cross-entropy..... | 22 |
| Autoregressive HMMs..... | 18 | cross-entropy approach..... | 22 |
| auxiliary function..... | 12 | cutset conditioning..... | 34 |
| B..... | 5 | DBN..... | 31 |
| backward procedure..... | 10 | Definition of Hidden Markov Models..... | 4 |
| backward variable..... | 10 | descent algorithm..... | 22 |
| Bakis model..... | 16 | diagnostic..... | 29 |
| Baum's Auxiliary Function..... | 12 | directed graphical models..... | 23 |
| Baum's auxiliary function's advantage .12 | | directed path..... | 24 |
| Baum-Welch method..... | 10 | discrete hidden states..... | 4 |
| Bayesian Networks..... | 26 | Discrete Time Markov Processes..... | 2 |
| Bayes' Rule..... | 24, 26 | discrimination information | 21 |
| Bayes' Theorem..... | 26 | d-separate..... | 24 |
| Belief Networks..... | 24 | d-separation..... | 24 |
| belief propagation..... | 29p. | Dynamic Bayesian Multinets..... | 35 |
| Berkenson's paradox..... | 29 | Dynamic Bayesian Networks..... | 31 |
| BN..... | 26 | Dynamic Parameters..... | 19 |
| BNs..... | 24 | elliptically symmetric density..... | 13 |
| bottom-up..... | 29 | EM..... | 10 |
| | | emission matrix..... | 5 |
| | | ergodic..... | 15 |

| | | | |
|---|--------|---|--------|
| Ergodic HMM..... | 15 | Processes..... | 4 |
| Evidence..... | 30 | local maxima..... | 12 |
| expectation-maximization method..... | 10 | local message passing..... | 30 |
| explaining away..... | 29 | log-concave density..... | 13 |
| exponential family..... | 34 | Markov networks..... | 23 |
| Factorial HMMs..... | 17 | Markov Processes..... | 2 |
| first order Markov chain..... | 2 | Markov property..... | 4 |
| Forward Algorithm..... | 7 | Markov Random Fields..... | 23 |
| Forward-Backward Algorithm..... | 11 | maximum entropy model..... | 18 |
| forward procedure..... | 8 | Maximum Likelihood..... | 12, 20 |
| fully connected HMM..... | 15 | Maximum Mutual Information..... | 20p. |
| Gaussian..... | 13 | MDI..... | 20 |
| generative..... | 29 | mean value..... | 13 |
| GM..... | 23 | message..... | 30 |
| gradient descent..... | 34 | Minimum Discrimination Information..... | 20 |
| gradient techniques..... | 10, 12 | mixture..... | 13 |
| grammars..... | 2 | mixture coefficients..... | 13 |
| Graphical Models..... | 23 | mixture gains..... | 13 |
| graphs..... | 23 | ML..... | 12, 20 |
| graph theory..... | 23 | MMI..... | 20p. |
| hidden..... | 4, 31 | model mismatch situation..... | 20 |
| Hidden Markov Model..... | 2 | MRFs..... | 23 |
| hidden variable..... | 32 | multiply connected networks..... | 29 |
| hidden variables..... | 31 | multivariate Gaussian distribution..... | 13 |
| HMM..... | 2 | Neural Networks..... | 32 |
| IID..... | 19 | normal distribution..... | 13 |
| IIR..... | 18 | observable..... | 31 |
| independence..... | 23 | observable Markov model..... | 3 |
| independent and identically distributed.... | 19 | observable variables..... | 31 |
| infinite impulse response filter..... | 18 | observation matrix..... | 5 |
| initial state probability distribution..... | 4 | observation sequence..... | 4 |
| inter-slice topology..... | 33 | observation vector..... | 13 |
| intra-slice topology..... | 33 | Optimal State Sequence..... | 6, 8 |
| joint probability distributions..... | 23 | output probability density..... | 19 |
| joint rule of probability..... | 28 | parallel path left-right HMM..... | 17 |
| junction tree algorithm..... | 29 | Parameter Estimation..... | 6 |
| Kalman Filter..... | 32 | parent..... | 24 |
| learning problem..... | 10 | partial observation sequence..... | 7 |
| least squares..... | 19 | partial probabilities..... | 7 |
| left-right HMM..... | 16 | partial probability..... | 8 |
| likelihood ratio..... | 29 | pdf..... | 13 |
| Limitations of Discrete Time Markov | | predictor coefficients..... | 19 |
| | | probability density function..... | 13 |

| | | | |
|-------------------------------------|-------|------------------------------------|-----|
| Probability Evaluation..... | 5p. | Tree structured HMMs..... | 18 |
| probability theory..... | 23 | two-slice temporal BN..... | 33 |
| random variable..... | 13 | undirected graphical models..... | 23 |
| reestimation formulas..... | 11 | undirected path | 24 |
| second-order statistics..... | 2 | variance..... | 13 |
| selection bias..... | 29 | Vector Linear Predictor HMMs..... | 19 |
| singly connected | 29 | Vector Linear Predictors..... | 19 |
| standard Gaussian distribution..... | 13 | vector quantization codebooks..... | 13 |
| standard normal distribution..... | 13 | virtual control..... | 29 |
| templates..... | 2 | Viterbi Algorithm..... | 8p. |
| temporal Bayesian network..... | 31 | VLP..... | 19 |
| time-invariant..... | 5, 31 | Yule-Walker equations..... | 19 |
| top-down..... | 29 | | |

IV BIBLIOGRAPHY

- 1: L.K. Saul, M. Rahim, "Modeling acoustic correlations by factor analysis", 1998
- 2: K. Murphy, "A Brief Introduction to Graphical Models and Bayesian Networks", 2001
- 3: A.P. Dempster, N.M. Laird, D.B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm", 1977
- 4: L. Rabiner, B.-H. Juang, "Fundamentals of Speech Recognition", 1993
- 5: Zoubin Ghahramani, "An Introduction to Hidden Markov Models and Bayesian Networks",
- 6: A.J. Viterbi, "Error bounds f. convolutional codes and asymptotically optim. decod. algor.", 1967
- 7: L.R. Rabiner, S.E. Levinson, M.M. Sondhi, "Application of the Theory of Probabilistic Functions to ASR", 1983
- 8: O. Kerner, T. Thode, J. Steffens, R. Voller, J. Maurer, "Vieweg Mathematik Lexikon", 1995
- 9: B. Juang, S.E. Levinson, M.M. Sondhi, "ML estimation for multivariate mixture observations of Markov chains", 1985
- 10: L.K. Saul, M.I. Jordan, "Mixed memory Markov models", 1997
- 11: Paul Bourke, "Autoregression Analysis (AR)", 1998
- 12: K.K. Chin, P.C. Woodland, "Maximum Mutual Information Training of HMMs with Vector Linear Predictors", 2002
- 13: J. Besag, "Spatial interaction and the statistical analysis of lattice systems", 1974
- 14: S. Geman, D. Geman, "Stochastic relaxation, Gibbs distributions, Bayesian restoration of images", 1984
- 15: J. Pearl, "Prob. Reasoning in Intelligent Systems: Networks of Plausible Inference", 1988
- 16: G.G. Zweig, "Speech Recognition with Dynamic Bayesian Networks", 1998
- 17: G.G. Zweig, "A forward-backward algorithm for inference in Bayesian networks", 1996
- 18: Z. Ghahramani, M.I. Jordan, "Factorial hidden Markov models", 1997
- 19: J.H. Kim, J. Peal, "A comp. model for causal and diagnostic reasoning in inference systems", 1983
- 20: F.V. Jensen, S.L. Lauritzen, K.G. Olesen, "Bayesian updating in recursive graphical models by local computations", 1990
- 21: S.L. Lauritzen, D.J. Spiegelhalter, "Local computations with probabilities on graphical structures and their ...", 1988
- 22: R.D. Schachter, M.A. Peot, "Simulation approaches to general probabilistic inference on belief networks", 1989
- 23: J. Binder, D. Koller, S. Russell, K. Kanazawa, "Adaptive probabilistic networks with hidden variables", 1997
- 24: J.A. Bilmes, "Dynamic Bayesian Multinets", 2000
- 25: A.P. Dempster, "Covariance Selection", 1972

V ILLUSTRATIONS

| | |
|--|----|
| 1. the relationships between the various models in this essay..... | 1 |
| 2. Markov model of the weather..... | 3 |
| 3. partial best paths in a simplified 3-state model for the first 3 observations | 8 |
| 4. Equivalence of a state with a mixture density to a multistate single-density distribution. i is the current state, w a wait state and s the substates..... | 15 |
| 5. three state ergodic HMM..... | 16 |
| 6. three state left-right HMM..... | 16 |
| 7. three state factorial HMM..... | 18 |
| 8. three state tree structured HMM..... | 18 |
| 9. a simple graph..... | 24 |
| 10.a directed acyclic graph (DAG) consistent with the conditional independence relations in $P(W, X, Y, Z)$ | 26 |
| 11.a directed graph modeling the weather..... | 27 |
| 12.separation of evidence in singly connected graphs..... | 30 |
| 13.a Bayesian network representing a first-order Markov process..... | 31 |
| 14.HMM as DBN with one discrete hidden node and one discrete observed node, unrolled for 4 time slices..... | 33 |