A Bayesian Approach to Learning Hidden Markov Model Topology with Applications to Biological Sequence Analysis

Inaugural–Dissertation
zur
Erlangung des Doktorgrades
der Mathematisch–Naturwissenschaftlichen Fakultät
der Universität zu Köln

vorgelegt von
Alexander Schliep
aus Düsseldorf

2002

Berichterstatter: Prof. Dr. Rainer Schrader
Prof. Dr. Ewald Speckenmeyer
Dr. David C. Torney

Tag der mündlichen Prüfung: 28. Juni 2001

Kurzzusammenfassung

In der statistischen Modellierung und der statistischen Mustererkennung werden Hidden-Markov-Modelle (HMM) in zahlreichen Anwendungsgebieten erfolgreich eingesetzt. Dabei besteht ein fundamentales Problem in der Auswahl der zugrundeliegenden Architektur bzw. Topologie, also der Anzahl der Zustände des HMMs und der zwischen ihnen erlaubten Übergänge. Dies gilt insbesondere, wenn das Wissen aus dem Anwendungsbereich unzureichend für eine gesicherte Topologie-Wahl ist, oder bei der sogenannten black-box Modellierung. Die Bedeutung der HMM-Topologie ergibt sich aus der Tatsache, daß eine zu große Anzahl an Zuständen zu unverläßlichen Parameterschätzungen führt und andererseits eine zu geringe Anzahl an Zuständen es nicht erlaubt, relevante statistische Eigenschaften der Datenquelle zu modellieren.

Wir haben einen Algorithmus entwickelt, der ausgehend von Sequenzdaten, die von einem ergodischen Prozess erzeugt wurden, ein HMM samt Topologie und Parametern lernt. Der dafür benutzte Bayes'sche Ansatz erlaubt eine Steuerung der erzielten Generalisierung mittels einer a-priori Verteilung über einen zentralen Parameter.

Abstract

Hidden-Markov-Models (HMMs) are a widely and successfully used tool in statistical modeling and statistical pattern recognition. One fundamental problem in the application of HMMs is finding the underlying architecture or topology, particularly when there is no strong evidence from the application domain — e.g., when doing black box modeling. Topology is important with regard to good parameter estimates and with regard to performance: A model with "too many" states — and hence too many parameters — requires too much training data while an model with "not enough" states impedes the HMM from capturing subtle statistical patterns.

We have developed a novel algorithm that, given sequence data originating from an ergodic process, infers an HMM, its topology and its parameters. We introduce a Bayesian approach, where a suitable prior forces generalization while giving the user control with a single prior on one parameter.

Acknowledgments

Foremost I would like to thank my adviser, Prof. Dr. Rainer Schrader, for his support, his encouragement and the excellent work environment during my years in his group at the Zentrum für Angewandte Informatik, Köln (ZAIK). Thanks to Prof. Dr. Achim Bachem, who hired me at the predecessor of the ZAIK, the Zentrum für Paralleles Rechnen (ZPR).

Thanks to my part-time boss, full-time mentor and friend, Dr. David C. Torney (Los Alamos National Laboratory) for getting me started in bioinformatics, for his encouragement and his enthusiasm, many helpful discussions, not only pertinent to bioinformatics and this thesis, and also for serving as a member of my thesis committee. I am very grateful to Dr. Catherine Macken (Los Alamos National Laboratory) who got me involved with Hidden Markov Models and particular with this problem. Also, I would like to thank the further members of my thesis committee, Prof. Dr. Ewald Speckenmeyer, Prof. Dr. Diethard Tautz and Dr. Stefan Porschen. As for bringing my prose up to the appropriate level of editorial quality, my deepest thanks go out to Lars Kaderali, Matthias Hayer, Bernhard Knab, Alexander Schönhuth, Karin Weinbrecht and Winfried Hochstättler.

With regards to my colleagues at the ZAIK (though it probably always will be the ZPR for me): It was a great pleasure and an utmost privilege to work in a group of so many enthusiastic, talented, driven individuals who shared their knowledge and their wit so freely. Thanks to everybody! I just would like to thank particularly Matthias Hayer, my long-time office mate, Dr. Winfried Hochstättler, as well as my collaborators in statistical modeling — Bernhard Knab, Bernd Wichern, Filippo Castiglione and Barthel Steckemetz — as well as the students I was allowed to collaborate with: Eva Bolten, Ramazan Buzdemir, Achim Gädke, Lars Kaderali, Torsten Pattberg, Peter Pipenbacher, Thordis Linda Thorarinsdottir, Olaf Wendisch

To everybody in T-10 at Los Alamos National Laboratory: Thank you for your hospitality, the many interesting discussions and the stimulating environment during my numerous visits.

Thanks to Dr. Sebastian Schneckener, Olav Zimmermann, Vera Grimm, Holger Klein and Oliver Hofmann in the biochemistry department at Cologne for talking some biological sense into a computer scientist over many lunches and much, much more.

Last, but not least, I have to thank the agencies and foundations, who supported my research: US Department of Energy, European Union, Deutscher Akademischer Austauschdienst, Program in Mathematics and Molecular Biology, MSWWF NRW, Sponsors and organizers of the ISMB and RECOMB conferences and the Käte-Hack-Stiftung, Köln.

On a more personal note: Thanks to my family and my friends for their love, understanding and support, particularly during the last two years.

Part of this work was performed under the auspices of the U.S. Department of Energy, Office of Biological and Environmental Research, under contract Nr. W-7405-ENG-36.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As computers capable of storing and processing large data-sets became abundant, methods from statistical pattern recognition and classification found their way into the applied sciences and industrial research. They are used, for example, for tasks such as speech recognition within a fixed problem frame. The same methods are an important tool in *explorative* data analysis, a help on the quest for squeezing the most information out of whatever data one has at hand.

Hidden Markov Models (HMMs) [65], as one particular class of statistical models used in pattern recognition and classification, have been applied with great success in both cases. They provide a sound statistical framework, and allow for efficient and numerically stable algorithms. They can be visualized as (mathematical) graphs, which constitutes an effective user interface in the process of creating models for a particular application. They have become a basic and well-understood tool in the applied sciences; cf. [19, 33, 54].

In explorative data analysis however, particularly when the true nature of the process generating the data is unknown or only partially understood, there is one specific problem limiting the use of HMMs. That is, how should one choose the right HMM *topology* — basically the number of states in the model and how those states should be interconnected? The two main aspects contributing to the bottleneck are insufficient knowledge about the problem domain to support the topology-choosing process, and, even if enough information is available, the insufficient throughput of the (mostly manual) process used so far.

Also, in data analysis, often an increased *sensitivity* or, equivalently, a higher degree of *generalization* is required. That is, instead of having one fixed model, one would prefer a choice among general models, or, more exactly, a method which produces models at varying levels of specificity.

Finally, smaller models allow for more reliable parameter estimates given the same amount of training data, since the number of parameters depends, in general, quadratically on the model size. As the model size also enters quadratically in the computational complexity of the fundamental HMM algorithms, there is even motivation from a purely practical point of view for a method which allows inference of models and their topologies, while giving the user a handle on the size of the model produced.

The algorithms known in the literature treat the problem of learning HMM topology in an ad-hoc manner [19], with one exception [73], which uses a Bayesian approach but requires very

detailed specification of priors on the space of HMMs.

The main contribution of this work is the development of a robust and efficient algorithm which learns an HMM — including its topology — from data. The algorithm is formulated in a Bayesian setting, where a suitable prior distribution influences the generalization power of the model inferred. The prior is embodied in one internal parameter of the algorithm. It is easy to interpret, which is helpful in making sensible choices thereof.

Additionally, we introduce a novel measure on the space of HMMs, indicating the level of distinguishability between states and thus allowing to quantify how close an HMM is to being a Markov chain. We argue the relevance of the measure with respect to learning HMMs and with respect to comparing HMMs.

This thesis is organized in six chapters.

Following this introduction, we describe the statistical pattern classification problem in chapter two with the goal to formalize the particular problem of HMM inference in an appropriate framework. Subsequently, we give a concise definition of Bayesian statistics, alluding to the technique of *explicit* formulation of a priori assumptions relevant for inferring statistical models from data. Concepts from information theory — such as cross-entropy — are defined to provide a measure for the importance of observations when going from prior to posterior. This is followed by notations and concepts from the theory of stochastic processes, Markov chains, graphs and strings, which we require in the further development.

Chapter three gives an overview over the theory of HMMs with discrete observations. Following the definition of an HMM with discrete observations over a discrete-state, time-homogeneous, first-order Markov chain, efficient algorithms for the three fundamental problems [65] are developed. That is, we present algorithms for computing the likelihood of an observation sequence given a model, for finding the state sequence which is the "optimal" explanation for an observation sequence, and for adjusting the model parameters as to increase the likelihood of the training data. This is followed by a thorough discussion of HMM topology — a component of HMMs usually chosen by experts during the modeling process — and an overview of the literature of learning HMM topology. Also, the known distance measures on the space of HMMs are reviewed. Two alternative representations of HMMs conclude the chapter.

In chapter four the inference algorithm is developed for the case of ergodic stochastic sources. A motivation is given for using the so-called k-tails as identifiers of HMM states in the underlying data structure of a prefix tree, which provides a compact representation of a realization of the ergodic source. The influence of a Bayesian prior on the generalization capabilities is investigated. Subsequently, the peculiarities of the algorithm and the particular choices for its individual components are discussed, followed by an investigation of the computational complexity.

Chapter five is dedicated to the evaluation of the inference algorithm. We perform an evaluation on fabricated data. That is, training data produced by a known HMM is used as input to the algorithm, and the distance measures described in chapter three are used to quantify the differences between the source and the inferred HMM. The problem of inferring a known model is either trivial or impossible, depending on whether the HMM is in fact a Markov chain or whether it is completely hidden. To allow a distinction, we define a novel measure of distinguishability and investigate it analytically. We extend it to the case of transient HMMs and demonstrate its

heuristic suitability in that case. The evaluation is performed on two families of HMMs to obtain an overall performance evaluation on a wide range of source processes and on some particular HMM, chosen for specific topological features, which we are able to recover. These topological features, basically *deterministic* signals in a *stochastic* sequence, are prevalent in various disguises in many biological sequences. Examples are TATA boxes in eukaryotic promoter sequences, start and stop codons in DNA.

This is followed in Chapter 6, by a conclusion summarizing the results obtained in this work on improving inference of HMM in a black-box setting. Also, a mention is made of possible extensions to the method and further promising application areas.

Chapter 2

Basic Concepts and Definitions

In this chapter, we will first introduce statistical pattern classification, in order to establish a framework for problem formulation and for putting the particular problem of HMM learning investigated in this thesis into perspective. Subsequently, necessary definitions from statistics, information, graph theory, and "stringology" will be given.

2.1 Statistical Pattern Classification

All applications of Hidden Markov models can be viewed more abstractly as statistical pattern classification problems [23]. In its simplest form, see Fig. 2.1, there is

- an *abstract class* corresponding to an unknown process — the *stochastic source* — generating data,
- a *set of data*, usually finite, collected and annotated with respect to membership in the abstract class,
- a *stochastic model* representing the unknown process, which is inferred from the data, and

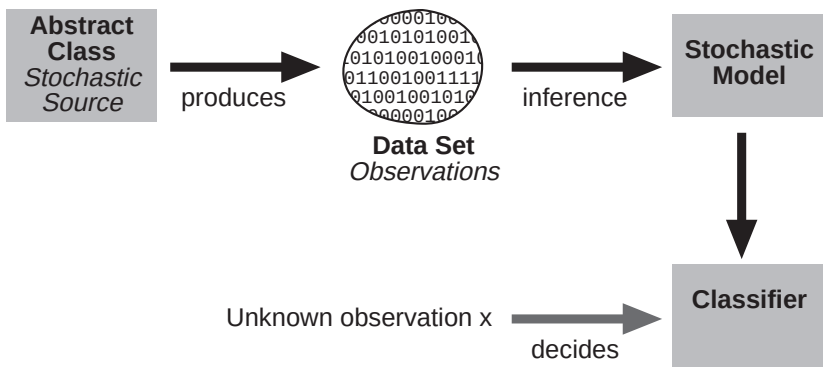

Figure 2.1: *The abstract class, represented by an unknown stochastic source, produces a set of observations. A stochastic model is inferred, with the goal of predicting membership in the abstract class for additional data.*

- a *decision function* based on the stochastic model, indicating whether a data point belongs to the abstract class or not.

In the following, we will formalize the foregoing concepts.

Definition 2.1 (Stochastic source) *The* stochastic source *[31] is the process generating the data belonging to the abstract class. Typically, neither process type nor process parameters of the stochastic source are known. The source is sometimes also called the* true *model.*

Definition 2.2 (Training data) *With* training data*, we refer to a collection of data, say*

$$\mathcal{S} := \{x_1, x_2, \ldots, x_n\}, \tag{2.1}$$

and an annotation*, indicating whether the data originates from the given stochastic source or not. That is, for* $n \in \mathbb{N}$

$$\mathcal{S}_a := \{(x_1, i_1), (x_2, i_2), \ldots, (x_n, i_n)\}, \tag{2.2}$$

where i_k*,* $1 \leq k \leq n$ *is one in the former case and zero otherwise. All* x_k *are elements of some fixed domain.*

Definition 2.3 (Training) *Choose a collection of, usually parameterized, statistical models, say* $\mathcal{M}(\theta)$*, from which a model for the abstract class will be inferred. The choice of model collection — e.g., a particular class of HMMs — is typically done manually. The process of adjusting the parameters* θ *is called* training*. The two most frequently used objective functions for training are the following:*
Maximum likelihood (ML): *Maximize the* data *likelihood, i.e. the joint probability of observing the data, given the model,*

$$P(\mathcal{S}|\mathcal{M}(\theta)). \tag{2.3}$$

Maximum a posteriori probability (MAP): *In this setting, we are given a prior distribution over model parameters, and the objective is to maximize the likelihood of the* model *(Cf. Eq. (2.9). Note, we can take* $P(\mathcal{S})$ *as constant):*

$$P(\mathcal{M}(\theta)|\mathcal{S}) \propto P(\mathcal{M}(\theta))P(\mathcal{S}|\mathcal{M}(\theta)). \tag{2.4}$$

Definition 2.4 (Classifier) *A trained model allows computation of* $P(x|\mathcal{M}(\theta))$ *for arbitrary* x *from some fixed base set. We can then choose a threshold* ε *and define the* classifier C *as*

$$C(x) := \begin{cases} 0 & \text{if } P(x|\mathcal{M}(\theta)) < \varepsilon, \\ 1 & \text{otherwise}. \end{cases} \tag{2.5}$$

Note, that the concept of classification with respect to membership in one class can naturally and readily be extended to classification for a finite number of classes.

2.2 Bayesian Statistics

Bayes' formula appears innocuous and an implausible candidate for causing philosophical, and scientific, arguments on a first encounter. A comprehensive account on its history and the history and development of the statistical school of thought spawned by it can be found in [48]. Here, we can only try to give a flavor and fix the notation we will use. The basic statistical terminology can be found in a large number of textbooks, e.g. in [28].

We will use $P(\cdot)$ to denote the probability of an event, which will always be understood to be a subset of an implicitly given sample space. If the set of events $\{B_1, B_2, \ldots, B_k\}$, $k \in \mathbb{N}$ forms a partition of the sample space, we can define a discrete probability distribution $P = (P(B_1), P(B_2), \ldots, P(B_k))$. Performing an experiment described by P corresponds to drawing a sample according to P.

The *joint probability* of two events, $P(A, B)$, is simply defined as

$$P(A, B) := P(A \cap B). \tag{2.6}$$

The *conditional probability* of two events, $P(B|A)$, where

$$P(B|A) := \frac{P(A, B)}{P(A)}, \tag{2.7}$$

is the probability of observing B given that one has already observed A.

Theorem 2.5 (Bayes' theorem (1763)) *Let A and B be two events. Then*

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}. \tag{2.8}$$

If we consider the formula above with data D and a model M instead of the events A and B, we obtain

$$P(M|D) = \frac{P(D|M)P(M)}{P(D)}, \tag{2.9}$$

and the Bayesian point of view in modeling the data with a statistical model becomes apparent. It allows to assign probabilities to *models* given the data and, often, to efficiently compute the conditional probability on the right hand side. In this context, $P(M|D)$ is the *a posteriori* probability of M. It is called posterior, since it is the probability after having seen the data. Analogously, the probability $P(M)$ is called the *a priori* probability of M, prior to observing any data. The corresponding distributions are called *posterior* and *prior*, respectively. The *likelihood*, $P(D|M)$, is the probability that the model M produces the data D.

If we select a model M such that Eq. (2.9) is maximized, we call M a *maximum a posteriori* (MAP) model. Since $P(D)$ is constant in the maximization, we will typically consider

$$P(M|D) \propto P(D|M)P(M). \tag{2.10}$$

The prior $P(M)$ should encode our complete *belief* about the peculiarities of the model. Alternatively, we can use the prior to drive the process of MAP model selection towards models we

would *prefer*, such as more parsimonious models or models with a larger degree of generalization power. Note, that in case of a uniform prior the MAP coincides with the maximum likelihood model. A uniform prior is also called an *uninformed* prior.

We will also need the following two basic definitions.

Definition 2.6 *A discrete probability distribution* $P = (P(A_1), \ldots, P(A_n))$ *is called* singular, *if it puts all its weight on one event. That is, if there is one* $1 \leq j \leq n$, *s.t.*

$$P(A_i) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

Definition 2.7 *A vector* $P = (p_1, \ldots, p_n)$ *is called* stochastic, *if for* $1 \leq i \leq n$, $p_i \geq 0$ *and* $\sum_{i=1}^{n} p_i = 1$. *Similarly, a matrix A is called* (row) stochastic, *if its rows are stochastic vectors.*

2.3 Stochastic Processes and Markov Chains

Stochastic processes can represent data such as time series:

Definition 2.8 *A* stochastic process *X with discrete time is a sequence of random variables*

$$X = \{X_t\}_{t \geq 0}, \tag{2.11}$$

where X_t *denotes the value at time* $t = 0, 1, 2, \ldots$. *A* realization *of a stochastic process is a sequence* x_t *such that*

$$\{X_t = x_t\}_{t \geq 0}. \tag{2.12}$$

Stochastic processes can have unbounded memory in general. That is, the probability $P(X_{t+1}|X_t, X_{t-1}, \ldots, X_1)$ depends on *all* $X_1, \ldots, X_t$. In the following, we will introduce some relevant properties of stochastic processes.

Definition 2.9 *A stochastic process with discrete time is called* stationary (in the strong sense), *if its finite joint distributions are translation invariant. That is, if*

$$P(X_{t_1+t} = x_1, \ldots, X_{t_m+t} = x_m) = P(X_{t_1} = x_1, \ldots, X_{t_m} = x_m) \tag{2.13}$$

for arbitrary $m \in \mathbb{N}$, x_i, *and all* $t, t_i \geq 0$, $1 \leq i \leq m$.

Another relevant concept is ergodicity, whose concern is the limiting behavior of averages over time.

Theorem 2.10 *[59] Let* $\{X_i\}_{i \geq 0}$ *be a sequence of independently and identically distributed random variables with expectation value* $\mathbf{E}(X_1) = \mu$. *Then the* time average

$$\mu_t := \frac{1}{t+1} \sum_{t=0}^{t} X_t, \quad t \geq 0, \tag{2.14}$$

converges to μ *as t goes to infinity with probability one.*

If the limit condition above holds, we will call a process ergodic:

Definition 2.11 *Given a stochastic process X stationary in the strong sense. We will call X* ergodic, *if for* all *realizations of X,* $\{X_i = x_i\}_{i\geq 0}$*, the time averages*

$$\mu_{st} := \frac{1}{t-s}\sum_{u=s}^{t} x_t, \quad t > s, \tag{2.15}$$

converge to the expectation $\mathbf{E}(X_s)$ *with probability one, as — for arbitrary fixed* $s \geq 0$ *— t goes to infinity.*

An important special class of processes are Markov chains.

Definition 2.12 *A stochastic process is said to have the* Markov property, *iff*

$$P(X_{t+1}|X_t, X_{t-1}, \ldots, X_1) = P(X_{t+1}|X_t). \tag{2.16}$$

Stochastic processes with the Markov property are also called (first-order) Markov chains. *The events in the sample space of the* X_i *are then called* states, *denoted* $S_1, S_2, \ldots, S_n$*. The relevant probabilities can be represented by the so-called* transition matrix $A = \{a_{ij}\}$, *where*

$$a_{ij} := P(X_{t+1} = S_j|X_t = S_i), \tag{2.17}$$

for $1 \leq i, j \leq n$.

Definition 2.13 *Given a Markov chain, the stochastic vector* $\pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ *is called an* equilibrium vector, *if*

$$A^T\pi = \pi, \tag{2.18}$$

where A^T *denotes the transpose of A. Equilibrium under these circumstances is understood as an invariance of the state probabilities* $\pi_i = P(S_i)$, $1 \leq i \leq n$ *as time progresses.*

Theorem 2.14 *[59] A Markov chain is ergodic, if it has a unique equilibrium distribution, to which the relative sample state frequencies* $\frac{1}{t}\{|S_1|, \ldots, |S_n|\}$ *converge as* $t \to \infty$.

Sufficient conditions for existence and uniqueness of an equilibrium distribution are rather mild and given in the following.

Definition 2.15 *A set* $\Gamma \neq \emptyset$ *of states* S_i *is called* stochastically closed, *iff for all* $i \in \Gamma$*:*

$$\sum_{j\in\Gamma} a_{ji} = 1. \tag{2.19}$$

The set Γ *is called* minimal, *if it contains no proper closed subset. A Markov chain is called* irreducible, *iff its state space is minimally closed.*

Definition 2.16 *An irreducible Markov chain with transition matrix A is* periodic, *if there exists an integer $d \geq 2$ and a sequence of states $\{S_0, S_1, \ldots, S_{d-1}\}$ such that for all $i = 0, \ldots, d-1$*

$$a_{ij} = 1 \quad \text{for } j \equiv i+1 \ (\text{mod } d). \tag{2.20}$$

Otherwise the chain is called aperiodic.

Theorem 2.17 *[59] If a Markov chain is irreducible and aperiodic, then it is ergodic.*

Markov chains seem to have a "memory" of length one at first, since only the immediate preceding states are relevant for transitions. Nevertheless, this is not true [33], as can be seen from the following example, which shows how to represent a higher-order process as a (first-order) Markov chain.

Example 2.18 *Given a process $\{Z_t\}_{t\geq 0}$ of memory length $k \in INTS$, such that*

$$P(Z_0, Z_1, Z_2, \ldots) = \prod_{i\geq 0} P(Z_i | Z_{i-1} Z_{i-2} \cdots Z_{i-k}). \tag{2.21}$$

We can define another process by

$$X_i := Z_i Z_{i-1} \cdots Z_{i-k+1}, \tag{2.22}$$

and this process has the Markov property, even if at the cost of a larger state space of size N^k, with N denoting the size of the state space of $\{Z_t\}_{t\geq 0}$.

2.4 Information Theory

In the following, we will only briefly define the relevant concepts and definitions from information theory which we need for the further exposition. An introduction to information theory and the concept of information, as well as an axiomatic development of entropy etc., can be found in the ground-breaking article by Shannon [70] and also, for example, in [5]. Ibidem, proofs for the results in this section can be found. The concepts introduced are meaningful both for random variables and for probability distributions, even if we will only define them for one or the other. Either ways, we will only deal with discrete random variables and discrete probability distributions.

Definition 2.19 (Entropy) *The entropy $\mathcal{H}(P)$ of a discrete probability distribution $P = (p_1, \ldots, p_n)$ is defined by*

$$\mathcal{H}(P) := -\sum_{i=1}^{n} p_i \log p_i. \tag{2.23}$$

If the base of the logarithm is 2, as it is the usual convention, we say that entropy is measured in bits.

One interpretation of entropy is the prior uncertainty in the outcome of the random experiment described by the probability distribution P, or, alternatively, the gain in information when the outcome is observed.

Lemma 2.20 *[70] Given a discrete probability distribution $P = (p_1, \ldots, p_n)$, the following holds:*

$$0 \leq \mathcal{H}(P), \tag{2.24}$$

with equality iff P is singular, and

$$\mathcal{H}(P) \leq \log(n), \tag{2.25}$$

with equality if and only if P is uniform. The former makes use of the convention $p \log p := 0$ for $p = 0$.

When interpreting entropy as a measure on stochastic sources with possibly differently sized discrete state spaces, we may want to normalize it:

Definition 2.21 (Normalized entropy) *With P as above, define*

$$\mathcal{HN}(P) := \frac{\mathcal{H}(P)}{\log(n)}, \tag{2.26}$$

the normalized entropy. *$\mathcal{HN}(P)$ is in the interval $[0, 1]$.*

A more general concept than entropy is relative entropy:

Definition 2.22 (Relative Entropy) *The relative entropy $\mathcal{H}(P, Q)$ between two probability distributions $P = (p_1, \ldots, p_n)$ and $Q = (q_1, \ldots, q_n)$, defined by*

$$\mathcal{H}(P, Q) := \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i}, \tag{2.27}$$

is a "measure" of the distance between P and Q.

Relative entropy is also known as cross-entropy, Kullback-Liebler distance, or discrimination [70]. It is *not* a distance in the mathematical sense, as it is not symmetric. If symmetry is crucial, divergence [70] can be used instead of relative entropy:

Definition 2.23 (Divergence) *The divergence $\mathcal{D}(P, Q)$ between two probability distributions $P = (p_1, \ldots, p_n)$ and $Q = (q_1, \ldots, q_n)$ is defined by*

$$\mathcal{D}(P, Q) := \mathcal{H}(P, Q) + \mathcal{H}(Q, P). \tag{2.28}$$

Lemma 2.24 *Let P and Q be two probability distributions, then*

$$\mathcal{H}(P, Q) \geq 0, \tag{2.29}$$

with equality if and only if P = Q.

That entropy is a special case of relative entropy can be seen by considering $\mathcal{H}(P, U)$. If U is the uniform distribution and P is defined as above, then $\mathcal{H}(P, U) = \log(n) - \mathcal{H}(P)$.

In a Bayesian setting, relative entropy can be used to measure the gain in information due to observation of an experiment. That is, going from the *prior* distribution $(P(A_1), P(A_2), \ldots, P(A_n))$ on events A_i to the *posterior* distribution conditioned on the outcome B of the experiment, $(P(A_1|B), P(A_2|B), \ldots, P(A_n|B))$:

$$\mathcal{H}(\text{posterior}, \text{prior}) := \sum_{i=1}^{n} P(A_i|B) \log \frac{P(A_i|B)}{P(A_i)}. \tag{2.30}$$

Another information theoretic concept relating two probability distributions is mutual information:

Definition 2.25 (Mutual Information) *Given two random variables X and Y, and their respective marginal distributions P(X) and P(Y) as well as the joint distribution P(X, Y). Then,* mutual information*, defined as*

$$\mathcal{I}(X; Y) := \sum_{x,y} P(X = x, Y = y) \log \left(\frac{P(X = x, Y = y)}{P(X = x)P(Y = y)} \right), \tag{2.31}$$

measures the degree of independence between the two random variables.

Alternatively, mutual information can be expressed in terms of conditional entropy:

Definition 2.26 (Conditional entropy) *Let X and Y be two random variables. Then, the* conditional entropy *of X given Y is defined as*

$$\mathcal{H}(X|Y) := -\sum_{y} P(Y = y) \sum_{x} P(X = x|Y = y) \log(P(X = x|Y = y)). \tag{2.32}$$

It reflects the average amount of uncertainty about X after observation of Y. From a Bayesian point of view, this corresponds to the uncertainty in the posterior distribution. The uncertainty in the prior is simply $\mathcal{H}(X)$. The difference between the uncertainties captures the average gain in information an observation of Y brings about X.

Remark 2.27 *We can express mutual information in terms of (conditional) entropy as follows:*

$$\mathcal{I}(X; Y) = \mathcal{H}(X) - \mathcal{H}(X|Y). \tag{2.33}$$

2.5 Graphs

Graphs are natural mathematical models for many real-world entities — communication networks, electronic circuits, to just name two examples — and constitute important data-structures in computer science. We will use special types of graphs, so called trees, as the core of our algorithm, and also use directed graphs as alternative representation of Markov chains and Hidden Markov models.

We will use the following definition: A graph $G = G(V, E)$ consists of a finite set of vertices, $V = V(G)$, and a finite set of edges, $E = E(G)$. An edge $e \in E$ is a pair $\{v, w\}$ with $v \in V$ and $w \in V$. The cardinality of the set V is called the *order* of G, the cardinality of the set E is called the *size* of G. A graph is *labeled* if names, such as $v_1, v_2, \ldots, v_n$ or simply $1, 2, \ldots, n$, are assigned to all its vertices.

A graph is called *simple*, if it is undirected, and has no loops or multiple edges. That is, the edges of the graph G are unordered pairs, $\{v, w\} \in E$ implies $v \neq w$, and for all $v \in V$ and $w \in V$ there is at most one $\{v, w\} \in E$. A graph is called *directed*, or, for short, *digraph*, if the edges are ordered pairs (v, w).

If $e = \{v, w\} \in E$, then we say that the vertices v and w are *adjacent* and that v and w are *incident* to edge e; v and w are called *nonadjacent* otherwise. If w is adjacent to v we also say that w is a *neighbor* of v. The set of all neighbors of a vertex v is called the *neighborhood* of v and is denoted by $N(v)$. The cardinality of the neighborhood of v is the *degree* or *valency* of v in G, written as $deg_G(v)$ or simply $deg(v)$ if the graph G is understood from the context. An *isolated vertex* is a vertex which has no neighbors. If all the vertices in a graph have the same degree, then the graph is called *regular* or *d-regular* where d is the degree of any vertex.

An alternating sequence $v_1, e_1, v_2, \ldots, v_k, e_k, v_{k+1}$ of vertices and edges is called a *walk* of the graph G, if the vertices v_i and v_{i+1} are incident with the edge e_i for $i = 1, 2, \ldots, k$. The number k of edges in the walk is the *length* of the walk. Since in a simple graph the edges in the walk are uniquely determined by the vertices, they are often omitted and the walk is denoted by $v_1, v_2, \ldots, v_k, v_{k+1}$. A walk is said to be *closed* if the first and the last vertex on a walk are equal, and *open* otherwise. If all the edges on a walk are distinct, we speak of a *trail*. Furthermore, if all vertices, except possibly the first and the last one, are distinct we speak of a *path*. A *cycle* in a graph G is a closed path. In all cases the short notation v_1–v_k for a walk, trail or path between v_1 and v_k is used. The *distance* between two vertices is defined as the number of edges of a shortest path connecting them.

A (directed) graph is called (strongly) connected if there there is a (directed) path connecting any (ordered) pair of vertices. Maximal, with respect to inclusion, subsets of vertices fulfilling the previous conditions are called (strongly) connected components of a graph.

A connected graph T is called a tree, if $size(T) = order(T) - 1$. Note that this is equivalent to T containing no cycles. We will usually consider *rooted trees*; i.e., trees in which one vertex, the *root*, is distinguished. The vertices v with $deg(v) = 1$ are called *leaves* or *terminal vertices*, the vertices v with $deg(v) \geq 2$, except the root, are called *internal* or *nonterminal* vertices.

Each vertex v in a rooted tree has exactly one *predecessor* or *parent*, which is the adjacent vertex on the unique path from v to the root. All vertices which have the same vertex w as a parent, are called *children* of w. The depth, depth(v) of a vertex is its distance from the root. The

depth of a tree is the maximum depth of vertices in the tree. We can partition vertices into *levels*. Level k contains all vertices of depth k.

Edge weights are maps from the set of edges to some set $\mathcal{W}$,

$$\begin{aligned} w: \quad E(G) &\longrightarrow \mathcal{W} \\ e &\longmapsto w(e) \quad . \end{aligned}$$

Edge weights can for example be probabilities or symbols or characters associated with an edge. Similarly, we will use vertex weights.

We say that H is a *subgraph* of G, if $V(H) \subset V(G)$ and $E(H) \subset E(G)$. If $\{v, w\} \in E(H)$ whenever $v, w \in V(H)$ and $\{v, w\} \in E(G)$, we call H an *induced subgraph* of G.

If $V(H) = V(G)$, the subgraph H of G is called a *spanning subgraph*. In all cases G is called a *super graph* of H.

2.6 Strings and Things

We will model the biological sequences we will ultimatively encounter as words or strings over an alphabet. The definitions follow [27].

Definition 2.28 *A finite set of characters, or symbols, $\Sigma = \{a, b, c, \dots\}$ we will call an* alphabet. *A sequence s of characters from Σ is called a* string, *or a* word. *We denote by*

- $|s|$ *its* length, *that is its number of characters, by*
- s_i *its i-th character, if $i > 0$, extending the notion to negative i corresponding to the $(-i)$-th character from the end of s, that is $s_{|s|-i+1}$, and by*
- $s[i, j]$ *the continuous* sub-string *starting at position i and ending in position j.*

In the biological sciences and in bioinformatics, the term sequence is used predominantly instead of string. If we do so, *sub-sequence* denotes a continuous *sub-string* and *not* a sub-sequence in the usual mathematical sense.

Definition 2.29 *Given an alphabet Σ, we denote with*

```
ACCGGTTAGGAGTTCC
ACCGGTT
 CCGGTTA
  CGGTTAG
     ...
        AGGAGTT
         GGAGTTC
          GAGTTCC
```

Figure 2.2: *For a DNA-sequence of length 16 (top) the windows of length 7 are depicted (bottom).*

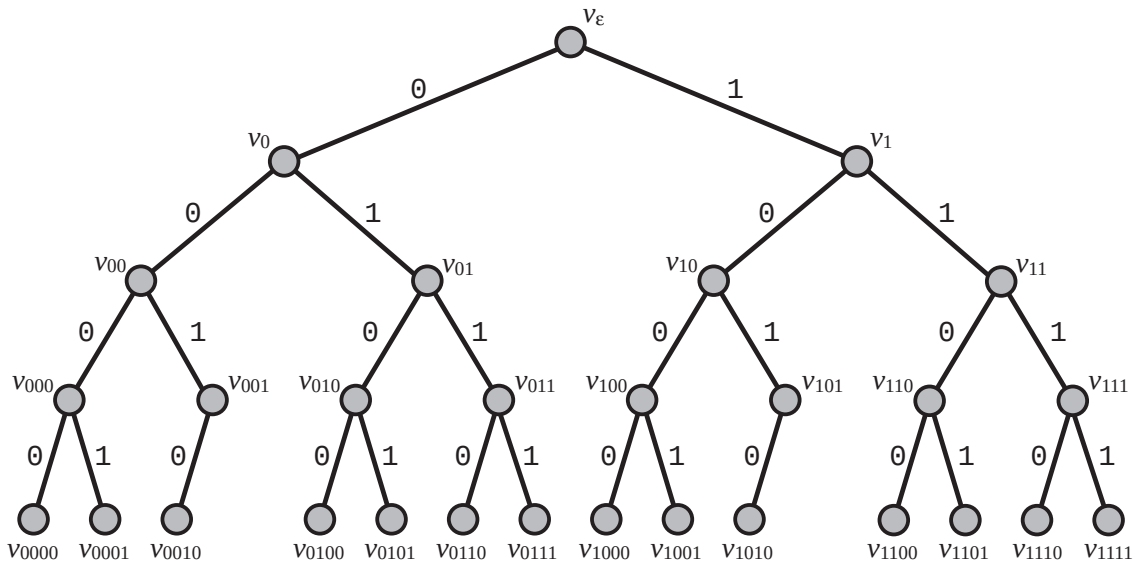

Figure 2.3: *A Prefix tree. The counting variables $c(\cdot)$ associated with vertices are omitted in this picture.*

- $\Sigma^\star$ *the set of all possible strings of arbitrary length over the alphabet* Σ*, with*
- $\Sigma^{\leq k}$ *the subset of* $\Sigma^\star$ *containing strings of length at most* $k \in \mathbb{N}$*, and with*
- Σ^k *the subset containing strings of length exactly* $k \in \mathbb{N}$*.*

Definition 2.30 *Let $s \in \Sigma^\star$. Then, with $1 \leq i, j \leq |s|$, we call*

- $s[1, j]$*, a* prefix*, and*
- $s[i, |s|]$ *a* suffix

of s. A suffix or a prefix are called proper *if they are not equal to the whole string.*

Definition 2.31 *We will write the* concatenation *of two strings s and t as st, s repeated k times as s^k, and the empty string as ε.*

Remark 2.32 ([51]) *If we denote concatenation of strings with + then, $(\Sigma^\star, +)$ is a monoid with the identity element ε.*

We will later on need to divide a given long string into all continuous pieces of some fixed length by sliding a window over the string (e.g., Fig. 2.2).

Definition 2.33 (Window set) *Given a string s and an integer w. $W_w(s)$ is the following set of sub-strings of length w:*

$$W_w(s) := \{s[1, w], s[2, w+1], \ldots, s[|s| - w + 1, |s|]\}$$

Originally motivated from speeding up multi-pattern exact string matching algorithms, *keyword* trees [27] provide a compact representation for a set of strings. To signify the use of prefixes and since, as will be seen, we perform additional counts, we will call them prefix-trees.

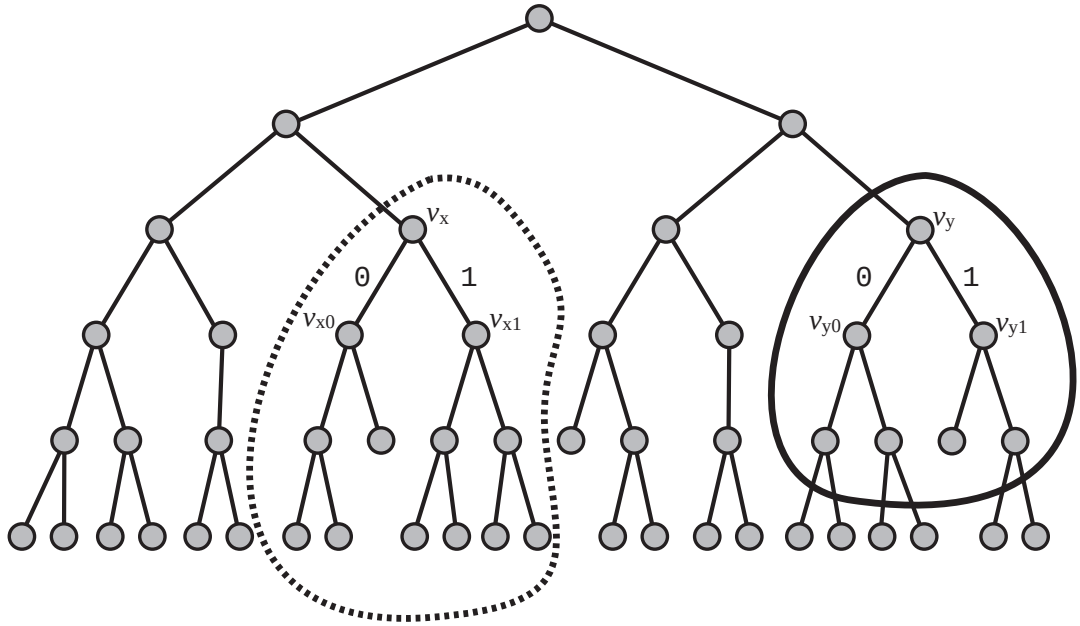

Figure 2.4: *In this example of a prefix tree, $PT(\mathcal{S})$, $PT_x(\mathcal{S})$ is the subtree rooted in v_x circled with the dashed curve and the 2-tail rooted in v_y is the subtree circled with the solid curve on the right. The vertices in both subtrees have x respectively y as a prefix, as exemplified with v_{x0}, v_{x1} etc.*

Definition 2.34 (Prefix Tree) *Given a set $\mathcal{S} = \{s^1, s^2, \ldots, s^n\}$ of strings over an alphabet Σ. The prefix tree is a rooted tree, satisfying*

- *each edge is labeled with exactly one character,*
- *any two edges out of the same node have distinct labels,*
- *there is a bijection between the s^i and leaves, such that the concatenation of edge labels on the path from the root to v equals s^i.*

More formally, the vertices v_x of the prefix tree $PT = PT(\mathcal{S})$ correspond to prefixes x present in $\mathcal{S}$. Thus, PT has vertex set

$$V(PT) := \{v_x | \exists\, y \in \mathcal{S} : x \text{ is a prefix of } y\} \tag{2.34}$$

and edge set

$$E(PT) := \{(v_x, v_y) | x = y[1, |y| - 1]\}. \tag{2.35}$$

The vertex degree of PT is bounded by $|\Sigma|$ and its root is v_ε. Define $c(v_x)$ to be the number of strings in $\mathcal{S}$, whose prefix is x. By the convention of considering ε to be a prefix of any string, this yields that $c(v_\varepsilon)$ is $|\mathcal{S}|$.

An example of a prefix tree is depicted in Fig. 2.3.

Lemma 2.35 *A prefix tree can be built in time $O(n)$, where n is the total of the lengths of the individual strings in $\mathcal{S}$.*

Proof. Since the counting variables $c(v_x)$ can be updated with one elementary operation, the proof for keyword trees [27, pp. 52] carries through. □

Our approach relies on particular subtrees, as a distinguishing characteristic of prefix tree vertices.

In Fig. 2.4 one can find an example for the concept of *k*-tails introduced in the following definition.

Definition 2.36 (*k*-Tail) *Given a prefix tree $PT(\mathcal{S})$, we will denote with $PT_z(\mathcal{S})$ the sub-tree of $PT(\mathcal{S})$ rooted in z induced by the vertex set*

$$V(PT_z) = \{v_x | v_x \in V(PT), z \text{ is a prefix of } x\}.$$

Similarly, we denote with $PT_{z,k}(\mathcal{S})$, $1 \leq k \leq \text{depth}(PT(\mathcal{S})) - \text{depth}(z)$ the subtree rooted in v_z containing only vertices of distance at most k from v_z. That is, the subtree induced by the vertex set

$$V(PT_{z,k}) = \{v_x | v_x \in V(PT), z \text{ is a prefix of } x, |x| - |z| \leq k\}.$$

$PT_{z,k}(\mathcal{S})$ will be called the k-subtree or k-tail of v_z. We will omit the argument $\mathcal{S}$ when it is clear from the context.

Chapter 3

Hidden Markov Models

Hidden Markov Models (HMMs) have been applied to a large number of applications in statistical pattern recognition and classification with great success. They provide a sound statistical framework, and allow for efficient and numerically stable algorithms. Their visualization as graphs, see Fig. 3.1, constitutes an effective user interface for modeling.

HMMs have been investigated under the name of *(probabilistic) functions of a Markov chain* from the 1940s to the 1960s [12,14,24,29,62]. The classical development was mostly theoretical, considering questions of uniqueness and identifiability, and did not result in wide spread popularity. Starting in the 1970s, after Baum et al. [9] had derived an efficient training algorithm for HMMs, and fueled by their prowess when applied to engineering problems, this quickly changed. By now they have become a basic tool in the applied sciences. The most prominent applications can be found in speech recognition and in the computational analysis of biological sequences. In fact, the field was dominated by a group of researchers at AT&T [35, 36, 38], working on the former, in the beginning. A tutorial covering the theoretical foundations of HMMs and their application in speech recognition can be found in [65], and a description of the current state of the art in a recently published book [33].

In molecular biology, HMMs are used to compute multiple sequence alignments [20], model classes, families and domains of protein sequences [13,26,43,44,46], identify specific functional protein classes based on motifs [78], recognize promoter sites in eukaryotic DNA [61] and translational units in procaryotes [75], find genes [2,6,42,45,47,77], predict protein structure [39,40], perform fold recognition [18] and identify remote homologs [71]. Overviews and tutorials can be found in books [5, 19] and articles [21, 25, 43].

HMMs have also been applied to problems from a wide range of other fields. They have been used to predict international crises [68] in the Political Sciences, to model and analyze ion channels [10] in Physiology, and to control anti-tank guided missiles [56–58] in Electric Engineering.

In the following, we will introduce the basic theory, following the notation in Rabiner's tutorial [65].

3.1 Definition of an HMM

HMMs are statistical processes consisting of two distinct components. There is a discrete-state, time-homogeneous, first-order (cf. assumption 3.2) Markov chain (MC) with appropriate transition probabilities between states and an initial distribution. Associated with each state is a discrete or continuous distribution over possible emissions or outputs, which is, in general, distinct per state. The probability of an emission depends only on the state the MC is in; neither history of states nor previous outputs matter.

When producing sequences of emissions, see Sec. 3.5, only the output values — in the discrete case typically from an alphabet of size *smaller* than the number of states — can be observed. The sequence of states of the underlying MC *cannot* be observed, which motivates the name Hidden Markov Model.

Definition 3.1 (Hidden Markov Model) *We will use the following notation for a Hidden Markov Model (HMM):*

N	*number of states*
$\mathcal{S} = \{S_1, S_2, \ldots, S_N\}$	*set of states*
$A = \{a_{ij}\}_{1 \le i,j \le N}$	*transition matrix*
$\pi = (\pi_1, \pi_2, \ldots, \pi_N)$	*initial distribution over states*
M	*size of the output alphabet*
$\Sigma = \{v_1, v_2, \ldots, v_M\}$	*output alphabet*
$B = \{b_{jm}\}_{1 \le j \le N, 1 \le m \le M}$	*emission matrix*
T	*length of an observation sequence*
$O = O_1 O_2 \cdots O_T$	*observation sequence;* $O_t \in \Sigma$
$Q = q_1 q_2 \cdots q_T$	*state sequence;* $q_t \in \mathcal{S}$

The elements of the matrices A, B and π correspond to the following probabilities:

$$\begin{array}{lcll} \pi_i & := & P(q_1 = S_i) & \text{for all } i \\ a_{ij} & := & P(q_{t+1} = S_j | q_t = S_i) & \text{for all } i, j, t \\ b_j(O_m) = b_{jm} & := & P(O_t = v_m | q_t = S_j) & \text{for all } j, m, t \end{array} \tag{3.1}$$

Hence A and B are row-stochastic matrices and π is a stochastic vector; i.e., all the elements are non-negative and the sums

$$\sum_{j=1}^{N} a_{ij} = \sum_{j=1}^{N} \pi_j = \sum_{m=1}^{M} b_{im} = 1; \qquad \text{for all } i. \tag{3.2}$$

A HMM is completely specified by the quantities A, B, π, N, M, if we use the tacit convention that states and output symbols are denoted by $1, 2, \ldots, N$ and $1, 2, \ldots, M$ respectively. Let

$$\lambda = (A, B, \pi) \tag{3.3}$$

denote both the model and the set of parameters.

This definition implies the following assumptions, which we would like to state explicitly, since they are the basis for our further development.

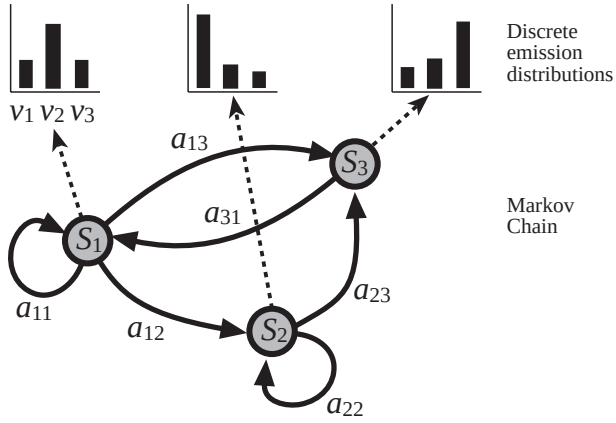

Figure 3.1: *A simple HMM displayed as a directed graph*

Assumption 3.2 *HMMs satisfy:*

1. **First order MC**. *The transition probability from one state to another state depends only on the two states, and is independent from states previously visited and from observations. That is:*

$$P(q_{t+1} = S_j | q_t = S_i, q_{t-1} = S_{i_{t-1}}, \ldots, q_1 = S_{i_1}) = P(q_{t+1} = S_j | q_t = S_i) \quad \textit{for all } i, j, t. \tag{3.4}$$

   This is often referred to as the Markov condition.

2. **Independent emissions**. *The probability of an emission at time t depends only on the state at time t*

3. **Time homogeneity**. *The transition probabilities are independent of time; i.e.:*

$$P(q_{t+1} = S_j | q_t = S_i) = P(q_{u+1} = S_j | q_u = S_i) \quad \textit{for all } t, u \geq 1. \tag{3.5}$$

In Fig. 3.1, we have depicted an HMM as a directed graph. The vertices of the graph correspond to the states of the underlying MC, edges to transitions between states weighted by the respective transition probabilities. Zero entries in the transition matrix correspond to non-edges. The transition matrix, A, of the MC can be considered as the weighted adjacency matrix of the graph. The discrete output distributions is depicted by dashed arrows. The initial distribution is not displayed. We will switch freely between the matrix and graph representations in the sequel.

3.2 The Three Fundamental Problems

Rabiner [65] alludes to three fundamental problems arising naturally and immediately, when using HMMs in practical applications.

Problem 3.3 *Let a model λ and a finite observation sequence $O = O_1 O_2 \cdots O_T$ be given.*

1. *How can one compute the likelihood $P(O|\lambda)$ efficiently?*
2. *Which state sequence $Q = q_1 q_2 \cdots q_T$ is an "optimal" explanation for O?*

3. *Considering λ as an initial "guess", how should one adjust parameters of λ as to maximize $P(O|\lambda)$?*

We motivate the relevance of these questions with the following two examples:

Example 3.4 *A typical application of HMM in molecular biology is finding distantly related protein sequences [20]. Typically, some protein sequences of a certain family are known. With the parameter optimization one tries to find a model λ which maximizes the sum of probabilities of the training sequences (problem 3. above).*

Subsequently, we compute for every sequence O in a protein sequence database such as Swissprot [3] $P(O|\lambda)$ (problem 1. above) and rank the sequences according to their probability. Highly ranked sequences will likely be additional members of the protein family in question.

In this example the number of training sequences might be of the order of hundreds. As of March 19, 2001, the protein sequence database Swissprot for example contained 94,152 sequence entries.

Example 3.5 *Let a multiple alignment [63, pp. 123] of a set of protein sequences and a corresponding profile HMM [19, pp. 100] (see. Fig. 3.9) be given. In such an HMM, states fall into three different classes — match, insertion and deletion — with respect to the consensus [20] of the multiple alignment. The states indicate whether for a given sequence, a residue matches the consensus, has additional residues inserted, or has residues from the consensus removed.*

By computing the optimum state sequence, we compute the alignment of a sequence to the consensus (problem 2. above).

The exposition in the following six sections closely follows [65].

3.2.1 The Forward-Backward Algorithm

The likelihood $P(O|\lambda)$ can be computed in a closed form expression as a sum over all state sequences Q

$$\begin{aligned} P(O|\lambda) &= \sum_Q P(O, Q|\lambda) \\ &= \sum_Q P(O|Q, \lambda)P(Q|\lambda) \\ &= \sum_Q \pi_{q_1} b_{q_1}(O_1)a_{q_1q_2}b_{q_2}(O_2)\cdots a_{q_{T-1}q_T}b_{q_T}(O_T). \end{aligned}$$

The exponential complexity of this naïve approach — more exactly, we obtain a worst-case complexity of $O(TN^T)$ for a fully connected model as can be seen from Fig. 3.2 — can be substantially reduced with a dynamic programming approach.

Definition 3.6 (Forward variables) *The* forward variables $\alpha_t(i)$, $1 \le t \le T$, *denote the probability of being in state i and observing $O_1O_2\cdots O_t$ given the model λ:*

$$\alpha_t(i) := P(O_1O_2\cdots O_t, q_t = i|\lambda). \tag{3.6}$$

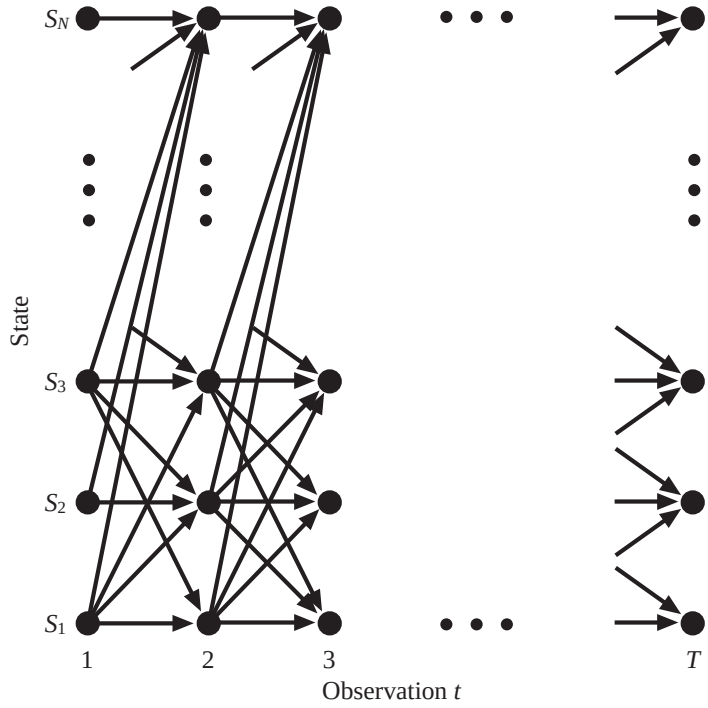

Figure 3.2: *This diagram shows the exponential number of paths through the state-observation space.*

Lemma 3.7 *The forward variables can be computed in time $O(TN^2)$ with the following induction:*

1. *Initialization for $t = 1$:*

$$\alpha_1(i) = \pi_i b_i(O_1), \quad \textit{for all } i. \tag{3.7}$$

2. *Induction for $t = 1, 2, \ldots, T-1$:*

$$\alpha_{t+1}(i) = \sum_{j=1}^{N} \alpha_t(j) a_{ji} b_i(O_{t+1}), \quad \textit{for all } i. \tag{3.8}$$

Proof. As shown in Fig. 3.3, we have to perform $2N$ operations to sum up the probabilities $\alpha_t(j)$ for each $\alpha_{t+1}(i)$, weighted by the respective transition probability to state i and the probability of observing O_{t+1} in state i. For each point in time, there are N forward variables, yielding the aforementioned complexity. The correctness of the induction follows immediately from

$$P(O_1 O_2 \cdots O_t, q_{t+1} = i|\lambda) = \sum_{j=1}^{N} P(O_1 O_2 \cdots O_t, q_t = j|\lambda) a_{ji}. \tag{3.9}$$

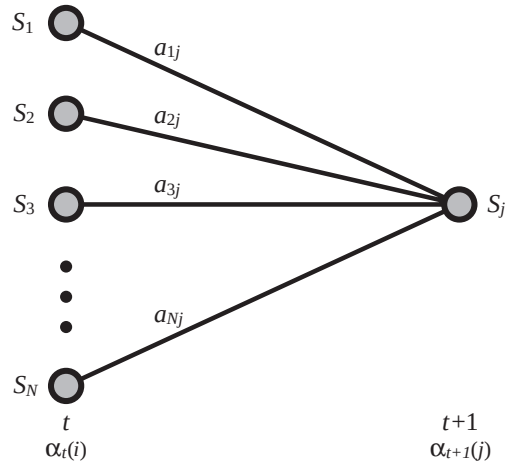

Figure 3.3: *The inductive definition of the forward variables $\alpha_{t+1}(j)$ is shown.*

□

To train the model parameters, we make use of the following auxiliary variables:

Definition 3.8 (Backward variables) *The* backward variables $\beta_t(i)$ *are defined as the probabilities of observing the suffix $O_{t+1}, O_{t+2}, \ldots, O_T$ of the observation sequence, given state i at time t and λ:*

$$\beta_t(i) := P(O_{t+1}, O_{t+2}, \ldots O_T | q_t = i, \lambda). \tag{3.10}$$

Note, that the $\beta_t(i)$ complement the $\alpha_t(i)$ in the sense that they capture the probability of observing the remainder of the observation sequence $O_{t+1}O_{t+2}\ldots O_T$ starting from state i at time t and given the model λ.

Lemma 3.9 *The backward variables can be computed in time $O(TN^2)$ by induction:*

1. *Initialization for $t = T$:*

$$\beta_T(i) := 1, \quad \text{for all } i. \tag{3.11}$$

2. *Induction for $t = T-1, T-2, \ldots, 1$:*

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j), \quad \text{for all } i. \tag{3.12}$$

Proof. Analogous to the proof of Lemma 3.7 using Fig. 3.4. □

Remark 3.10 *As $\alpha_t(i)\beta_t(i)$ is the probability of observing O and being in state i at time t, given the model λ, we can express the likelihood of the observation sequence in terms of the forward*

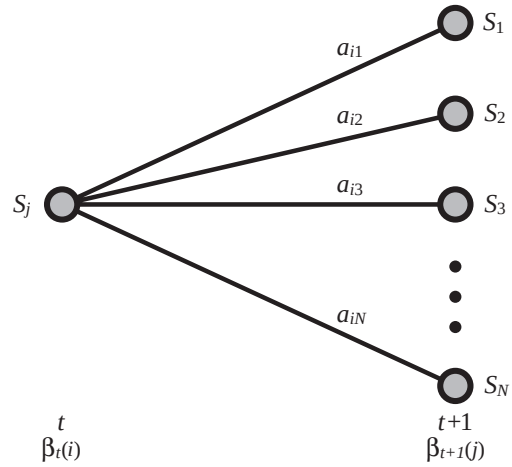

Figure 3.4: *The inductive definition of the backward variables $\beta_t(i)$ is shown.*

and/or backward variables, and the following equalities

$$\begin{aligned} P(O|\lambda) &= \sum_{i=1}^{N} P(q_t = i, O|\lambda); \ \textit{for all } t \\ &= \sum_{i=1}^{N} \alpha_t(i)\beta_t(i); \ \textit{for all } t \\ &= \sum_{i=1}^{N} \alpha_T(i). \end{aligned} \tag{3.13}$$

3.2.2 The Viterbi Algorithm

Asking for an "optimal" state sequence, the second problem in 3.3, can also be dealt with efficiently, if the criterion for optimality is chosen to be the maximization of the probability $P(O, Q|\lambda)$ over all possible state sequences Q.

Definition 3.11 (Viterbi path) *A state sequence Q^* is called a* Viterbi path, *if*

$$P^* := P(O, Q^*|\lambda) \geq P(O, Q|\lambda) \quad \textit{for all } Q. \tag{3.14}$$

Analogously to the previous section, we define auxiliary variables $\delta_t(i)$. For a partial observation sequence $O[1, t] = O_1 O_2 \cdots O_t$ we define

$$\delta_t(i) := \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1 q_2 \cdots q_{t-1} q_t = i, O_{[t]}|\lambda). \tag{3.15}$$

That is, $\delta_t(i)$ is the maximal probability over all partial state sequences $Q[1, t] = q_1 q_2 \cdots q_t$ ending in $q_t = i$, while observing $O[1, t]$.

Lemma 3.12 *Given an HMM λ and using the auxiliary variables $\delta_t(i)$ we can compute a Viterbi path Q^* in time $O(TN^2)$.*

Proof. Note, that we immediately obtain the following inductive formula from the definition

$$\delta_{t+1}(j) := \max_{1\le i\le N}(\delta_t(i)a_{ij})b_j(O_{t+1}),$$

and thus $P^* = \max_{1\le i\le N}\delta_T(i)$. We introduce auxiliary variables $\psi_t(i)$ to store the state i maximizing Eq. (3.15). In detail we use the following procedure

1. Initialization for $t = 1$:

$$\begin{aligned}\delta_1(i) &:= \pi_i b_i(O_1), \quad \text{for all } i,\\ \psi_1(i) &:= 0.\end{aligned}$$

2. Induction for $t = 1, 2, \ldots, T-1$:

$$\begin{aligned}\delta_{t+1}(j) &= \max_{1\le i\le N}(\delta_t(i)a_{ij})b_j(O_{t+1}), \quad \text{for all } j,\\ \psi_{t+1}(j) &= \arg\max_{1\le i\le N}(\delta_t(i)a_{ij}).\end{aligned}$$

3. Computation of P^*:

$$\begin{aligned}P^* &= \max_{1\le i\le N}\delta_T(i),\\ q_T^* &= \arg\max_{1\le i\le N}\delta_T(i).\end{aligned}$$

4. Backtracking to construct Q^*:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \ldots, 1. \tag{3.16}$$

Analogously to the case for computing the likelihood of an observation sequence, we obtain a worst-case complexity of $O(TN^2)$. □

Of course, other definitions for a "most likely" sequence of states, corresponding to a given observation sequence, are conceivable. Churchill et al. [15] present a Bayesian approach using a Markov Chain Monte Carlo sampling procedure, to obtain state sequences which are most reliable with respect to perturbations of the input.

3.2.3 Baum-Welch Re-estimation

Training a model is the process of adjusting the parameters of a HMM λ. The objective function is to maximize the likelihood $P(O|\lambda)$ of observing a given observation sequence O. Due to the complexity of the likelihood landscape, no efficient, general, global optimization procedures are known.

The most widely used training procedure, called Baum-Welch re-estimation or the Baum-Welch algorithm [7, 8, 17], belongs to the class of algorithms dealing with statistical *missing value problems* or *missing data problems* [48]. *Expectation maximization* algorithms, or, for

short, EM algorithms [11, 55, 67], provide an effective and robust *local* optimization procedure for those problems.

In the typical setting of maximum-likelihood estimation, one tries to find parameters θ governing a distribution from which some data X is drawn, such that the likelihood $\mathcal{L}(\theta|X)$ is maximized. In missing value or, as they are sometimes called in the literature, *incomplete data* problems, θ now governs a distribution, which not only produces X, but also the missing values Y. The main idea behind EM is the insight that one can consider the so-called *complete-data likelihood* $\mathcal{L}(\theta|X, Y)$ and maximize it by iterating the following two steps:

1. **Expectation Computation:** Take the parameter estimate at iteration i, θ^i and X as constant. Compute the expectation of the complete-data log-likelihood and express this log-likelihood in terms of θ with respect to Y, which is a random variable governed by θ^i and X. This is also called the E-step.
2. **Maximization:** Choose θ such as to maximize the expectation. Set θ^{i+1} equal to θ. This is referred to as the M-step.

The EM algorithm will not decrease the likelihood, and thus converges at least to a local maximum [7, 8], which implies the same behavior for the Baum-Welch algorithm.

We will only formulate the Baum-Welch algorithm. A proof of correctness and local convergence can be found in [65]. In the case of HMMs, the missing values Y are the states in the state sequence Q leading to an observation sequence O, which corresponds to the variable X in the general formulation above. The expectation of the complete-data log-likelihood equals

$$\mathcal{Q}(\lambda, \lambda') = \sum_{Q} \log\left(P(O, Q|\lambda)P(O, Q|\lambda')\right), \tag{3.17}$$

which we will have to evaluate in the E-step. This yields [7] in the M-step the update rules

$$\bar{a}_{ij} := \frac{\text{expected number of transitions from } S_i \text{ to } S_j}{\text{expected number of transitions leaving } S_i}, \tag{3.18a}$$

(3.18b)

$$\bar{b}_{jm} := \frac{\text{expected number of observations } v_m \text{ in state } S_j}{\text{expected number of arrivals in state } S_j}, \text{ and} \tag{3.18c}$$

(3.18d)

$$\bar{\pi}_i := \text{probability of starting in state } S_i \text{ at time } t = 1. \tag{3.18e}$$

The expectation values can be efficiently computed with the help of additional variables $\gamma_t(i)$ and $\xi_t(i, j)$, where

$$\gamma_t(i) := P(q_t = i|O, \lambda), \text{ and} \tag{3.19}$$

$$\xi_t(i, j) := P(q_t = i, q_{t+1} = j|O, \lambda). \tag{3.20}$$

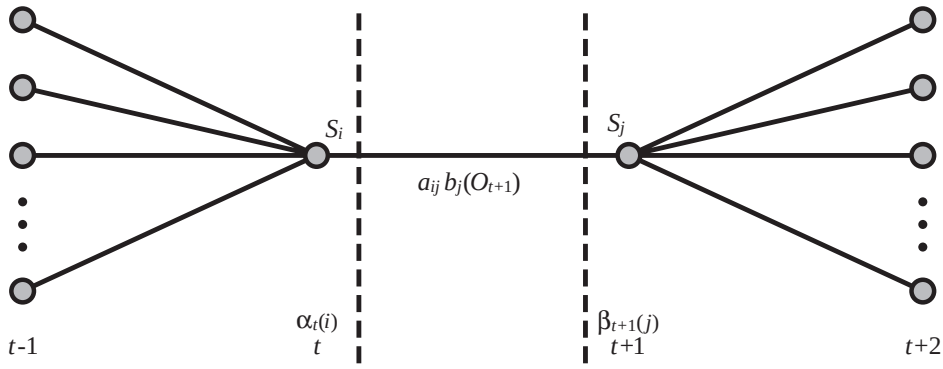

Figure 3.5: *Breakdown of events and respective probabilities making up* $\xi_t(i,j)$

We can express the additional variables using Eq. (3.14) in terms of the forward and backward variables introduced in Sec. 3.2.1

$$\gamma_t(i) = \frac{P(q_t = i, O|\lambda)}{P(O|\lambda)} \tag{3.21}$$

$$= \frac{\alpha_t(i)\beta_t(j)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}, \tag{3.22}$$

and, as illustrated by Fig. 3.5,

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}, \tag{3.23}$$

yielding the identity (using 3.12)

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j). \tag{3.24}$$

By summing over time, we arrive at the desired expectations, namely

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{expected number of } S_i\text{, and} \tag{3.25}$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{expected number of transitions from } S_i \text{ to } S_j. \tag{3.26}$$

We can now rewrite the re-estimation or update formulas 3.18:

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} = \frac{\sum_{t=1}^{T-1} \alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(i)}{\sum_{t=1}^{T-1} \alpha_t(i)\beta_t(j)}, \tag{3.27}$$

$$\bar{b}_{jm} = \frac{\sum_{t=1, O_t=v_m}^{T} \alpha_t(j)\beta_t(j)}{\sum_{t=1}^{T} \alpha_t(j)\beta_t(j)} \quad \text{and} \tag{3.28}$$

$$\bar{\pi}_i = \gamma_1(i) = \frac{\alpha_1(i)\beta_1(i)}{\sum_{j=1}^{N} \alpha_1(j)\beta_1(j)}. \tag{3.29}$$

Note, that the stochasticity constraints still hold for all re-estimated parameters. Also, as will become relevant in Sec. 3.4.1, the Baum-Welch algorithm preserves zero entries in A, B and π.

3.2.4 Implementation and Numerical Issues

Since the Baum-Welch algorithm converges to a stationary point, we can simply run it until the likelihood does not change anymore between iterations. More typically, we will stop as soon as the change is below some pre-specified small constant, or a maximal number of iterations has been performed. The exact regime used will be application-dependent.

As it is routine with numerical codes in statistics, one has to take precautions against numerical underflow errors, as, e.g., $P(O|\lambda) \to 0$ exponentially as the length of O goes to infinity. We will only compute log-likelihoods in practice, and also employ a scaling scheme for the algorithms introduced in the previous section, which ensures that at every time step all parameters are within the floating point range. A detailed development can be found in [65]. We will briefly introduce scaling to the computation of the forward and backward variables.

1. $t = 1$:

$$\begin{aligned} \tilde{\alpha}_1(i) &:= \alpha_1(i) = \pi_i b_i(O_1)\,, \qquad 1 \le i \le N\,, \\ c_1 &:= \frac{1}{\sum_{i=1}^{N} \tilde{\alpha}_1(i)}\,, \\ \hat{\alpha}_1(i) &:= c_1 \cdot \tilde{\alpha}_1(i)\,. \end{aligned} \tag{3.30}$$

2. $t = 2, \ldots, T$:

$$\begin{aligned} \tilde{\alpha}_t(i) &:= \left(\sum_{j=1}^{N} \hat{\alpha}_{t-1}(j) a_{ji} \right) b_i(O_t)\,, \quad 1 \le i \le N\,, \\ c_t &:= \frac{1}{\sum_{i=1}^{N} \tilde{\alpha}_t(i)}\,, \\ \hat{\alpha}_t(i) &:= c_t \cdot \tilde{\alpha}_t(i)\,. \end{aligned} \tag{3.31}$$

By induction, we can obtain

$$\hat{\alpha}_t(i) = \left(\prod_{\tau=1}^{t} c_\tau \right) \alpha_t(i)\,, \tag{3.32}$$

and thus, using 3.31,

$$\hat{\alpha}_t(i) = \frac{\alpha_t(i)}{\sum_{j=1}^{N} \alpha_t(j)}\,. \tag{3.33}$$

Remark 3.13 *For the conditional probability of being in state i given the observations and the model, the following equation holds:*

$$P(q_t = i | O_1 \cdots O_t, \lambda) = \hat{\alpha}_t(i). \tag{3.34}$$

The backward variables $\beta_t(i)$ will be multiplied with the same scaling factors as above, yielding

$$\begin{aligned}
\hat{\beta}_T(i) &:= \beta_T(i) = 1\,, \\
\tilde{\beta}_T(i) &:= c_T \hat{\beta}_T(i)\,, \\
\hat{\beta}_t(i) &:= \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \tilde{\beta}_{t+1}(j) = \Big(\prod_{\tau=t+1}^{T} c_\tau\Big) \beta_t(i)\,, \\
\tilde{\beta}_t(i) &:= c_t \hat{\beta}_t(i)\,. \qquad (3.35)
\end{aligned}$$

The re-estimation formulas, for example for $\bar{a}_{ij}$, can be rewritten with the scaled forward and backward variables. Observe, with $C_t := \prod_{\tau=1}^{t} c_\tau$ and $D_{t+1} := \prod_{\tau=t+1}^{T} c_\tau$, that $C_t D_{t+1} = C_T$ for all t, and thus, as $D_{T+1} := 1$,

$$\begin{aligned}
\bar{a}_{ij} &= \frac{\sum_{t=1}^{T-1} \alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i)\, \beta_t(i)} \\
&= \frac{\sum_{t=1}^{T-1} C_T\, \alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)}{\sum_{t=1}^{T-1} C_T\, \alpha_t(i)\, \beta_t(i)} \\
&= \frac{\sum_{t=1}^{T-1} C_t \alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, D_{t+2} \beta_{t+1}(j)\, c_{t+1}}{\sum_{t=1}^{T-1} C_t \alpha_t(i)\, D_{t+1} \beta_t(i)} \\
&= \frac{\sum_{t=1}^{T-1} \hat{\alpha}_t(i)\, a_{ij}\, b_j(O_{t+1})\, \hat{\beta}_{t+1}(j)\, c_{t+1}}{\sum_{t=1}^{T-1} \hat{\alpha}_t(i)\, \hat{\beta}_t(i)}\,. \qquad (3.36)
\end{aligned}$$

The remaining re-estimation formulas translate as well [65]. From the following equations,

$$\begin{aligned}
\Big(\prod_{\tau=1}^{T} c_\tau\Big) P(O|\lambda) &= \Big(\prod_{\tau=1}^{T} c_\tau\Big) \sum_{i=1}^{N} \alpha_T(i) \\
&= \sum_{i=1}^{N} \Big(\prod_{\tau=1}^{T} c_\tau\Big) \alpha_T(i) \\
&= \sum_{i=1}^{N} \hat{\alpha}_T(i) \\
&= 1\,, \qquad (3.37)
\end{aligned}$$

we obtain an expression for the log-likelihood, only making use of the scaled variables, and thus safely within a computer's floating-point range, namely

$$\log P(O|\lambda) = -\sum_{\tau=1}^{T} \log c_\tau\,. \qquad (3.38)$$

3.2.5 Extension to Multiple Observation Sequences

In many applications one has to deal with a set of observations; e.g., a family of protein sequences or a set of samples of the same word spoken. Generally, given are *K independent* observation sequences $O^{(1)}, O^{(2)}, \ldots, O^{(K)}$, where the $T^{(k)}$ denote their respective lengths, and the observations

within one sequence are denoted by $O^{(k)} = O_1^{(k)} O_2^{(k)} \cdots O_{T^{(k)}}^{(k)}$. The joint likelihood of the observed sequences is

$$P(O^{(1)}, O^{(2)}, \ldots, O^{(K)}|\lambda) = \prod_{k=1}^{K} P(O^{(k)}|\lambda). \tag{3.39}$$

Fortunately, we can adapt the Baum-Welch algorithm to the likelihood of Eq. (3.39). As in the case of a single observation sequence, the expected number of transitions from S_i to S_j for a fixed sequence $O^{(k)}$ equals $\sum_{t=1}^{T^{(k)}-1} \xi_t^{(k)}(i,j)$. By summation over k, we arrive at the expected number of transitions from S_i to S_j for *all* sequences. Similarly, we can adjust other parameters in a similar manner and obtain the required re-estimation formulas; e.g., for the transition probabilities this yields:

$$\bar{a}_{ij} = \frac{\sum_{k=1}^{K} \sum_{t=1}^{T^{(k)}-1} \xi_t^{(k)}(i,j)}{\sum_{k=1}^{K} \sum_{t=1}^{T^{(k)}-1} \gamma_t^{(k)}(i)}. \tag{3.40}$$

The proof of convergence can equally easily be transferred to multiple observation sequences, as can the scaling technique introduced in Sec. 3.2.4. As a matter of fact, the only difference are the additional summations over k in numerator and denominator.

Instead of simply using the k sequences in the training, we can also assign a positive weight w to each sequence. This is equivalent to adding w copies of the same sequence to the training data set.

3.2.6 Continuous Observations and further Extensions

While we only deal with strings, and thus with HMMs with discrete observations, the development of the theory carries through even if one considers continuous distributions or their mixtures as outputs [34, 65, 66]. Some precautions have to be taken to assure convergence of the Baum-Welch algorithm. In general, sufficient conditions on the probability density functions are ellipsoidal symmetry [50] or, much stronger, log concavity. In [34], this is extended to the case of finite mixtures of (multi-variate) Gaussian, or, more generally, elliptically symmetric, distributions. Note, that via a routine approximation [72] this allows an extension to arbitrary probability density functions.

Another area of research interest is using alternatives to the Markov chain in an HMM. Since especially the Markov assumption (cf. 3.2) is a limiting factor in some applications, one approach taken has been employing what amounts to an inhomogeneous Markov chain. The inhomogeneity can be either controlled, just as in time-inhomogeneous Markov chains, by the time of the observation, or by some external quantity, conditioned on observations [59]. In [41], HMMs are used for the analysis and simulation of economic time series data, which basically encodes cash flow. The sum of the observations up to time t was the controlling quantity in this case.

Also, the combination of Artificial Neural Networks and HMMs [5], extending the set of states to a continuous set [76], and the generalization of HMMs to — or embedding them in — the framework of *graphical models* [5] has been investigated.

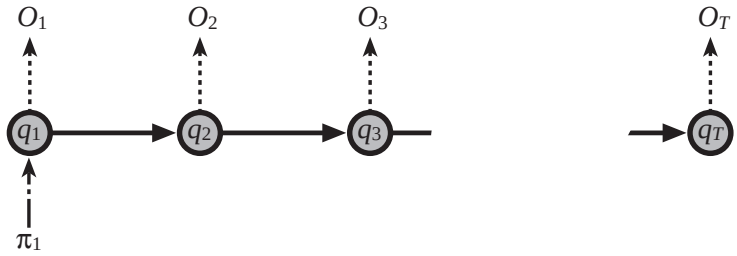

Figure 3.6: *The unconstrained maximum likelihood model for an observation sequence* $O = O_1 O_2 \cdots O_T$

3.3 An even more Fundamental Problem: Choosing HMM Topology

While the three problems described in 3.3 are indeed essential for working with HMMs, they do not cover a problem which we find even more important: How does one obtain an HMM in the first place? More specifically, how does one arrive at the right number of states, or the "correct" arrangement of connections between states given some training data?

To formalize the problem, let us define the topology of a model.

Definition 3.14 (Topology of an HMM) *Given an HMM* λ*. With topology of a model we refer to the set of states, and, most importantly, the allowed and forbidden transitions between the states of the underlying Markov chain; that is, the non-zero and zero entries, respectively, of the transition matrix.*

Alternatively, one can also include the allowed and forbidden emissions in each state in the definition of topology.

Definition 3.15 (Full topology of an HMM) *Given an HMM* λ*. With full topology of a model we also refer — in addition to the set of states, the set of emissions, and the allowed and forbidden transitions between the states of the underlying Markov chain — to the allowed and forbidden emissions in each state; that is, the non-zero and zero entries, respectively, of the transition* and *the emission matrix.*

The dilemma of choosing the "correct" topology is caused by the fact that finding the topologically unconstrained maximum likelihood model is trivial but, alas, completely useless (cf. [73]). For an observation sequence $O = O_1 O_2 \cdots O_T$, the maximum likelihood model, see also Fig. 3.6, is simply a path of length T, $Q = Q_1 Q_2 \cdots Q_T$. The transition probability between states Q_i and Q_{i+1} equals one and all the other transition probabilities equal zero. Each state Q_i emits symbol O_i with probability one.

This can be extended to several observation sequences, cf. Sec. 3.4.3. Nevertheless, such a model evincing absolutely no *generalization* is effectively an *exact string matcher*, and hence not a suitable tool for statistical pattern classification. Later on, we will describe ways to force generalization.

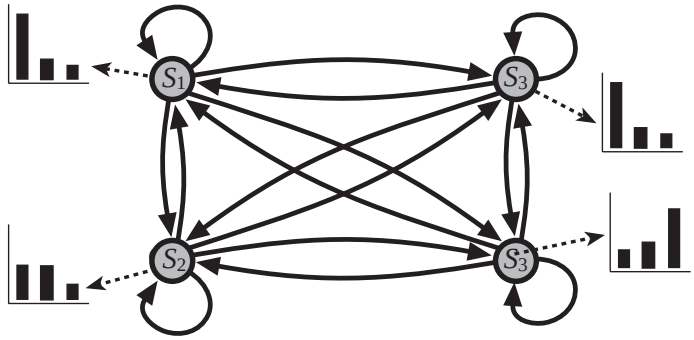

Figure 3.7: *A fully connected HMM*

One also finds the term architecture or structure for the concept of topology in the literature. Note, that neither the actual values of transition probabilities nor those of the emission probabilities matter, provided they be non-zero.

Lemma 3.16 *No further transitions are added to HMM topology by Baum-Welch training.*

Proof. Observe the presence of the a_{ij} factors in all the summands in the numerator of the re-estimation formula Eq. (3.18a), which forces the re-estimated variables $\overline{a}_{ij}$ to remain zero. □

Remark 3.17 *If all transition and emission probabilities are non-zero no transitions are removed by Baum-Welch training.*

3.3.1 Example Topologies

To give some intuition regarding differences in topology, we will introduce basic types of HMM topologies and discuss their respective features.

Definition 3.18 (Fully connected HMM) *We speak of a* fully connected HMM*, when the states are pairwise connected; i.e., when the underlying digraph is complete.*

That is the case, if the transition matrix has no zero entries, except possibly on the diagonal. Diagonal entries of the matrix correspond to *loops* or *self transitions*. There are typically no distinguished initial or final states. In the language of Markov chains, the underlying MC is *irreducible*, without *absorbing* states. Fig. 3.7 shows a fully connected HMM with four states and self transitions.

Definition 3.19 (Left-right model) *A HMM λ is called a* left-right model*, if the underlying directed graph is acyclic, except possibly loops, hence, supporting a* partial order *on the states.*

We can compute the partial order by sorting the graph topologically [69]. In such a left-right model, see Fig. 3.8, this yields one or several distinguished initial and final states. By convention there is only one initial state and one final state, which can always be achieved by introducing a special symbol for the end of an observation sequence and introducing *silent* states.

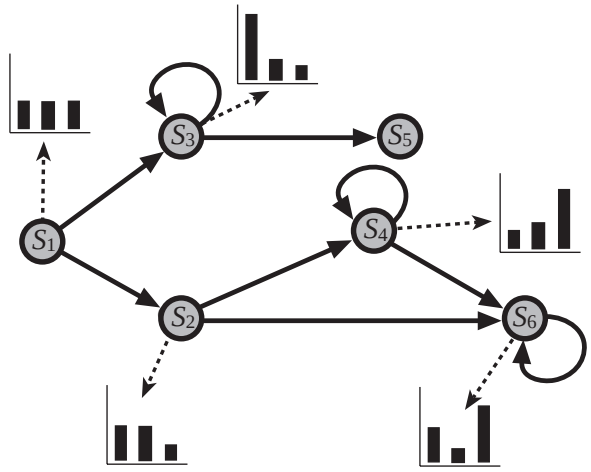

Figure 3.8: *A left-right HMM (cf. Def. 3.19): Note the partial order on the states*

Definition 3.20 (Silent state) *A state s which produces no output (or, equivalently, always the empty string ε) is called* silent.

Also, the initial distribution π puts all its weight on the distinguished initial state in the case of left-right models. For such a model the transition matrix has to be upper triangular. In the language of Markov chains, the final state is called *absorbing*; note, that in this case a loop on the absorbing state is required to ensure stochasticity of the transition matrix.

Definition 3.21 (Strict left-right model) *A left-right model is called* strict, *when there are no loops and there are only transitions going from a state of graph-theoretical distance d from the initial state to one of distance $d + 1$.*

An important aspect of left-right models are the limits to the length of the observation sequences which can be produced by the model.

Remark 3.22 *Note, that a left-right model without loops will only produce finite length observation sequences. More exactly, all observations will have length at least equal to the length of the shortest path from the initial to the final state and at most equal to the length of the longest path.*

A special case of left-right models are so called profile HMMs used widely in molecular biology applications.

Example 3.23 (Profile HMM) *Profile HMMs [19, pp. 100], see Fig. 3.9, which are used for various tasks in the context of protein sequence analysis, are examples of left-right models. All states in a profile HMM have bounded degree and, neglecting the loops in the insert and deletion states, they are strict left-right models.*

As we can see in Table 3.1, for a fixed number of states the chosen topology controls the number of parameters. This influences both the running time of the algorithms introduced in the previous sections and the amount of training data required.

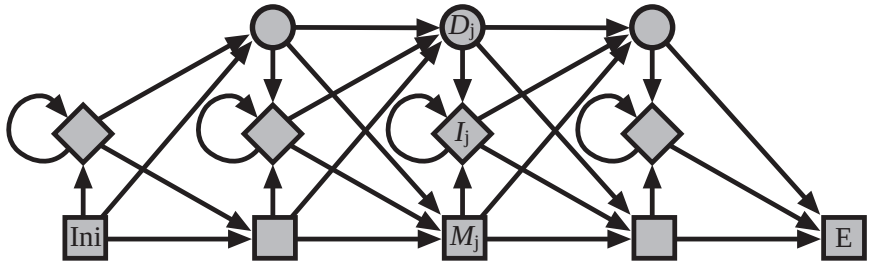

Figure 3.9: *Profile HMMs (cf. Def. 3.23), are widely used in the analysis of (protein) sequences. The square states are so-called match states, the diamonds are insert states and the circles are delete states corresponding to the classification of the observations they produce with respect to a consensus sequence. Delete states are silent.*

	Number of free parameters		
Model	transitions	emissions	initial
Fully connected	$(N-1)^2$	$(M-1)N$	$N-1$
Full left-right	$\frac{1}{2}(N-1)(N-2)$	$(M-1)N$	$N-1$
Profile HMM	$6\,L+4$	$2\,(M-1)\,L$	1

Table 3.1: *The number of free parameters are shown as a function of the number of states for different classes of HMMs. The remaining formal parameters are determined by the stochasticity constraints. In case of profile HMMs, the number of parameters are given in terms of L, the number of match states (cf. Fig. 3.9). Full left-right refers to a left-right model with the maximal number of transitions.*

3.3.2 Topology components

We can also look at the components making up the topology from a different angle, by classifying the states from a Markov-chain point of view.

Definition 3.24 *Given an HMM λ and an infinite sequence of states, $q_0q_1q_2\cdots$, define*

$$P^\star_{ij} := \sum_{n=1}^{\infty} P(q_n = j, q_k \neq j, \text{ for all } 0 < k < n | q_0 = i). \tag{3.41}$$

With the help of the preceding definition, we can classify the states of an HMM according to their probability of *recurring*, just as in the case of a Markov chain.

Definition 3.25 *Given an HMM λ. A state S_i of λ is called*

- transient *if* $P^\star_{ii} < 1$, *and*
- recurrent *if* $P^\star_{ii} = 1$.

Note, that if the HMM λ has end states, the definition makes only sense when we formally add a self-transition with unit probability to every end state. Then, in the case of left-right models,

all but the distinguished end states are transient. All the states in the fully connected model in Fig. 3.7 are recurrent, assuming proper choice of the transition probabilities. In the statistical or stochastic literature dealing with HMMs, ergodicity [59] of the source process and the model, implying recurrence of all states, is often required to allow for the use of analytical tools such as limit theorems.

In general, an HMM topology can consist of one or more components of two basic types, which can be characterized according to the types of states they contain, namely

- a transient component, and
- a recurrent component.

Examples of arbitrarily complex HMMs have been constructed [15].

3.4 Computing HMM Topology

Let us return to the issue of selecting an appropriate topology for a specific application. The search for systematic and automated procedures for selecting HMM topology is motivated by two key shortcomings:

- insufficient throughput of the mostly manual process used so far, and
- insufficient knowledge about the problem domain to support the decision-making process.

If the application itself naturally prescribes a topology, as, for instance, for speech recognition and for profile HMMs in bioinformatics, then experts can choose appropriate models. This is often combined with a classical model engineering approach of giving an application specific meaning to each individual feature of the model, and adjusting the model topology accordingly. These hand-crafted models are unquestionable highly desirable, if there is sufficient knowledge available on the problem domain. Still, an automated procedure capable of producing a number of models of varying sizes is highly desirable for the following reasons:

- Need for higher throughput. Even if models can be hand-crafted, the requirements of data mining applications often demand hundreds of distinct models, which makes manual intervention infeasible. This is often addressed by training individual models starting from the same base topology, which is not necessarily the most effective approach.
- Insufficient training data. As we can see in Table 3.1, the number of states also enters quadratically into the number of transition parameters and linearly into the number of emission parameters. Since the amount of training data is always too small in practice, a model too large can cause parameter estimates to be unreliable or counterproductive. Two mitigations are so called *parameter tying* [19] and Bayesian approaches with *prior distributions on observations* [71]. Nevertheless, a model with the "minimal" number of states necessary would also be advantageous from this point of view.
- Increased sensitivity. As we will see in Sec. 3.4.3, the number of states also influences sensitivity and specificity of models. Grossly oversimplifying we can state that, the smaller the number of states is, the more sensitive and the less specific the model will be. With an automated procedure more sensitive or more specific models could easily be created.

If the application knowledge is insufficient, that is, the "true" inner mechanism of an observed process is completely unknown or only partially understood, then manual inspection may not be a reliable procedure for choosing model topology. A rigorous framework, which requires explicit specification of a priori assumptions, would allow to select a maximum a posteriori model from training data alone.

Alternatively, learning families of models with varying numbers of states, corresponding to varying combinations of sensitivity and specificity, would provide a family of "fishing nets" of different mesh topology — not only mesh size — for such black-box applications. Sometimes application knowledge might well be *extracted* from those models, by observing which model feature causes changes in specificity.

Another aspect applying to both cases is a possible increase in computational efficiency. As stated in Sec. 3.2, the number of states enters quadratically in the complexity of the basic algorithms, and hence smaller models with a comparable sensitivity are highly advantageous. In the biological sciences, likelihood computations are done routinely for large numbers of HMMs and large sequence data bases, containing on the order of 100,000 sequences.

In the following we will discuss known algorithmic approaches to computing the topology of an HMM.

3.4.1 Baum-Welch with Thresholds

The Baum-Welch algorithm can be used to *reduce* the number of states of a model in the following way.

Algorithm 3.26 (Baum-Welch pruning) *Starting from some model λ and some training data set $\mathcal{O}$, we can iterate the following steps:*

1. *Perform Baum-Welch iterations; see 3.2.4 for details.*
2. *Prune the transition matrix by setting entries smaller than some $\varepsilon > 0$ to zero.*

Terminate after a certain number of iterations or if all transition probabilities are larger than ε after training.

There is no established theory guiding the choice of the initial model, of ε, or of the number of iterations. One has to monitor the likelihood at every iteration and use external criteria such as evaluation on a test data set to decide on the right model. Depending on the choice of λ, ε may have to be large in order to obtain any reduction in the number of states. Also, all transition probabilities might have similar large values, making the choice of ε appear arbitrary. Note, that this procedure may assign zero probability to sequences which were originally in the training set.

3.4.2 Model Surgery

An extension to using Baum-Welch with thresholds is the strategy known as *model surgery*. With model surgery, one tries not only to eliminate transitions which are rarely made, but to

- identify and eliminate states rarely visited, as states not present in the stochastic source, which is assumed to be some unknown "true" model, and
- identify states often visited, as possible unions of several states in the unknown "true" model.

The rationale behind model surgery is the assumption that states should contribute equally to the model. That is, the probabilities of visiting states should be of comparable value. Note, that this yields similarly reliable parameter estimates.

Algorithm 3.27 (Model Surgery) *Given a start model λ. Iterate the following steps:*

- *Perform Baum-Welch training, cf. 3.2.4.*
- *Consider the sum of transition probabilities to a given state. If the sum is very small, the state and all incident transitions are removed from the model. If the sum is large, split the state into two identical copies, with the same transitions and identical emission and transition probabilities. In some implementations the probabilities in the new states obtained by splitting are randomly perturbed, to accelerate divergence of the split states' parameters in the subsequent training.*

The algorithm terminates after a fixed number of iterations or if some sort of "equal use"[1] *of states has been achieved.*

Again, there is no established theory guiding a user in applying model surgery. The choice of the thresholds for deleting states and for the splitting of states is arbitrary, as are the modification and the termination criterion. Hence, model surgery has to be performed interactively with expert knowledge, based upon external factors such as performance evaluations, guiding the decisions being made.

3.4.3 Model Merging

Model merging is a Bayesian approach to learning both a model's topology and its parameters, developed by Stolcke and Omohundro [73]. The three major elements of model merging are [74]:

1. A method to construct an initial model from data.
2. A way to identify and merge sub-models.
3. An error measure to compare the goodness of various candidates for merging, and to limit the generalization.

One starts with the unconstrained maximum likelihood model for a given set of observation sequences $\mathcal{O} = \{O^{(1)}, O^{(2)}, \ldots, O^{(k)}\}$, see Fig. 3.10. As mentioned earlier, this model can only reproduce the training data set, each observation with probability $1/k$, and thus does *not* generalize at all.

In the merging step, individual states of the HMM are merged, or identified. A weighted average of their transition and emission probabilities yields the corresponding distributions for the merged state.

[1] Model surgery is an ad-hoc procedure without clearly formalized objectives.

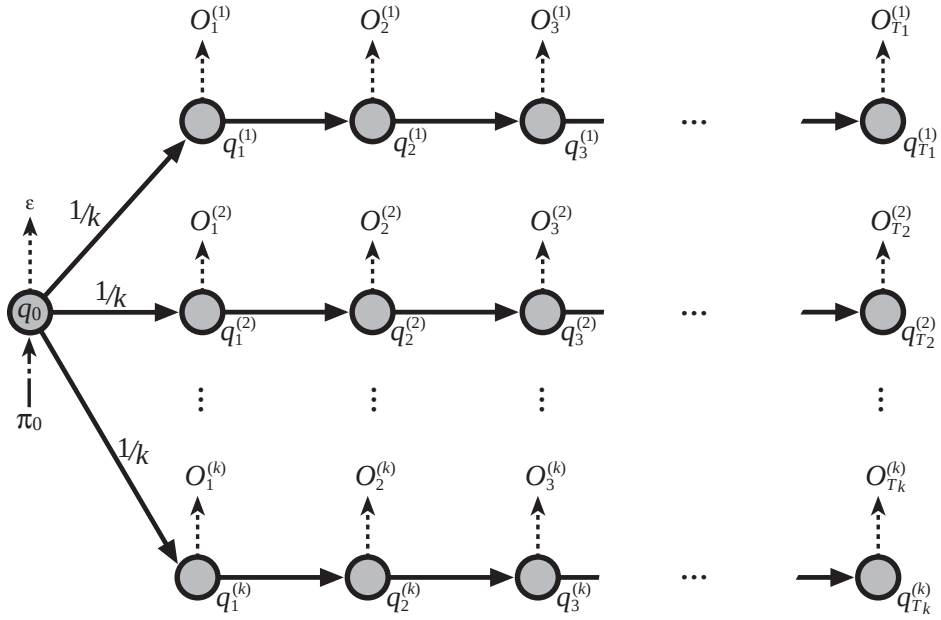

Figure 3.10: *The unconstrained maximum likelihood model for a set of observation sequences* $\{O^{(1)}, O^{(2)}, \ldots, O^{(k)}\}$ *is depicted. Here,* q_0 *is an initial state with the* silent *output* ε*. All transition probabilities equal one, except the ones from* q_0 *to the respective first states of each observation sequence.*

A Bayesian posterior is used to select models. One component of the posterior is a prior on the model's *topology*, which is the driving force of the generalization. The *posterior probability* can be computed as

$$P(\lambda|\mathcal{O}) = \frac{P(\lambda)P(\mathcal{O}|\lambda)}{P(\mathcal{O})}. \tag{3.42}$$

The *maximum a posteriori probability* (MAP) model λ_{MAP} is the one maximizing Eq. (3.42), where $P(\mathcal{O})$ is constant during model selection and $P(\mathcal{O}|\lambda)$ is the likelihood. The model prior $P(\lambda)$ factors into global structural, $P(\lambda_G)$, a per state structural, $P(\lambda_S^q|\lambda_G)$, and per state parameter priors, $P(\theta_\lambda^q|\lambda_S^q, \lambda_G)$, where θ_λ now denotes all the non-zero parameters in A, B and π, and θ_λ^q those which are pertinent to state q; i.e.,

$$P(\lambda) = P(\lambda_G) \prod_{q \in \mathcal{Q}} P(\lambda_S^q|\lambda_G) P(\theta_\lambda^q|\lambda_S^q, \lambda_G). \tag{3.43}$$

As it is routine for using priors in the context of HMMs [71], Dirichlet priors [19] were chosen as the per state parameter priors, yielding

$$P(\theta_\lambda^q|\lambda_S^q, \lambda_G) = \frac{1}{B(\alpha_t, \ldots, \alpha_t)} \prod_{r \in \text{ neighbors}(q)} a_{qr}^{\alpha_t - 1} \frac{1}{B(\alpha_e, \ldots, \alpha_e)} \prod_{e \in \text{ emissions}(q)} b_q(e)^{\alpha_e - 1}, \tag{3.44}$$

where α_t and α_e are the prior weights for transitions and emissions, respectively, and $B(\cdot, \ldots, \cdot)$ is the n-dimensional Beta function.

The structural prior per state allows control over the number of emissions in and transitions from each individual state. We can choose the average number of transitions, p_t, and the average number of emission symbols with non-zero emission probability, p_e, yielding the prior

$$P(\lambda_S^q|\lambda_G) = p_t^{|\text{neighbors}(q)|}(1-p_t)^{N-|\text{neighbors}(q)|}p_e^{|\text{symbols}(q)|}(1-p_e)^{N-|\text{symbols}(q)|}. \tag{3.45}$$

Finally, a global structural prior can bias the process towards choosing models with a smaller number of states; for example,

$$P(\lambda_G) \propto C^{-N}, \tag{3.46}$$

for some constant $C > 1$.

Algorithm 3.28 (Batch best-first model merging) *Given a data set, compute the unconstrained maximum likelihood model.*

1. *Determine a set of candidate merges, i.e., all pairs of states.*
2. *For each candidate merge compute the merged model and its posterior.*
3. *Select the MAP merge.*
4. *Terminate, if the merged model has a posterior* smaller *than the one in the previous iteration.*

The crucial step and governing factor is the computation of the set of candidate merges, which unfortunately will in general yield a complexity of $O(N^2)$, where initially N equals the total number of symbols in the training data set.

The empirical results Stolcke and Omohundro [73] have obtained are promising, but not very conclusive. Even though a great number of clever ideas and heuristics were used to speed up the most time-consuming parts of the algorithm, running time remains a critical issue. Also, having such fine-grained control over all parameter priors turned out to be problematic. All *prior parameters* have to be chosen sensibly, which is a problem different from choosing HMM topology, but not necessarily much easier in its most general formulation. Also, for the applications reported in [74], the authors had to resort to approximating likelihood computations by computing emission probabilities along Viterbi paths in order to keep the computational effort necessary within reasonable bounds. This is clearly unacceptable in general.

3.5 HMMs Producing and Accepting Strings

HMMs induce a probability distribution over $\Sigma^\star$, as they assign a likelihood $P(O|\lambda)$ for every finite observation sequence O. There is a distinguished subset of $\Sigma^\star$ defined by λ.

Definition 3.29 *Given an HMM λ, let $\mathcal{L} = \mathcal{L}(\lambda) \subset \Sigma^\star$ be the set of observation sequences $O = O_1 O_2 \cdots O_t$ such that*

1. *there exists a state sequence $Q = q_1 q_2 \cdots q_t$ with non-zero probability given λ, and that*
2. *the joint probability $P(O, Q|\lambda)$ is non-zero.*

We will call $\mathcal{L}$ the language associated with λ.

We can use λ to produce observations or strings. That is, we can *simulate* the stochastic process described by λ with the following Monte-Carlo algorithm:

Algorithm 3.30 *Given an HMM λ. Choose an initial state q_1 randomly according to the initial distribution π and iterate the following two steps.*

1. *Choose an emission value randomly, according to the distribution $\{B_{q_1 k}\}$.*
2. *Transition to a state, chosen randomly according to $\{A_{q_1 i}\}$.*

$\mathcal{L}(\lambda)$ is the set of all observation sequences λ can produce with rates according to their likelihood.

By analogy with non-stochastic finite state automata, we can also consider an HMM as an *acceptor* of strings. In order to do this sensibly, we need to augment the alphabet with a special *terminating symbol*, typically \$, and add one or more *final states* which only emit the terminating symbol, and have no out-going transitions. Note, that these states cannot be subjected to the stochasticity constraints. We can now try to find a state sequence Q for a given observation sequence O, ending in \$. Q will necessarily end in one of the final states. If at least one such path exists with non-zero probability, we say that the model *accepts* O.

3.6 Distance between HMMs

Especially when one tries to improve upon algorithms for learning or *inferring* HMMs, a distance function on the space of models becomes important. One way of assessing performance is to start with a known model λ, use this model to generate data, and then use that data to infer a model $\hat{\lambda}$. If the inference algorithm performs well, one would expect to see a decrease in distance between λ and $\hat{\lambda}$ as the inference progresses.

The task of defining a distance between HMMs is complicated by the fact that HMMs are non-unique. The parameterization is fixed, up to relabeling of the states or, more technically, a permutation group acting on the states. In case of a fully connected model, the group is the symmetric permutation group. In case of the profile HMM in Fig. 3.9, neglecting the different fixed outputs, it is a product of several S_3 groups. Each S_3 acts on the three states on the same level; i.e., the three states at the same graph-theoretic distance from the initial state. In terms of the transition and emission matrices this corresponds to permutations of the rows and columns. Obviously, this rules out straight-forward distance measures defined solely on the matrices, as entries in the same position of, e.g., the transition matrices, might correspond to completely different model parameters.

Also, a probability distribution over $\Sigma^\star$ does not necessarily fix an HMM nor its parameters, even if we ignore relabeling. This can easily be seen by considering multiple copies of a given HMM λ. We then add a silent initial state with transitions to each state in each copy, and set

the transition probabilities from the initial state to the equivalent state in all the copies equal to that state's original initial probability divided by the number of copies. This newly constructed family of models induces the same probability distribution over $\Sigma^\star$ as λ itself.

Whether a probability distribution fixes the parameters depends upon the origin of, or true statistical model behind, the distribution. If the stochastic source is a fair coin, we cannot do better than inferring an HMM with uniform output distributions and *arbitrary* transitions. If a Markov chain is the true source, we could recover that in the limit. In the former case, the transition matrix A should not contribute to the distance and in the latter case, the minimum distance over all relabelings between the respective transition and emission matrices might actually be used as a distance function.

In the following, we will introduce distance functions on HMMs, which dispatch quite differently with the aforementioned difficulties.

3.6.1 A Matrix Distance

Levinson et al. [49] introduced a distance function for discrete-observation HMMs by employing the Euclidean distance between the emission matrices, minimized over all permutations of the states.

Definition 3.31 (HMM matrix distance) *Given two HMMs, λ_1 and λ_2, let N be the number of states, M the number of emissions, and $b_{ij}^{(i)}$ the entries of the respective emission matrices of models λ_1 and λ_2. Then the matrix distance is defined as*

$$MD(\lambda_1, \lambda_2) := \min_p \sqrt{\frac{1}{MN} \sum_{j=1}^{N} \sum_{k=1}^{M} \left(b_{jk}^{(1)} - b_{p(j)k}^{(2)}\right)^2}, \tag{3.47}$$

where the minimization is over all permutations p of $\{1, \ldots, N\}$.

The permutation p minimizing Eq. (3.47) can be computed via minimum bipartite matching [69]. Clearly, there are shortcomings of this approach. Differences in transition and initial probabilities *do not* contribute to the distance, although they *do* contribute to the probability distributions over $\Sigma^\star$ induced by the two models.

3.6.2 A Probabilistic Distance Measure

Juang and Rabiner [37] defined a distance measure on pairs of HMMs, λ_0 and λ, both of which have the same number of states N and are ergodic (cf. Def. 2.11). They avoid the problem of identifying which states to compare by defining their distance on differences in the likelihood of observations.

They approximate Kullback-Liebler distance or relative entropy, see Def. 2.22, between probability distributions by defining an estimator based upon a finite sample of observations.

Definition 3.32 (Probabilistic distance) *Given two HMMs, λ_0 and λ, let O_{λ_0} be an observation sequence of length T produced by the model λ_0. Define*

$$PD(\lambda_0, \lambda) := \log P(O_{\lambda_0}|\lambda_0) - \log P(O_{\lambda_0}|\lambda). \tag{3.48}$$

The setup allows to establish a probability space [37] where an appropriate limit theorem holds, asserting convergence to the distance as the length of the observation sequence, T, goes to infinity, due to ergodicity.

The distance function is clearly not symmetric, but can be modified naturally by defining

Definition 3.33 (Symmetric probabilistic distance) *Given two HMMS, λ_0 and λ, define*

$$PD_s(\lambda_0, \lambda) := \frac{1}{2}\left(PD(\lambda_0, \lambda) + PD(\lambda, \lambda_0)\right). \tag{3.49}$$

The requirements of ergodicity and equal number of states arise from technicalities of the proof and can be relaxed. Hence, the distance function can also be used for transient HMMs. By replacing one observation O by a set of independent observation sequences with sum of lengths equal to T, one can use the distance also for HMMs with transient states.

Note, that if the observation sequence produced by one model *cannot* be produced by the other, this yields an infinite distance between the models, although they may differ only marginally.

3.6.3 A Co-emission based Distance

Another distance measure, recently introduced by Lyngsø et al. [52, 53], is obtained by considering so-called *co-emission probabilities*. Let λ_1 and λ_2 be two HMMs and define

$$C(\lambda_1, \lambda_2) := 2 \sum_{\mathcal{O} \in \Sigma^\star} P(\mathcal{O}|\lambda_1) P(\mathcal{O}|\lambda_2). \tag{3.50}$$

Note, that if one considers the probability distribution induced by a HMM as a vector in the infinite dimensional space spanned by all finite sequences, the co-emission probability can be written as

$$C(\lambda_1, \lambda_2) = <\lambda_1, \lambda_2> = |\lambda_1||\lambda_2| \cos\alpha, \tag{3.51}$$

where α is the angle between the vectors, and $|\lambda| = \sqrt{<\lambda, \lambda>}$ is the length of λ as a vector.

Then, one can make the following definitions:

Definition 3.34 *For two HMMs λ_1 and λ_2 define the two distances*

$$D_{angle}(\lambda_1, \lambda_2) := \arccos\left(\frac{C(\lambda_1, \lambda_2)}{\sqrt{C(\lambda_1, \lambda_1) C(\lambda_2, \lambda_2)}}\right), \tag{3.52}$$

and

$$D_{diff}(\lambda_1, \lambda_2) := \sqrt{C(\lambda_1, \lambda_1) + C(\lambda_2, \lambda_2) - 2C(\lambda_1, \lambda_2)}. \tag{3.53}$$

The disadvantages are on the one hand the $O(n^6)$ complexity for the computation of the distance in the general case, although for left-right models more efficient exact algorithms and fast approximation schemes are known [52]. On the other hand, there are problems in distinguishing between models, caused by the fact that the co-emission probability is not maximal for the same model, but for a model which assigns the same probability to all the sequences in $\mathcal{O}$ [52].

3.7 Alternative Representations

In the following we will give two alternative definitions for Hidden Markov Models. The first one is of a more historical importance, but nevertheless helpful in obtaining a deeper — and sometimes more suitable — understanding of HMMs. The second one will be used as an intermediate representation in the algorithm introduced in the following chapter.

3.7.1 Functions of Markov Chains

Recall Def. 2.12 of a Markov chain from chapter 2. If we have a realization of a Markov chain $\{X_t = x_t\}_{t\geq 0}$, we can consider functions applied to each x_t yielding another realization of a process $\{Y_t = y_t\}_{t\geq 0}$.

Definition 3.35 *A* function of a Markov chain *is a stochastic process $\{Y_t\}_{t\geq 0}$, which is defined by a Markov chain $\{X_t\}_{t\geq 0}$ and a function*

$$\begin{aligned} f: \quad \mathcal{D} &\longrightarrow \mathcal{D}' \\ x &\longmapsto y = f(x) \end{aligned}$$

where the random variables X_t take on values from $\mathcal{D}$ and the random variables Y_t values from $\mathcal{D}'$, by extending f to operate on $\{X_t\}_{t\geq 0}$ by

$$f(\{X_t\}_{t\geq 0}) := \{f(X_t)\}_{t\geq 0}. \tag{3.54}$$

Definition 3.36 *We speak of a* stochastic function of a Markov chain *in the setting of the previous definition, if f is a stochastic function. That is, if $\mathcal{D}'$ is a finite set, and for each $x \in \mathcal{D}$ there is a discrete probability distribution $\{P(Y = y|X = x)\}_{y\in\mathcal{D}'}$; evaluating $f(x)$ is understood as sampling from the corresponding conditional probability distribution.*

Quite clearly, this definition supports the following:

Proposition 3.37 *Any function of a Markov chain is a Hidden Markov Model.*

In case of a non-stochastic function, all the emission probability distributions are singular. If f is a bijection, then we have a Markov chain again.

3.7.2 HMMs as Mealy Machines

Instead of associating emissions with states, we can alternatively associate emissions with transitions. In the context of automata theory the former is called a *Moore machine* and the latter a *Mealy machine* [1]. In case of HMMs this means that, instead of the discrete emission probability distributions for each individual state we have — possibly multiple — edges between a pair of states s and t and each edge is labeled with an emission a and weighted with the *joint* probability of transitioning from s to t and emitting a, conditioned on state s.

Definition 3.38 (Mealy HMM) *Given a digraph G and an alphabet Σ. Each edge is labeled with a letter a from Σ and weighted with a function*

$$\begin{aligned} w: \quad & V(G) \times V(G) \times \Sigma \to [0,1] \\ & (s,t;a) \to w(s,t;a) = P(t,a|s). \end{aligned}$$

Note, in general G will have multiple edges between the same pair of vertices and also loops.

It is possible to switch freely between both representations.

Proposition 3.39 *Given a Mealy HMM. It is equivalent to an HMM with set of states $S = V(G)$, output alphabet Σ and transition and emission probabilities $P(t|s)$ and $P(a|s)$ respectively, where*

$$P(t,a|s) = P(t|s) \cdot P(a|t). \tag{3.55}$$

We can compute the probability of a transition from s to t in the HMM as the sum of the probabilities over transitions from s to t regardless of the symbol produced in N.

$$P(t|s) = \sum_{a \in \Sigma} P(t,a|s). \tag{3.56}$$

The probability of emitting a specific symbol is given by

$$P(a|s) = \sum_{t \in S} P(t,a|s). \tag{3.57}$$

Chapter 4

The Inference Algorithm

In this chapter, we develop an algorithm for inferring Hidden Markov Model topology. At first, we will motivate our algorithmic approach by interpreting the k-tails defined in Sec. 2.6 from the point of view of stochastic processes. Subsequently, we give an abstract definition of the algorithm, or, more correctly, of a class of inference algorithms. In the remainder of this chapter, we describe the peculiarities of the individual components and analyze the computational complexity.

In this chapter we will assume — unless explicitly stated otherwise — that the source of the data is a stationary and ergodic stochastic process.

4.1 k-Tails from a Stochastic Point of View

Assume we are given a realization of a stationary and ergodic process, represented as a prefix tree (cf. Def. 2.34) of a window set (cf. Def. 2.33). Consider a k-tail of some fixed vertex v_x (cf. Fig. 4.1). The counts $c(\cdot)$ of children of v_x divided by the count $c(v_x)$ are the relative frequencies of the prefix x followed by $o_1, \ldots, o_M$, where the o_i are the corresponding edge labels coming from some fixed alphabet Σ. This extends to the children of the children etc., such that we finally obtain a vector with relative frequencies for the leaves of the k-tail,

$$\left(\frac{c(v_{xs_1})}{c(v_x)}, \ldots, \frac{c(v_{xs_k})}{c(v_x)} \right), \tag{4.1}$$

where the s_i are all strings from Σ^k in lexicographical order, and the count of non-existing vertices is assumed to be zero.

What do v_x and the relative frequencies above correspond to, if we assume that the stochastic source is in fact an HMM? Let us consider what the distinguishing *observable* characteristics between states of an HMM are. On the one hand, there are the immediate emission probabilities, which constitute a sufficient, yet not a necessary criterion for two given states being distinct. In other words, it is not sufficient to consider the conditional probabilities of all observation sequences of length one to distinguish between states. On the other hand, one can look at conditional probabilities of all observation sequences of length two, three and so on. While equality

of probability distributions does not necessarily imply equality of states[1], it is an indicator strong enough for all practical purposes.

Let us return to the question at hand. The vertex v_x corresponds to some state of the HMM. With the Markov assumption, the relative frequencies are estimates of the probabilities of observing the strings s_i, conditioned on being in the state corresponding to v_x. That is, the vector in Eq. (4.1) is an estimate of the discrete probability distribution

$$\big(P(s_1|v_x), \ldots, P(s_k|v_x)\big) \,. \tag{4.2}$$

The main idea now is to use these probability estimates to fix a mapping from prefix tree vertices to states of the HMM. A one-to-one correspondence between vertices and states cannot be expected in general. Furthermore, the relative frequencies are only reliable estimates for the discrete probability distributions above, when a large amount of data is available. Hence, one has to resort to mapping sets — or clusters — of vertices with sufficiently "similar" relative frequencies to the same state of the HMM. Transitions between such clusters of prefix tree vertices are given by edges connecting prefix tree vertices they contain; emissions are given through edge labels. The probabilities for those joint emissions and transitions can be determined using the prefix tree vertex counts, yielding a Mealy HMM (cf. Def. 3.38). In the following we will discuss how to control the clustering procedure, as to obtain a model satisfying our ultimate goals with respect to pattern classification.

4.2 A Bayesian Prior driving Generalization

As noted in Sec. 3.4.3, precaution has to be taken that the model inferred is able to generalize from the training data. Similarly to the model-merging algorithm 3.28, we employ a prior distribution to control the generalization power of the inferred model. However, we do not require to specify a prior on the space of HMMs.

From the previous section it is clear that the choice of the clustering — i.e., the partition of prefix tree vertices — determines the size of the model, as the number of clusters in the prefix tree equals the number of states in the model inferred. The number of states, though, is one of the crucial factors contributing to generalization. A one-state HMM, for example, is the most general model one can infer from data, as it will assign the same probability to *any* sequence with the same *sequence composition* as the training sequence(s). That is, the relative frequencies of letters from the alphabet Σ are all that matters for the computation of the likelihood. As the number of states increases, more subtle statistical features of the training sequences can be learned, such as bi-gram frequencies [59].

By using a parameterized clustering algorithm, where an input parameter specifies the maximal permissible distance between elements of the same cluster as in single-link-clustering [22], or the maximal permissible distance to the center of a cluster, we can control the number of clusters and hence the number of states by modifying the parameter. Thus, we can specify a prior distribution on this parameter and select the MAP model according to the given prior and the likelihood assigned to the training sequences by the HMM corresponding to a given clustering.

[1]The obvious counterexamples employ variations of multiple states with identical emission probabilities. A

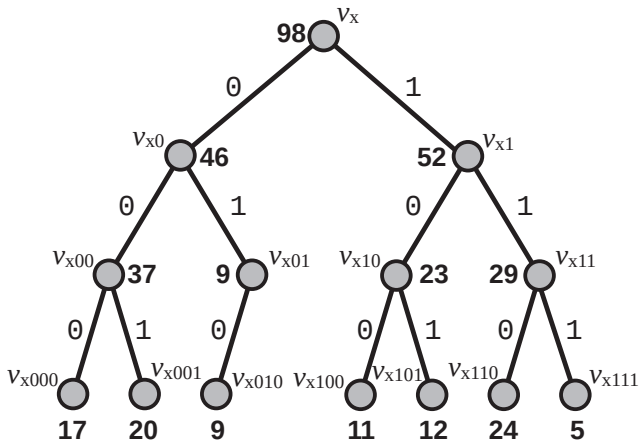

Figure 4.1: *In this* 3*-tail* $PT_x(\mathcal{S})$ *of a binary example prefix tree* $PT(\mathcal{S})$*, we can observe the relative frequencies induced by the counts of the leaves, indicated by bold numbers, and the count of the* 3*-tail's root* v_x*. Note, that missing counts in an incomplete k-tail are assumed to be zero. The relative frequencies* $c(v_{xs})/c(v_x)$*, where s is a string from* $\{0,1\}^3$*, are estimators for P(observing* $s|v_x$*). Thus, we can associate the discrete probability distribution* $\{P(s|v_x)\}_s$*, where the s are in lexicographical order, with* v_x*. In this example, the estimate for this distribution equals* $(0.18, 0.21, 0.09, 0, 0.11, 0.12, 0.25, 0.05)$*.*

4.3 The Topology Inference Algorithm: An Overview

We will introduce abstract features, which allows to characterize prefix tree vertices. Note, that we need not restrict ourselves to relative frequencies of strings as the objects by which we define a distance between prefix tree vertices. As a matter of fact, comparing *k*-tail *topology* is another feasible approach. The definition of the abstract *feature* encompasses all these possibilities in a general formulation.

Definition 4.1 (Feature) *Given a subset V of prefix tree vertices, which all possess a k-tail. A* feature *is a function*

$$\begin{aligned} f: \quad V &\longrightarrow \mathcal{D} \\ v_x &\longmapsto f(v_x) = f(PT_{x,k}) \,, \end{aligned}$$

where D is some domain, which allows for a distance measure or distance function. We will refer to D as feature space.

We will cluster prefix tree vertices based on the distance between the corresponding features. For the purpose of formulating the inference algorithm in generality we define the following

Algorithm 4.2 (Parameterized Clustering Algorithm) *Given a set V of prefix tree vertices and a distance function d between vertices of V. A* parameterized clustering *algorithm takes an argument* $\omega \geq 0$ *and computes a clustering,* $\mathcal{C}(\omega)$*. That is, it computes a partition of V into disjoint*

notion of minimality of an HMM might be useful in investigating this particular problem analytically.

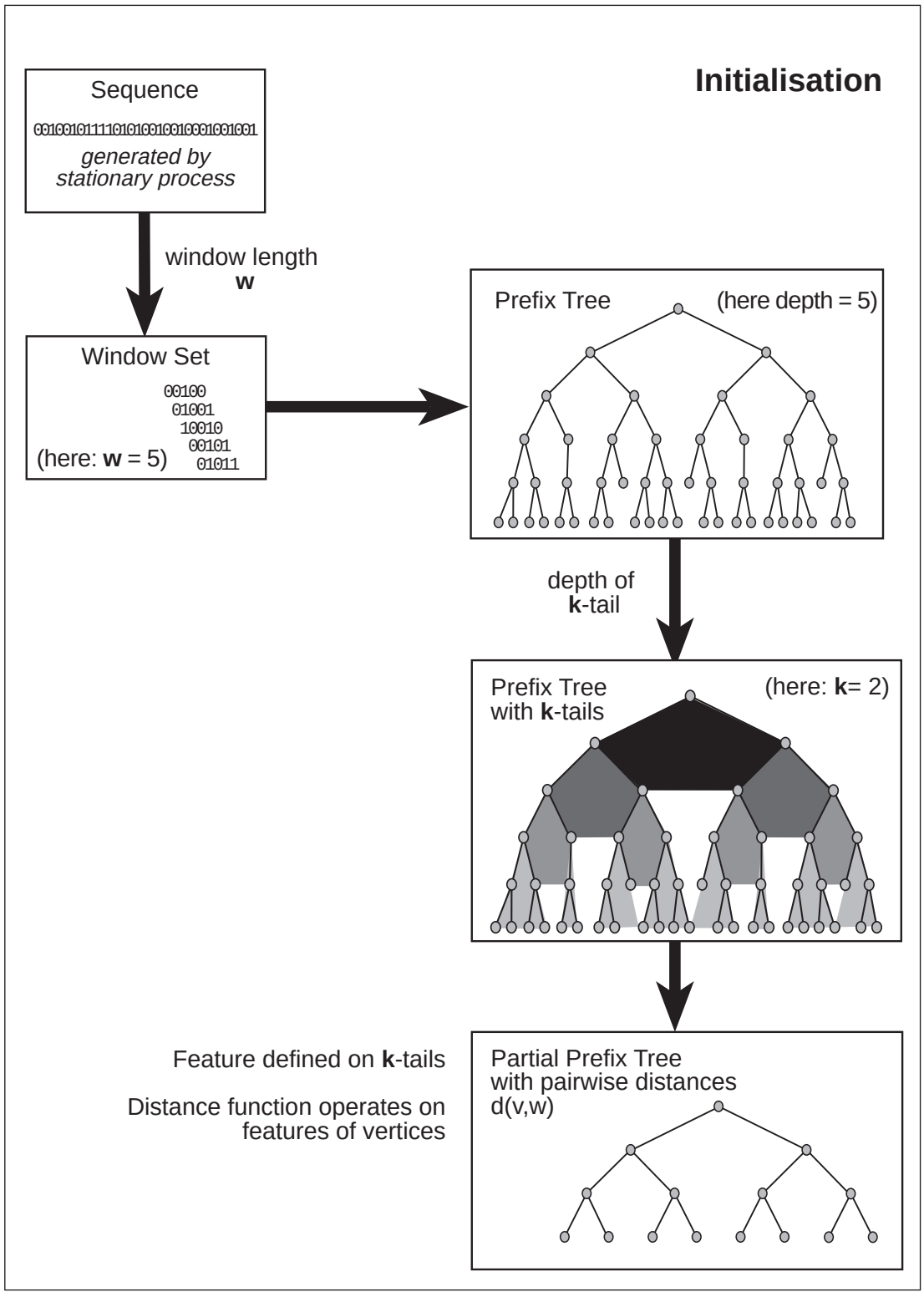

Figure 4.2: *Here we depict the data-flow in the inference algorithm during its initialization. Externally chosen parameters are displayed in bold face. The different shades of grey in the box containing the prefix tree with k-tails correspond to k-tails at different levels of the tree.*

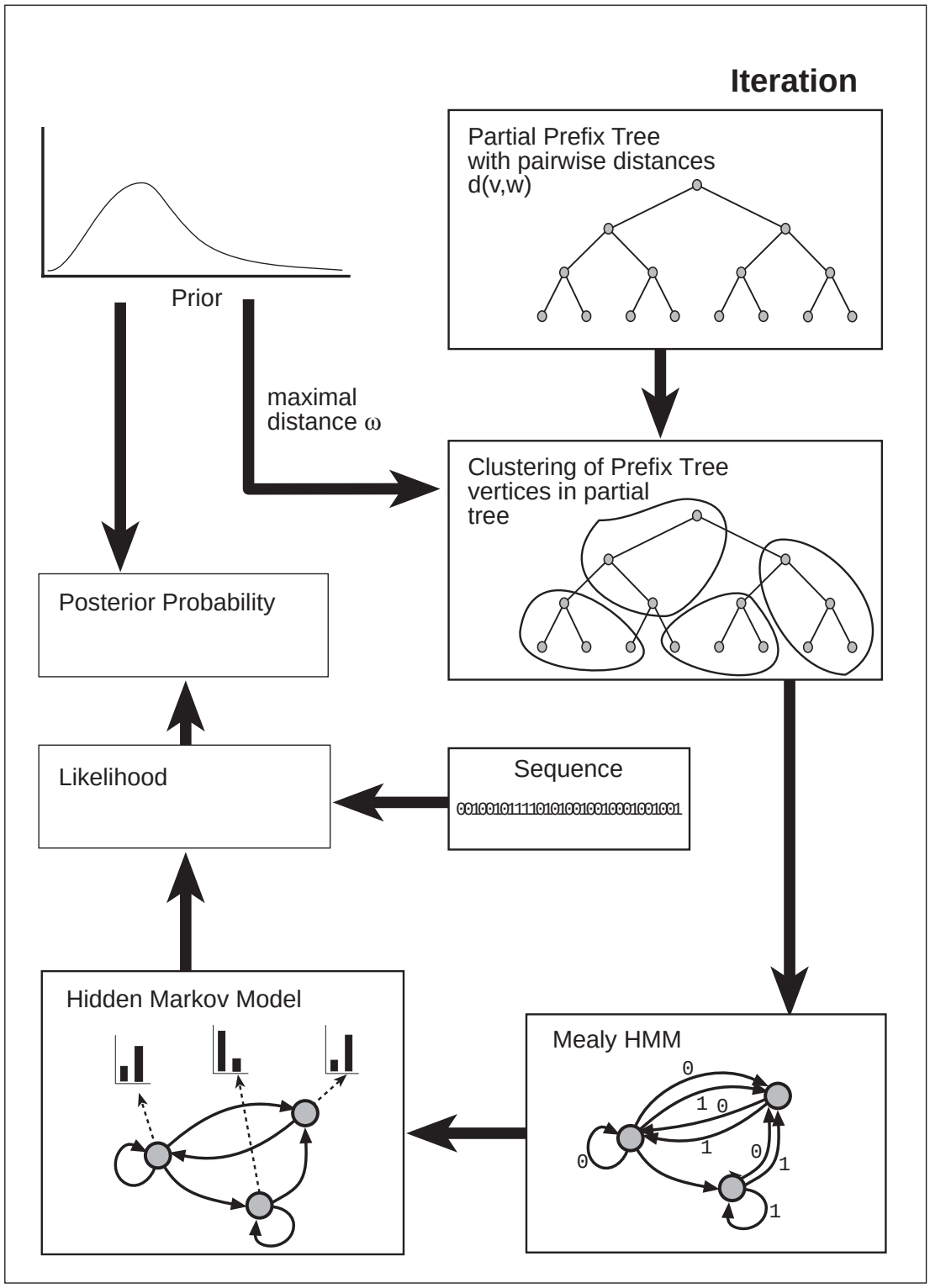

Figure 4.3: *The outline and the data-flow in the inference algorithm during its iteration phase are shown.*

subsets $\mathcal{C}(\omega) = \{C_1, C_2, \ldots, C_{k(\omega)}\}$. For any two clusterings $\mathcal{C}(\omega)$ and $\mathcal{C}(\bar{\omega})$, with $0 < \omega < \bar{\omega}$ and $|\mathcal{C}(\omega)| > |\mathcal{C}(\bar{\omega})|$, the following inclusion property holds. For every $C \in \mathcal{C}(\omega)$ there exists a $\bar{C} \in \mathcal{C}(\bar{\omega})$ such that $C \subset \bar{C}$. That is, we have a hierarchical clustering *[22], which we can control by using the parameter ω.*

Algorithm 4.3 (Inference Algorithm) *The inference algorithm computes the MAP-HMM defined by the likelihood of the training data and the prior given as input.*

Meta-Parameters:

- *A feature on k-tails*
- *A distance function on the feature space*
- *A parameterized clustering method*

Parameters:

- *The window length w*
- *The depth of the k-tails, k*
- *A prior distribution ϕ on ω, the parameter of the clustering, $\mathcal{C}(\omega)$*

Input:

- *A string s from $\Sigma^\star$ of length $T = |s|$*

Initialization: *(see Fig. 4.2)*

1. *Compute the window set $W_w(s)$.*
2. *Compute the prefix tree $PT := PT(W_w(s))$*
3. *For every vertex v_x in $PT_{\varepsilon,w-k}$, that is, all vertices up to depth $cd := w - k$ (cf. Fig. 4.4), compute the feature based on the k-tail $PT_{x,k}$.*
4. *For every unordered pair of prefix tree vertices in $PT_{\varepsilon,w-k}$, compute the pairwise distance based on the feature chosen.*

Iteration over ω*: (see Fig. 4.3)*
The iteration is performed with increasing ω. If $C(\omega)$ has been processed, the next value $\bar{\omega}$ is chosen such that $C(\omega) \neq C(\bar{\omega})$; i.e., only so-called critical values of the parameterized clustering are used in the iteration. The algorithm chooses the maximum a posteriori (MAP) HMM according to the prior ϕ as its output.

1. *Compute a clustering $\mathcal{C}(\omega)$ of vertices of the prefix tree $PT_{\varepsilon,w-k}$ with the chosen clustering method based upon the distance between their features.*
2. *Consider clusters as states of a Mealy HMM (cf. Def. 3.38). Edges between prefix tree vertices are added as edges between the corresponding states of the Mealy HMM. The joint probabilities of the edges in the HMM are estimated based on the counts $c(\cdot)$.*
3. *Use the marginals obtained from the Mealy HMM to compute emission and transition probabilities of a proper HMM, referred to as λ. Note, we use a uniform initial distribution.*

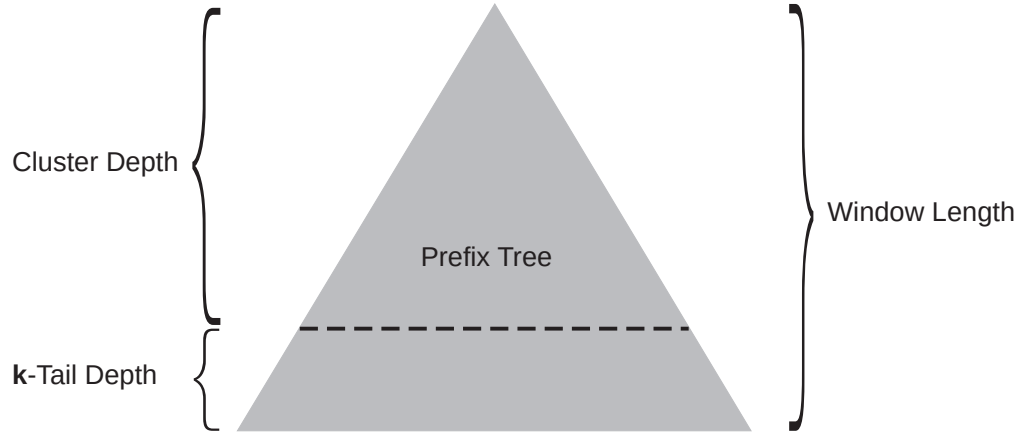

Figure 4.4: *The interpretation of the window length w, the parameter k and the cluster depth cd is visualized.*

4. *Compute the likelihood* $P(s|\lambda)$ *and, subsequently, the posterior probability*

$$P(\lambda|s) \propto P(s|\lambda)\phi(\omega). \tag{4.3}$$

In the following sections we will fill in the numerous blanks in the outline above. Note, that the performance of the algorithm depends crucially on the feature selected (see section 4.4.2) and the clustering algorithm used (see section 4.4.4).

4.4 The Topology Inference Algorithm: Details

In the following, we will discuss the details of the algorithmic framework introduced in the previous section. Let Σ be a fixed alphabet. The input for the algorithm is a string $s \in \Sigma^\star$, taken to be a realization of a stationary and ergodic stochastic source of length T. Parameters are the window length w, and k, the depth of the k-tails, both taken as constants in the algorithm and the subsequent investigations of theoretical complexity, and a prior ϕ on the parameter of the clustering. The cluster depth $cd := w - k$ is an additional constant, to simplify notation.

The iteration procedure is discussed together with the complexity of the algorithm in Sec. 4.5.

4.4.1 Computing a Window Set and Prefix Tree

If s is of length T, the window set $W_w(s)$, cf. Def. 2.33, contains $T - w + 1$ strings of length w for a window size of w. Clearly, as w is constant, the window set can be computed with complexity $O(T)$. From $W_w(s)$, we can compute the corresponding prefix tree, cf. Def. 2.34, $PT := PT(W_w(s))$. The tree PT has depth w and, in general, order $|\Sigma|^w$ and size $|\Sigma|^w - 1$. However, on *real* data, the tree will typically be sparse, as especially biological sequences are highly repetitive. This fact has been employed in a number of settings [27], to achieve a reduction of running-time in practice, even though the theoretical complexity remained unchanged.

From Lemma 2.35 we obtain that we can build the prefix tree in $O(T)$, including the vertex counts $c(\cdot)$ of the prefix frequencies corresponding to the vertices.

4.4.2 Features

A number of features are conceivable which allows one to distinguish between prefix tree vertices as representatives of HMM states. For example, one could use the topology of a k-tail, recall Def. 2.36, for this purpose.

We will restrict ourselves to the relative frequencies introduced earlier, cf. Eq. (4.1).

Definition 4.4 (k-tail frequencies) *Given a prefix tree PT and a vertex v_x. Let $PT_{x,k}$ be the k-tail rooted in v_x, and s_i the strings from Σ^k in lexicographical order. We define the* relative k-tail frequencies $r(x;k)$, *or $r(x)$ for short, if k is clear from the context, as*

$$r(x;k) := \left(\frac{c(v_{xs_1})}{c(v_x)}, \ldots, \frac{c(v_{xs_k})}{c(v_x)} \right). \tag{4.4}$$

Note, that the vector on the right has dimension $|\Sigma|^k$. If a vertex v_{xs_j} is not present in PT, we set the corresponding count $c(v_{xs_j})$ to zero.

For a fixed vertex v_x, the relative k-tail frequencies $r(x;k)$ can be computed with a number of operations proportional to $|\Sigma|^k$, for example using a truncated breadth-first-search on the tree to visit all descendants of v_x at distance k, and by setting the $|\Sigma|^k$ entries of $r(x;k)$ correspondingly. Since we have to compute the frequencies for all vertices in PT up to a depth cd, that is, for at most $|\Sigma|^{cd}$ vertices, the total number of operations required is proportional to $|\Sigma|^w$. Note, that the alphabet size as well as k, cd and w are constant, and hence are not relevant in the analysis of the computational complexity.

4.4.3 Robust Distance Functions

There are a number of approaches addressing the issue of defining a distance between discrete probability distributions, respectively relative frequencies obtained from sampling from an unknown discrete probability distribution. In the latter case, robustness with respects to artifacts caused by an insufficient amount of training data is crucial.

Divergence

Recall the definition of divergence, cf. Def. 2.23. If we choose to interpret the relative k-tail frequencies $r(x)$ and $r(y)$ for two given prefix tree vertices v_x and v_y as discrete probability distributions, we can simply use $\mathcal{D}(r(x), r(y))$ as a measure of their distance. Divergence will be zero, iff $r(x)$ and $r(y)$ are identical, and positive otherwise, so we can employ

$$\mathcal{D}(v_x, v_y) := \mathcal{D}(r(x), r(y)). \tag{4.5}$$

Regarding the robustness of this method with respect to sampling errors, there is one particular source for complications, namely counts which are exactly zero. As can be seen from Eq. (2.27), such a zero count causes the relative entropy to be unbounded, which can be deduced from considering an appropriate limit. Hence, the divergence would have to be taken as infinity. This is clearly undesirable, as on one hand it does not allow to distinguish between the case of

one or several of such zero relative frequencies. On the other hand equal differences between values have a drastically different impact. That is, a difference in counts of one does have a larger impact if it happens to be the difference between *not* observing and observing *once* than if it is the difference between two non-zero counts.

Possible methods of dealing with this are using pseudo-counts [19, pp. 115] or, alternatively, thresholding the individual terms in the cross-entropy computation.

Divergence with Pseudo-counts

We can simply add a count of one to each absolute count before dividing by the number of samples,

$$\overline{r}(x;k) := \left(\frac{c(v_{xs_1})+1}{c(v_x)+|\Sigma|^k}, \ldots, \frac{c(v_{xs_k})+1}{c(v_x)+|\Sigma|^k} \right). \tag{4.6}$$

This is known as "Laplace's rule" [19] in the literature.

A more involved but superior (cf. [19]) approach is to use the background distribution of the counted objects. In our case, we can use the expected numbers of k-strings over the alphabet Σ. That is, we can count how often the strings s_i from Σ^k appear in our input sequence s. We will denote the corresponding relative frequencies with

$$br(k) := \frac{1}{T-k+1} \left(\#s_1, \#s_2, \ldots, \#s_{|\Sigma|^k} \right). \tag{4.7}$$

Then, for the relative k-tail frequencies of some fixed prefix tree vertex v_x, we can use

$$\hat{r}(x;k)_i := \frac{c(v_{xs_i}) + A\, br(k)_i}{c(v_x)+A} \tag{4.8}$$

instead of $c(v_{xs_i})/c(v_x)$. The parameter $A > 0$ controls how much weight we put on the background distribution. Note, however, that $c(v_x)$ must be sufficiently large to assure that $br(k)_i > 0$. Unfortunately, for large k and large alphabet sizes this will be impossible in practice, as, roughly speaking, on the order of $|\Sigma|^k$ samples are required even for a uniform distribution over k-tuples to assure that the entries in $br(k)$ are non-zero. An investigation of the use of simpler background models, requiring less training data, might be fruitful.

We will denote with

$$\mathcal{D}^L(v_x, v_y) := \mathcal{D}^L(r(x), r(y)) := \mathcal{D}(\overline{r}(x), \overline{r}(y)) \tag{4.9}$$

the divergence for the "add one" pseudo-count relative k-tail frequencies and with

$$\mathcal{D}_A^{BR}(v_x, v_y) := \mathcal{D}_A^{BR}(r(x), r(y)) := \mathcal{D}(\hat{r}(x), \hat{r}(y)) \tag{4.10}$$

the divergence for the relative k-tail with the background distribution of weight A added.

Cross-entropy Cut-offs

Another simple possibility is to use a constant D, say of the order of a hundred at most, whenever the denominator in the $r(x)_i \log(r(x)_i/r(y)_i)$ term in the divergence computation is zero, instead of the correct value infinity. This obviously does not make mathematical sense. However, the fact that the $r(y)_i$ are rational numbers, with $c(x)$ as denominator and $c(xs_i)$ as numerator, implies that we can choose a value D, such that

$$r(x)_i \log\left(r(x)_i/r(y)_i\right) < D, \tag{4.11}$$

as the $r(y)_i > 1/max_x\{c(x)\}$. Let $I := [1, |\Sigma|^k]$ and $I_x^+ := \{i \in I \mid r(x)_i > 0\}$, I_y^+ analogously, then we can define

$$\mathcal{D}^D(v_x, v_y) := \mathcal{D}^D(r(x), r(y)) := \sum_{i \in I_y^+} r(x)_i \log \frac{r(x)_i}{r(y)_i} + \sum_{i \in I_x^+} r(y)_i \log \frac{r(y)_i}{r(x)_i} + D \times \left(|I \setminus I_y^+| + |I \setminus I_x^+|\right). \tag{4.12}$$

As long as none of the $r(x)_i$ and $r(y)_i$ are zero, we obtain the same value for the divergence as from the exact computation. If there are such zero values, we can still distinguish between one or several entries being zero, due to the small D. This yields a more informative divergence value in that case.

A somewhat similar approach has been used to define a distance between so-called expression profiles while clustering the results of DNA chip experiments [30]. There, two vectors x and y were discretized element-wise, and the following contingency variables were considered: n^{xy}, the number of indices with corresponding non-zero entries in both vectors, n_i^x and n_j^y, the number of times among those indices that the entries in vector x, respectively y, fell into discretization interval i, respective j, and finally n_{ij}^{xy}, the corresponding joint contingency table. These observed frequencies were then used to compute the mutual information between x and y for those indices where x_i and y_i are both non-zero, or where "signal is present in both vectors" [30]. Analogously, we can restrict the divergence computation to the set of indices

$$I^+ := \{i \in I \mid r(x)_i > 0, r(y)_i > 0\}, \tag{4.13}$$

yielding

$$\mathcal{D}^+(v_x, v_y) := \mathcal{D}^+(r(x), r(y)) := \sum_{i \in I^+} r(x)_i \log \frac{r(x)_i}{r(y)_i} + r(y)_i \log \frac{r(y)_i}{r(x)_i}. \tag{4.14}$$

Note, that using $\mathcal{D}^+$ is only a reasonable choice if $|I^+|$ is large on average, as $\mathcal{D}^+$ is zero when I^+ is empty.

Complexity

The computation of the pairwise distances is of quadratic complexity in the number of prefix-tree vertices up to and including depth cd, as an individual distance computation requires on the order of $|\Sigma|^k$ operations, which is a constant within the algorithm.

4.4.4 Clustering

In the following, we will discuss our choice of suitable clustering algorithms to use in the inference algorithm. Note, that this is not an exhaustive list. Also, there is an improvement in the performance of the algorithm to be gained from a different choice of clustering algorithm (cf. Sec. 5.4). We will first discuss the single-link clustering algorithm, which is an attractive choice due to its simplicity and computational efficiency.

Definition 4.5 (Threshold graph) *Given a complete graph G, where the weight $w((u,v))$ of an edge (u,v) corresponds to the distance between incident vertices, u and v. For a given $\tau > 0$, let $G(\tau)$ be the graph obtained from G by removing all edges (u,v) of weight $w((u,v)) > \tau$. $G(\tau)$ is called a* threshold graph.

Algorithm 4.6 (Single-link clustering [22]) *Given a threshold graph $G(\tau)$. Compute the connected components $\mathcal{C} = \mathcal{C}(\tau) = \{C_1, C_2, \ldots, C_l\}$ in $G(\tau)$, and return those connected components as the clustering.*

Lemma 4.7 *The single-link-clustering algorithm is an parameterized clustering algorithm, according to Def. 4.2.*

Proof. This follows immediately from the following observation. Increasing the threshold can be thought of as subsequently adding edges with increasing weight to the graph. Edges added are either contained in one connected component or join two distinct connected components. In either case, the relevant inclusion property is maintained. □

Subsequently, we will also need the following definition.

Definition 4.8 (Cluster weight) *Given a cluster C of prefix tree vertices, we call*

$$w(C) = \sum_{v_x \in C} c(v_x)$$

the weight of the cluster.

The following algorithm is a parameterized clustering algorithm by definition.

Algorithm 4.9 (Weighted average hierarchical clustering) *Define an initial clustering $\mathcal{C}(0)$ with $|\mathcal{C}(0)| = |PT_{\varepsilon,cd}|$ clusters, each of which contains an individual vertex. Iterate the following procedure: For each cluster compute a cluster representative as the weighted average*

$$r(C) := \sum_{v_x \in C} \frac{c(v_x)}{w(C)} \cdot r(x). \tag{4.15}$$

Compute the pairwise distances between clusters based on the representatives, and merge those clusters at distance of less than ω.

Note, that we now have two graphs on the same vertex set, namely the prefix tree and the threshold graph. In the following we will refer to the single-link clustering algorithm as *SLC* and to the weighted average hierarchical as *WAH*.

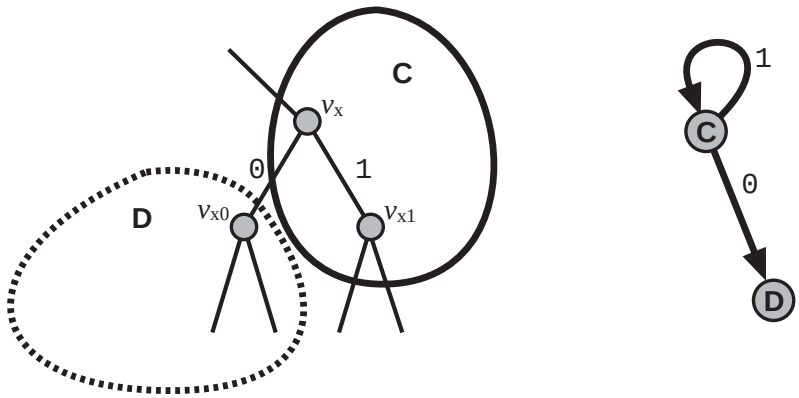

Figure 4.5: *Here we depict the connection between edges in the Mealy HMM and the edges between corresponding clusters in the prefix tree.*

4.4.5 Computing a Mealy HMM

Given the partial prefix tree and a clustering $\mathcal{C}$ of its vertices, we will show how to obtain a Mealy HMM, cf. Def. 3.38. Clusters in the prefix tree PT correspond to states of the HMM. That is, for every cluster $C_i \in \mathcal{C}$ of prefix tree vertices, we add a state S_i to the HMM. We add a transition from state S_i to state S_j emitting the symbol $a \in \Sigma$, if there are two vertices v_x and v_y in C_i and C_j, respectively, which are joined by an edge labeled with a in the prefix-tree; see Fig. 4.5.

Note, that such an edge exists if the following prefix relation between the prefixes x and y holds. If either $x = ya$ or $y = xa$, then there is an edge labeled with a joining v_x and v_y in the prefix tree. If C_i and C_j are the clusters containing x and y, respectively, we add the edge $(S_i, S_j; a)$ to the Mealy HMM. Clearly, there can — and usually will be — more than one edge in the prefix-tree corresponding to the same edge in the Mealy HMM. The probability associated with the edge $(S_i, S_j; a)$ will thus be defined as

$$P(S_j, a|S_i) := \sum_{v_x \in C_i, v_{xa} \in C_j} \frac{c(v_x)}{w(C_i)} \frac{c(v_{xa})}{c(v_x)} = \sum_{v_x \in C_i, v_{xa} \in C_j} \frac{c(v_{xa})}{w(C_i)}. \tag{4.16}$$

Lemma 4.10 *If we define the edge probabilities in the Mealy HMM obtained from the clustering as in Eq. (4.16), the appropriate stochasticity constraints hold.*

Proof. We have to show that the sum of probabilities over the outgoing edges is one; i.e., for

all C_i,

$$\sum_{S_j,a} P(S_j, a|S_i) = 1.$$

From the definition we have

$$\sum_{S_j,a} P(S_j, a|S_i) = \sum_{S_j,a} \sum_{v_x \in C_i, v_{xa} \in C_j} \frac{c(v_{xa})}{w(C_i)} = \frac{1}{w(C_i)} \sum_{S_j,a} \sum_{v_x \in C_i, v_{xa} \in C_j} c(v_{xa}) = \frac{1}{w(C_i)} \sum_{v_x \in C_i} c(v_x),$$

which proves the claim. □

4.4.6 Obtaining an HMM

As we have seen in proposition 3.39 we can, making use of the Markov assumption and the assumption that observations are independent of anything but the state, immediately obtain an HMM proper. We do assume an uniform initial distribution. In the last step of the algorithm, we use the forward variables, cf. Def. 3.6, to compute the likelihood of the input string s and combine it with the prior probability of the clustering parameter ω, $\phi(\omega)$, yielding the posterior probability $P(\lambda|s, \omega)$.

4.5 Computational Complexity of the Algorithm

The only input of the algorithm for which an investigation of the theoretical complexity makes sense is the input string s. Its length T can be arbitrarily large. A specific application has a fixed alphabet, the window length and tail depth are also fixed, when it is assured that the properties of the stochastic source can be captured.

The algorithm can be initialized with complexity $O(T)$, as shown in Sec. 4.4.1-4.4.3. Note, however, that the practical running time for the initialization is dominated by the constant, as w, k and $|\Sigma|$ are constant, term $|\Sigma|^{2cd}$.

We will analyze the iteration phase using SLC as the clustering method. As discussed in Sec. 4.4.3, the various distance functions based on the relative k-string frequencies as features do not make a difference, as the computational effort is constant in T.

When using SLC, the iteration over the clustering parameter ω can be performed efficiently, by sorting the threshold graph edges by increasing weight, corresponding to increasing distance and subsequently, starting from a set of isolated vertices, adding them to the graph in that order, merging two clusters, when they are joined by an edge. Whenever adding an edge changes the resulting clustering, we call that edge *critical* and subsequently re-compute the resulting Mealy HMM. Essentially, this is equivalent to computing a minimal spanning tree with Kruskal's algorithm [69]. We can compute such a minimal spanning tree, given a graph of size e, in $O(e \log(e))$ steps. Note, however, that there can be only $n - 1$ critical points, where n is the order of the threshold graph. The threshold graph has at most size $(|\Sigma|^{cd}(|\Sigma|^{cd} - 1))/2$ and order $|\Sigma|^{cd}$. The clustering can be performed in constant time, as it does not depend on T. Similarly, once we do

have a clustering, we can compute the Mealy HMM in constant time with respect to T. The likelihood computation, cf. Sec. 3.2.1, can be performed with $O(T N^2)$ steps, where N is the number of states in the HMM, which is bounded by $|\Sigma|^{cd}$.

The preceding proves the following theorem

Theorem 4.11 *For any of the distance measures in Sec. 4.4.3, the inference algorithm using SLC can compute a MAP model according to the parameters w, k and the prior ϕ with complexity $O(T)$.*

Note, that the theoretical complexity is misleading in practice, since the *constant* $|\Sigma|^{2cd}$ term clearly dominates the running time.

4.6 Implementation

We have implemented an HMM class and virtually all algorithms from Chap. 3 as well as the inference algorithm described in this chapter in the programming language Python [64]. For reasons of efficiency, the implementation made use of an external library for numerical analysis. Numpy [60] is a Python package (library module), which supplies an interface to the native Basic Linear Algebra System (BLAS) of the computing platform from within Python. The only overhead occurring in operations such as vector and matrix additions is for the calling overhead involved in mapping a Python operator to a library routine. The performance of the actual computation is as good as in the case of direct use of the BLAS-library from compiled languages such as C or Fortran. Nevertheless, this only applies to the likelihood computations etc., under the general disclaimer that a matrix based implementation of HMMs is inefficient, since it cannot make use of the typical sparseness. Most of computational effort goes into manipulating Python data structures.

The package was used and tested with Python version 2.0, compiled with the GNU ECGS, version 2.91.66 and release 1.1.2, on a Sun Enterprise 450 respectively Enterprise 4500 under Solaris 7, and with Python 2.0, compiled using Compaq's cxx compiler, version 6.20, on a Compaq ES40 running Tru64 Unix V5.1. In the former case the generic BLAS supplied with the Numpy package, version 17.3.0, was used, in the latter Compaq's CXML, version 4.1.0 [16].

4.7 Choosing Window Length and Tail Depth

The two parameters w and k limit the "horizon" of our method in two subtly different ways. Recall, that k is the depth of the tail we use to identify hidden states via the distribution of k-strings associated with them. The choice of window length and the choice of k define the effective "memory". We will use this term in an informal manner, for lack of a statistically sound quantitative concept encompassing the different aspects of what constitutes memory. That is, when visiting some state S_i always results in visiting state S_j some m time steps later, we will not be able to recognize it, if m is larger than $w - k$, i.e., the cluster depth cd. While this is a limitation from a theoretical point of view, this is not highly relevant in practice, if the long-range

interactions are infrequent. In that case, and for a restricted amount of training data, these interactions might not be present at all or drown in the noise floor induced by sampling artifacts due to insufficient training data.

The following aspects are relevant with regard to a choice of k and w.

- Amount of training data: For a fixed training data set, the counts in the leaves of the prefix tree will become more unreliable as we increase w. A reasonable choice of w should assure that the counts of prefix tree leaves are large enough to obtain reliable estimates of the relative k-tail frequencies even for vertices at level cd. Note, that this depends heavily on the data set. A uniform distribution — in the sense of a distribution over strings from $\Sigma^\star$ — produces a complete prefix tree, supporting a shallower prefix tree with reliable counts compared to a source which is a mixture of singular distributions.
- Range of significant correlations: Another area of investigation are correlations between observation symbols at increasing distances. If there are strong long-range correlations present, w and k should be chosen accordingly.
- "Memory" of the stochastic source: Sometimes, it will be possible to argue about memory length based upon knowledge from the application's domain, even when the specific mechanisms of the source are unknown.

Following the guidelines above, we will discuss our decisions for choosing w and k in Chap. 5, where we evaluate the performance on data sets.

4.8 Choosing a Prior

One interpretation of the clustering parameter ω is that of the threshold with respect to the distance between features of prefix tree vertices, which divides non-identical from identical states. It constitutes the maximal permissible distance, which we will attribute to pure chance, fluctuations of the stochastic source, or artifacts caused by unreliable estimates due to an insufficient amount of training data. Alternatively, it is the minimal distance, which convinces us that the prefix tree vertices correspond indeed to distinct states of the HMM.

The numerical range over which we will define a prior depends on the distance function used. Also, one should — as the number of states does not depend linearly on the clustering parameter — investigate the particular relation between ω and the number of states of the HMM. Highly informative priors certainly need careful adjustment. Broad priors, with more weight towards larger values of ω will drive the inference process "gently" towards smaller models and towards higher degrees of generalization, and are a sensible choice in general. We will elude to our choices in Chap. 5.

An alternative, which seems to be desirable from a purely practical point of view, is to define a prior directly on the number of states. Note, that our algorithmic framework easily and readily affords this change in the "user interface", and that we have used such a prior in the evaluation of the algorithm.

Chapter 5

Evaluation

In this chapter, we will evaluate the performance of the inference algorithm introduced in Chap. 4. The evaluation will be done from two different points of view, investigating two distinct aspects of performance evaluation.

There are two limiting factors in statistical inference from *real* data. On the one hand, the inference process might be inherently difficult or even infeasible for the given problem instance. E.g., in the case of HMMs, due to the complexity of the likelihood landscape, only *local* optimization algorithms are known. Therefore, even when we want to infer parameters starting from a prescribed topology, optimal training is impossible in general. On the other hand, the true process behind our stochastic source might be just impossible to describe with the class of models used.

From an engineering perspective, an evaluation of the overall performance on an annotated data set, typically based on cross-validation [23], is sufficient. This will measure the influence of both possible causes for errors mentioned above simultaneously. To allow the detection and quantification of possible inherent limitations in the inference process, we chose to test our algorithms on artificial data.

5.1 Artificial Data

One natural mode of testing the inference algorithm on artificial data can be formalized as follows:

Problem 5.1 (HMM recovery problem) *Let there be an HMM λ, the stochastic source, and a distance function, say d, between HMMs. If λ is ergodic, produce one observation sequence O of length T, otherwise a set of observation sequences $\mathcal{O}$, whose sum of respective sequence lengths equals T. Use this finite sample of the probability distribution over $\Sigma^\star$ induced by λ as an input to infer an HMM λ'. Evaluate $d(\lambda, \lambda')$.*

By varying T, one can obtain an estimate on the amount of training data needed for a particular type of source model, which is helpful in establishing guidelines for working on real data. Note, that it would be naïve to expect recovery of the *exact* parameters of the source model, as there will usually be a non-negligible number of models consistent with the finite sample data set.

Hence, we have to investigate the distance between source and inferred model. Beforehand, we will introduce a measure of distinguishability relevant with respect to the choice of the stochastic source used in the evaluation.

Another question which supports the importance of the measure of distinguishability, $\mathcal{MD}(\lambda)$, which we will define below, is its resolution of the amount of training data needed to assure reliable parameter estimates for a given model topology.

Remark 5.2 *Given an HMM λ. If $\mathcal{MD}(\lambda)$ is maximal, then bounds on the amount of training data needed can be obtained from the theory of Markov Chains [59]. If, on the other hand, $\mathcal{MD}(\lambda) = 0$, then the parameters of the underlying Markov Chain* cannot *be inferred at all. That is, one cannot obtain any confidence interval on the parameters, even with* infinite *amounts of training data.*

Given these two extremes with regard to reliability of parameter estimates and amounts of training data required, further investigations based on our measure of distinguishability seem a very worth-while task, even if beyond the scope of this thesis.

5.2 A Measure of Distinguishability

Inference of HMMs and their topology even from artificial data originating *from an HMM* is a task ranging from trivial to impossible. The reason for the former is that the topology is not hidden at all, when all the observation probability distributions are singular and pairwise distinct. In this case, the HMM is just a regular Markov chain, since the singular distributions support a bijection between states and observation symbols, and all the statistical results for estimating Markov chain parameters apply [59].

The other extreme is a *completely hidden* HMM:

Definition 5.3 *If all observation probability distributions of a given HMM λ are equal, we call λ and its states* completely hidden.

For a completely hidden model, it is clearly impossible to distinguish between states at all. The probability distribution over $\Sigma^\star$ induced by the HMM is only governed by the one observation probability distribution. Hence, inference of topology is impossible.

In the following section, we will introduce a measure for how well a HMM allows us to distinguish between individual states. Alternatively, the extent to which those states are obscured. To our knowledge, ours is the first measure proposed for this purpose. Besides the theoretical benefit of bringing order into HMM space, we also have a distinctive motivation from the application side. For the evaluation of an inference algorithm on artificial data produced by an HMM, the two extremes above have to be avoided, because the resulting inference problem is either trivial or impossible to solve. We can heuristically establish correctness of the inference algorithm from safe ground by use of the following protocol: Start with a Markov chain as the stochastic source, and iteratively use sources with a decreasing amount of distinguishability, i.e., the states becoming more and more hidden. Evaluate the distance between the model inferred and the model producing the data at each step.

Assumption 5.4 *For the following development, we will assume that the stochastic source is stationary and that the underlying Markov chain is ergodic.*

The measure we will introduce is developed from a Bayesian perspective. Given an HMM λ with n states and an observation sequence, O, of finite length t. Consider the state probabilities conditioned on O, σ, that is,

$$\sigma := \sigma(O) := (P(q_t = s_1|O), P(q_t = s_2|O), \ldots, P(q_t = s_n|O)). \tag{5.1}$$

Recall, that we can compute $\sigma(O)$ efficiently with the scaled forward variables (cf. Remark 3.13), since

$$\sigma(O) = (\hat{\alpha}_t(1), \hat{\alpha}_t(2) \ldots, \hat{\alpha}_t(n)). \tag{5.2}$$

Intuitively, it is clear that distinguishability is related to or can be measured in terms of the differences in σ. If σ is uniform for all possible O, then one cannot distinguish between states using observations; i.e., the states are completely hidden. The converse does not hold. Even if σ is singular, it might not be due to the observations, but rather due to peculiarities of the transition matrix, as can be seen from the following example.

Example 5.5 *Let A be a transition matrix such that s_1 is an absorbing state. That is, $P(q_t = s_1) = 1$ as $t \to \infty$. The distribution $\sigma(O)$ as defined above is singular for all O as $|O| \to \infty$.*

Something more involved, capturing the dependency of σ on the observation sequences, is called for.

If A denotes the transition matrix of the underlying Markov chain, let ρ be the equilibrium distribution with respect to A. That is,

$$\rho = (P(q_t = s_1), P(q_t = s_2), \ldots, P(q_t = s_n)) \text{ as } t \to \infty. \tag{5.3}$$

One can interpret the equilibrium distribution of the Markov chain as the *prior* distribution on states, that is, prior to making any observation. For an observation sequence O, the conditional state probability $\sigma(O)$ defined above can then be regarded as the corresponding *posterior* distribution. Naturally, and as an application of information theory, one can then investigate the gain of information attributed to the observation sequence. In the case of a Markov chain, the *observation* sequence is equivalent to the *state* sequence. Hence, we would expect a maximal gain of information in this case. Correspondingly, for a completely hidden HMM, an observation would yield zero information gain; i.e., prior and posterior distributions are equal.

In this setting, and for a fixed observation sequence O, relative entropy, cf. Def. 2.22, provides a measure for the gain in information between posterior and prior: For $O = O_1 \cdots O_T$,

$$\mathcal{H}(\sigma(O), \rho) := \sum_{i=1}^{n} P(q_T = s_i|O) \log \frac{P(q_T = s_i|O)}{P(q_T = s_i)}. \tag{5.4}$$

To avoid possible artifacts from considering only one particular observation sequence, we will average information gain over all observations. Note, that due to stationarity and ergodicity,

this is equivalent to averaging over all prefixes of an observation sequence O of infinite length, yielding the *average information gain*,

$$\mathcal{AI}(\sigma(O), \rho) := \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{n} P(q_t = s_i | O[1,t]) \log \frac{P(q_t = s_i | O[1,t])}{P(q_t = s_i)}. \tag{5.5}$$

In practice, we will restrict ourselves to averaging over all prefixes of a *finite* observation sequence.

Definition 5.6 (Measure of Distinguishability) *Given an HMM $\lambda = (A, B, \pi)$ and an integer T, define*

$$\mathcal{MD}(\lambda) := \mathcal{MD}(\lambda, T) := \frac{1}{T} \sum_{t=1}^{T} \mathcal{H}(\sigma(O[1,t]), \rho), \tag{5.6}$$

where O is a finite observation sequence of length T produced by λ, and ρ is the equilibrium distribution of A.

Lemma 5.7 *Under assumption 5.4, $\mathcal{MD}(\lambda)$ is a consistent and unbiased estimator of $\mathcal{AI}(\sigma(O), \rho)$.*

Proof. This follows immediately from the law of large numbers as implied by the ergodicity and stationarity assumptions. □

In the following we will prove that $\mathcal{MD}$ takes on large values if the HMM is in fact a Markov chain, and zero if it is completely hidden. Later, we investigate the space in-between by using parameterized families of HMMs, covering the spectrum spanning from one theoretically explored extreme to the other. First, we introduce some machinery.

Remark 5.8 *Let P and Q be two discrete probability distributions, P singular, say $p_i = 1$, and $q_i > 0$ for all i. Analogous to the definition of the entropy $\mathcal{H}(P)$ as zero for singular P, we will make use of the convention $p \log p := 0$ for $p = 0$ and thus obtain*

$$\mathcal{H}(P; Q) = -\log(q_i). \tag{5.7}$$

Lemma 5.9 *Iff λ is an HMM, where B supports a bijection between states and observation symbols, that is, λ is equivalent to a Markov chain, then $\sigma(O)$, for all $1 \le t \le T$, is also singular for all O with $P(O|\lambda) > 0$. Moreover, there is a bijection between the observation symbols and the N pairwise distinct singular distributions $\sigma(O)$.*

Proof. Without loss of generality assume that

$$b_i = (0, \ldots, 0, \overbrace{1}^{i}, 0, \ldots, 0). \tag{5.8}$$

We first show that $\sigma(O)$ is singular. From Eq. (3.30) and

$$b_i(O) = \begin{cases} 1 & \text{if } O = i, \\ 0 & \text{otherwise} \end{cases} \tag{5.9}$$

we obtain in that case that

$$\tilde{\alpha}_1(i) = \begin{cases} \pi_i & \text{if } O_1 = i, \\ 0 & \text{otherwise} \end{cases} \tag{5.10}$$

and thus, after scaling, using the non-zero likelihood of O to ensure that $\pi_i > 0$,

$$\hat{\alpha}_1(i) = \begin{cases} 1 & \text{if } O_1 = i, \\ 0 & \text{otherwise.} \end{cases} \tag{5.11}$$

Using Eq. (3.31), we obtain for the induction step

$$\tilde{\alpha}_t(i) = \begin{cases} \sum_{j=1}^{N} \alpha_{t-1}(j) a_{ji} & \text{if } O_t = i, \\ 0 & \text{otherwise.} \end{cases} \tag{5.12}$$

The only summands which do not vanish above are those with $j = O_{t-1}$, yielding, as $\alpha_{t-1}(O_{t-1}) = 1$,

$$\tilde{\alpha}_t(i) = \begin{cases} a_{ji} & \text{if } O_t = i, \\ 0 & \text{otherwise} \end{cases} \tag{5.13}$$

and, after scaling,

$$\hat{\alpha}_t(i) = \begin{cases} 1 & \text{if } O_t = i, \\ 0 & \text{otherwise.} \end{cases} \tag{5.14}$$

From the last equation follows that $\sigma(O)$ is singular, that $\sigma(O)$ and $\sigma(O')$ are equal, iff O and O' have the same last symbol. Whence the existence of the bijection between the $\sigma(O)$ and observation symbols.

For the converse, it suffices to consider N observation sequences $O^{(i)}$, each having non-zero likelihood, ending in N distinct observation symbols. Let f be the bijection between the observation symbols and the N pairwise-distinct singular distributions $\sigma(O)$, which we extend to operate on observation sequences, using their last symbol, and mapping to the unique unit entry in $\sigma(O)$. As the $\sigma(O^{(i)})$ distributions are singular, we obtain from Eq. (3.30) and Eq. (3.31) the following for $i = 1, \ldots, N$:

$$\kappa(O^{(i)}, j) \quad b_j(O^{(i)}_{-1}) \begin{cases} > 0 & \text{if } j = f(O^{(i)}), \\ = 0 & \text{otherwise.} \end{cases} \tag{5.15}$$

The $\kappa(O, i)$ are simply the factors multiplied by the $b_i(O_t)$ in Eq. (3.30) and Eq. (3.31) and are non-zero since $P(O|\lambda) > 0$. For a fixed $b_k(\cdot)$ we obtain by collecting the relevant (in)equalities above

$$\begin{array}{lllll} \kappa(O^{(i)}, k) & b_k(O^{(i)}_{-1}) & > & 0, \text{ if } k = f(O^{(i)}), \text{ and} \\ \kappa(O^{(j)}, f(O^{(j)})) & b_k(O^{(j)}_{-1}) & = & 0, \text{ for } k \neq f(O^{(j)}). \end{array} \tag{5.16}$$

This yields the singularity of $b_k(\cdot)$ and completes the proof. □

Lemma 5.10 *Let λ be an HMM and ρ the corresponding equilibrium distribution, then*

$$\mathcal{MD}(\lambda, T) \leq \mathcal{H}(\rho), \tag{5.17}$$

in the limit, as $T \to \infty$ with probability one. Equality above holds, iff the discrete probability distributions $\sigma(O)$ are singular for all O.

Proof. From the definition we have

$$\mathcal{MD}(\lambda, T) = \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{n} \sigma(O[1,t])_i \log \left(\frac{\sigma(O[1,t])_i}{\rho_i} \right), \tag{5.18}$$

which we can rewrite as

$$\frac{1}{T} \sum_{t=1}^{T} \left\{ \sum_{i=1}^{n} \sigma(O[1,t])_i \log \left(\sigma(O[1,t])_i\right) + \sum_{i=1}^{n} \sigma(O[1,t])_i \log \left(\frac{1}{\rho_i} \right) \right\} \tag{5.19}$$

and bound from above with

$$\frac{1}{T} \sum_{t=1}^{T} \left\{ \sum_{i=1}^{n} \sigma(O[1,t])_i \log \left(\frac{1}{\rho_i} \right) \right\}, \tag{5.20}$$

as $\log \left(\sigma(O[1,t])_i\right) < 0$. By interchanging the order of summation, we obtain

$$\sum_{i=1}^{n} \left\{ \sum_{t=1}^{T} \frac{\sigma(O[1,t])_i}{T} \right\} \log \left(\frac{1}{\rho_i} \right). \tag{5.21}$$

Under the assumptions of this section

$$\mathbf{E}[\sum_{t=1}^{T} \frac{\sigma(O[1,t])_i}{T}] \to \rho_i, \tag{5.22}$$

that is, the sums are estimators for the equilibrium state probabilities. Hence,

$$\mathcal{MD}(\lambda, T) \leq \sum_{i=1}^{n} \rho_i \log \left(\frac{1}{\rho_i} \right) \tag{5.23}$$

in the limit, which completes the proof of the inequality.

To show equality, consider

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{n} \sigma(O[1,t])_i \log \left(\sigma(O[1,t])_i\right). \tag{5.24}$$

Singularity of the $\sigma(O[1,t])$ is sufficient for the vanishing of this sum. That singularity is necessary follows from $\mathcal{H}(P) = 0$ implies P is singular. □

Theorem 5.11 *Iff λ is an HMM, where B supports a bijection between states and observation symbols, that is, λ is equivalent to a Markov chain, then $MD(\lambda) = \mathcal{H}(\rho)$.*

Proof. Let $f : \alpha \to \mathcal{S}$ be the bijection between states and observation symbols, which we extend to a map from observation sequences to states by applying it to the last symbol of the sequence. With the preceding lemmas we have that

$$\mathcal{H}(\sigma(O), \rho) = \log\left(\frac{1}{\rho_{f(O)}}\right), \tag{5.25}$$

as under the assumptions in that section $\rho_i > 0$ for all i. Hence we can rewrite Eq. (5.6) as

$$\mathcal{MD}(\lambda, T) = \frac{1}{T}\sum_{t=1}^{T} \log\left(\frac{1}{\rho_{f(O[1:t])}}\right). \tag{5.26}$$

If we introduce counting variables $c_T(i)$, which represent the number of $\log\left(\frac{1}{\rho_i}\right)$ summands above, we obtain

$$\mathcal{MD}(\lambda, T) = \frac{1}{T}\sum_{i=1}^{N} c_T(i) \log\left(\frac{1}{\rho_i}\right). \tag{5.27}$$

Note, that the counting variables $c_T(i)$ divided by T are estimators for the probability of being in state i under the assumptions in this section. That is,

$$\rho_i = \mathbf{E}[\frac{c_T(i)}{T}], \tag{5.28}$$

and moreover

$$\lim_{T\longrightarrow\infty} \frac{c_T(i)}{T} = \rho_i, \tag{5.29}$$

which yields

$$\lim_{T\longrightarrow\infty} \mathcal{MD}(\lambda, T) = -\sum_{i=1}^{N} \rho_i \log(\rho_i) = \mathcal{H}(\rho). \tag{5.30}$$

The converse follows immediately from Lemma 5.10, using Lemma 5.9. □

Theorem 5.12 *Iff λ is an HMM, where all observation probability distributions are equal, then $MD(\lambda) = 0$.*

Proof. To show that the conditions are sufficient, we compute $\sigma(O)$ with the help of the scaled forward variables defined in Sec. 3.2.4, and prove the theorem by showing that, given identical observation probability distributions for all states,

$$(\hat{\alpha}_t(1), \ldots, \hat{\alpha}_t(n)) = (A^T)^{(t-1)}\,\pi, \tag{5.31}$$

where A^T denotes the transpose of A. As the HMM is ergodic and stationary, successive powers of A^T applied to the initial and, under these assumptions, stationary distribution, is equal to ρ, yielding

$$\mathcal{H}(\sigma(O), \rho) = \mathcal{H}(\rho, \rho) = 0, \tag{5.32}$$

which proves the claim.

We will prove Eq. (5.31) by induction. For $t = 1$, $\hat{\alpha}_1(i) = \pi_i$ follows from Eq. (3.30), as the $b_i(O_1)$ are pairwise equal and hence $c_1 = (b_i(O_1))^{-1}$. In the induction, observe that Eq. (3.31) is just the element-wise written multiplication of A^T with

$$(A^T)^{(t-2)}\pi = (\hat{\alpha}_{t-1}(1), \ldots, \hat{\alpha}_{t-1}(n)) \tag{5.33}$$

since the $b_i(O_t)$ factors are equal and subsequently cancel out in the scaling step. Together this establishes Eq. (5.31).

Conversely, if $\mathcal{AI}(\sigma(O), \rho) = 0$, we have $\mathcal{H}(\sigma(O), \rho) = 0$ due to ergodicity and stationarity, and thus $\sigma(O) = \rho$ (cf. 2.24). Assume $b_i(O_1) \neq b_j(O_1)$ exists. This contradicts $\sigma(O) = \rho$, recalling Eq. (3.30), and completes the proof as O_1 is arbitrary. □

To demonstrate the usefulness of the measure we next consider the following two parameterized families of HMMs, which run the gamut from Markov chain to completely hidden HMM.

Definition 5.13 (2-Coin HMM) *Let $\lambda(x, y, \varepsilon)$ be an HMM with a uniform initial distribution and transition and emission matrix A and B, respectively, defined as*

$$A(x, y) := \begin{pmatrix} x & 1-x \\ 1-y & y \end{pmatrix}, \quad B(\varepsilon) := \begin{pmatrix} 1-\varepsilon & \varepsilon \\ \varepsilon & 1-\varepsilon \end{pmatrix}. \tag{5.34}$$

Remark 5.14 *For a 2-coin HMM $\lambda(x, y, \varepsilon)$, parameters x and y control the equilibrium distribution of $A(x, y)$, which is proportional to $(1-x, 1-y)$; x and y can be chosen from $[0, 1]$. The parameter $\varepsilon \in [0, \frac{1}{2}]$ controls the distinguishability of states of $\lambda(x, y, \varepsilon)$. For $\varepsilon = 0$ we obtain a Markov chain, for $\varepsilon = \frac{1}{2}$ the two states are completely hidden.*

Fig. 5.1 shows the dependency of the measure of distinguishability on the parameter ε for some fixed pairs of values x, y. In Fig. 5.2 we evaluate the measure on the following model family.

Definition 5.15 (3-Coin HMM) *Let λ be an HMM with three states and two output symbols. Let a_{12}, a_{13}, a_{21}, a_{23}, a_{31}, and a_{32} denote the free transition parameters and b_1, b_2, and b_3 free emission parameters. That is, the emission matrix B is defined as*

$$B := \begin{pmatrix} b_1 & 1-b_1 \\ b_2 & 1-b_2 \\ b_3 & 1-b_3 \end{pmatrix}. \tag{5.35}$$

The initial distribution is uniform.

Remark 5.16 *The parameters b_i control the distinguishability of states of λ. If the b_i are equal, then the states are completely hidden. Since the number of states is larger than the number of emission symbols, we cannot obtain a Markov chain for any choice of b_i.*

5.2.1 Comparing *MD* and the Probabilistic Distance

An interesting question is, whether the measure *MD* can be used to improve discrimination between HMMs by capturing differences not detectable by use of the probabilistic distance measure 3.32. To investigate this question heuristically, we computed the probabilistic distance between pairs of 2-coin models. In Fig. 5.3 (note the different scales on the *y*-axes), each of the graphs depicts the probabilistic distance between a fixed 2-coin model $\lambda(x, y, \varepsilon_0)$ and 2-coin models $\lambda(x, y, \varepsilon)$ for $\varepsilon > \varepsilon_0$. It can be observed that the probabilistic distance provides insufficient discrimination between Markov chains and completely hidden HMMs, when the transition probabilities are both non zero. By comparing the corresponding distance values with the *MD* versus ε plots for particular 2-coin models (cf. Fig. 5.1), it becomes apparent, that the difference in distinguishability $MD(\lambda(x, y, \varepsilon_0)) - MD(\lambda(x, y, \varepsilon)$ could be used to add further discrimination between models, particularly when the probabilistic distance fails.

5.2.2 Extension to transient HMMs

Relaxing the assumptions in the previous section makes a theoretical investigation infeasible. Nevertheless, we present experimental results supporting the usefulness of the measure introduced in the previous section even in this case. We can extend the measure to transient HMMs with the following approach. Instead of considering the equilibrium distribution of the underlying Markov chain, we will use the image of the *initial* distribution under successive powers of the transition matrix *A*. For an ergodic model, the distribution obtained in that way would converge to the unique equilibrium distribution. For transient models, neither existence nor uniqueness of the equilibrium distribution are assured in general. In case of left-right models (cf. Def. 3.19), the terminal states are recurrent with probability one due to stochasticity constraints, yielding for each of the terminal states a different equilibrium distribution. Moreover, the singular equilibrium distribution resulting for left-right models, would not capture the information gain caused by making observations, even if there were one unique distribution.

Definition 5.17 (Measure of Distinguishability: Transient case) *Given an HMM $\lambda = (A, B, \pi)$, and denote by $\mathcal{O}$ a set of observation sequences $\mathcal{O}$ produced by λ. Let $T := \sum_{O \in \mathcal{O}} |O|$. Define*

$$\mathcal{MDT}(\lambda) := \mathcal{MDT}(\lambda, \mathcal{O}) := \frac{1}{T} \sum_{O \in \mathcal{O}} \sum_{t=1}^{|O|} \mathcal{H}(\sigma(O[1:t]), A^{(t-1)} \times \pi). \tag{5.36}$$

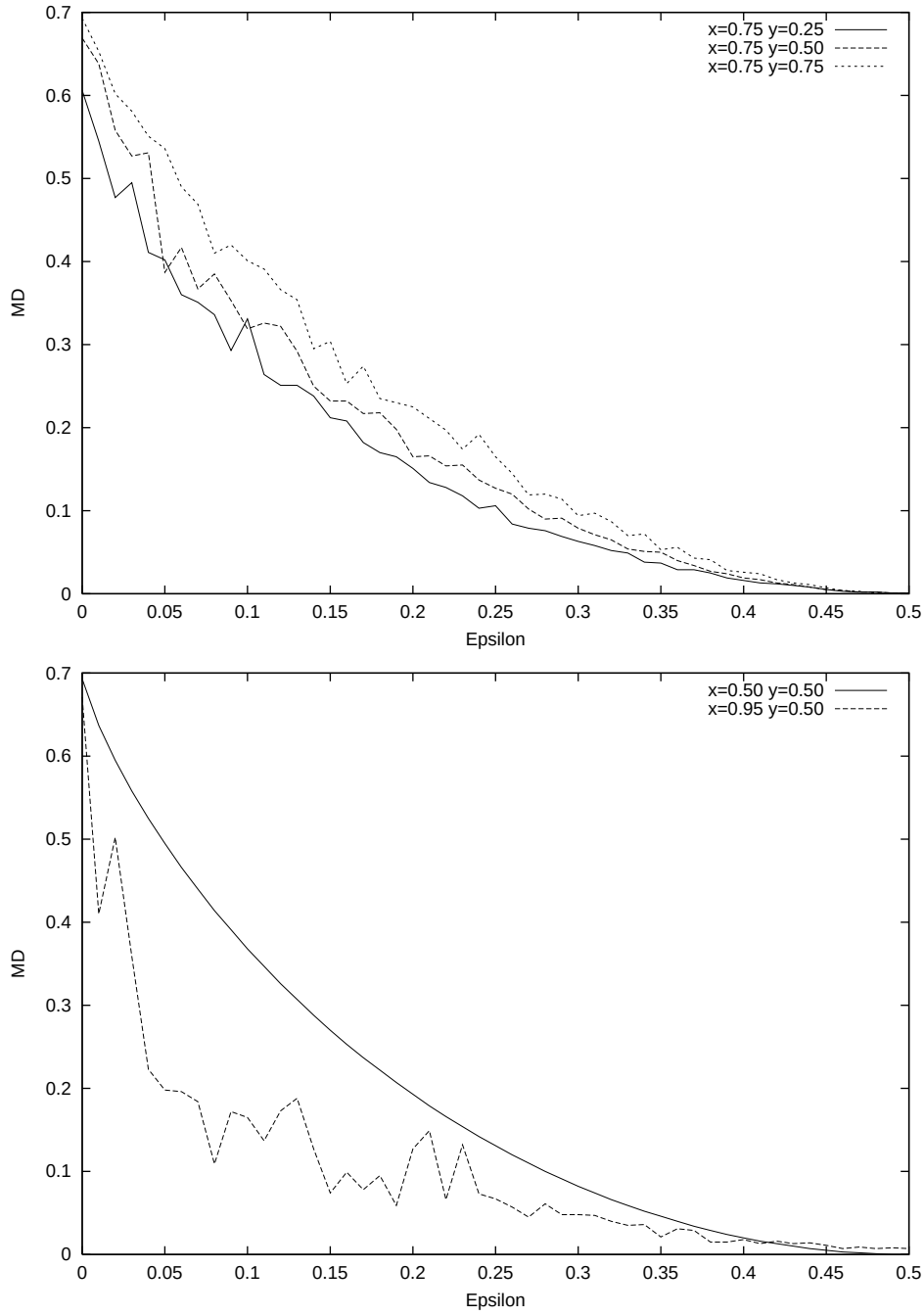

Figure 5.1: *Our measure of distinguishability MD*(λ, T) *evaluated on the parameterized family of 2-coin HMMs (cf. Def. 5.13) for* $x = 0.75$ *and* $y = 0.25, 0.5, 0.75$ *(top) respectively* $x = 0.95, y = 0.5$ *and* $x = y = 0.5$ *(bottom). Each individual data point was computed for an observation sequence of length* $T = 100$*; using larger T smoothes out curves (not shown). Note the dependency of the curve smoothness on the state duration (bottom).*

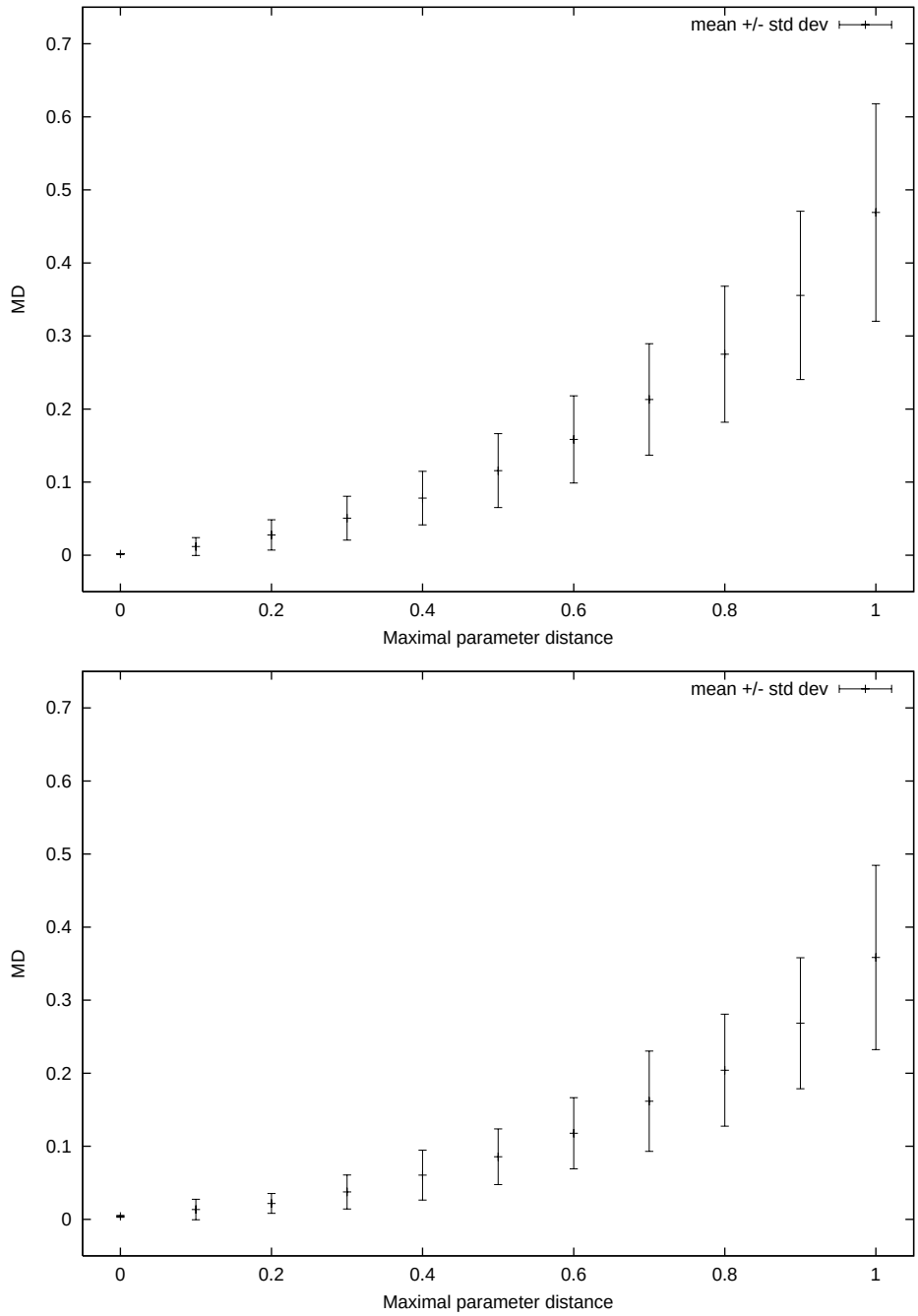

Figure 5.2: *Our measure of distinguishability MD*(λ, T) *is evaluated on the parameterized family of 3-coin HMMs (cf. Def. 5.15). For two fixed choices of transition parameters —* $a_{12} = 0.1$*,* $a_{13} = 0.4$*,* $a_{21} = 0.2$*,* $a_{23} = 0.5$*,* $a_{31} = 0.3$*, and* $a_{32} = 0.1$ *(top)* $a_{12} = 0.1$*,* $a_{13} = 0.4$*,* $a_{21} = 0.2$*,* $a_{23} = 0.5$*,* $a_{31} = 0.3$*, and* $a_{32} = 0.1$ *(bottom) — and* $b_1, b_2, b_3 = 0, 0.1, \ldots, 1$ *we have computed the measure of distinguishability MD*(λ) *for an observation sequence of length* $T = 100$*. We show the average value and the standard deviation of MD*(λ) *vs. the maximal distance between the* b_i *parameters. That is, the values on the x-axis are* $\max\{b_1, b_2, b_3\} - \min\{b_1, b_2, b_3\}$ *rounded to one decimal place.*

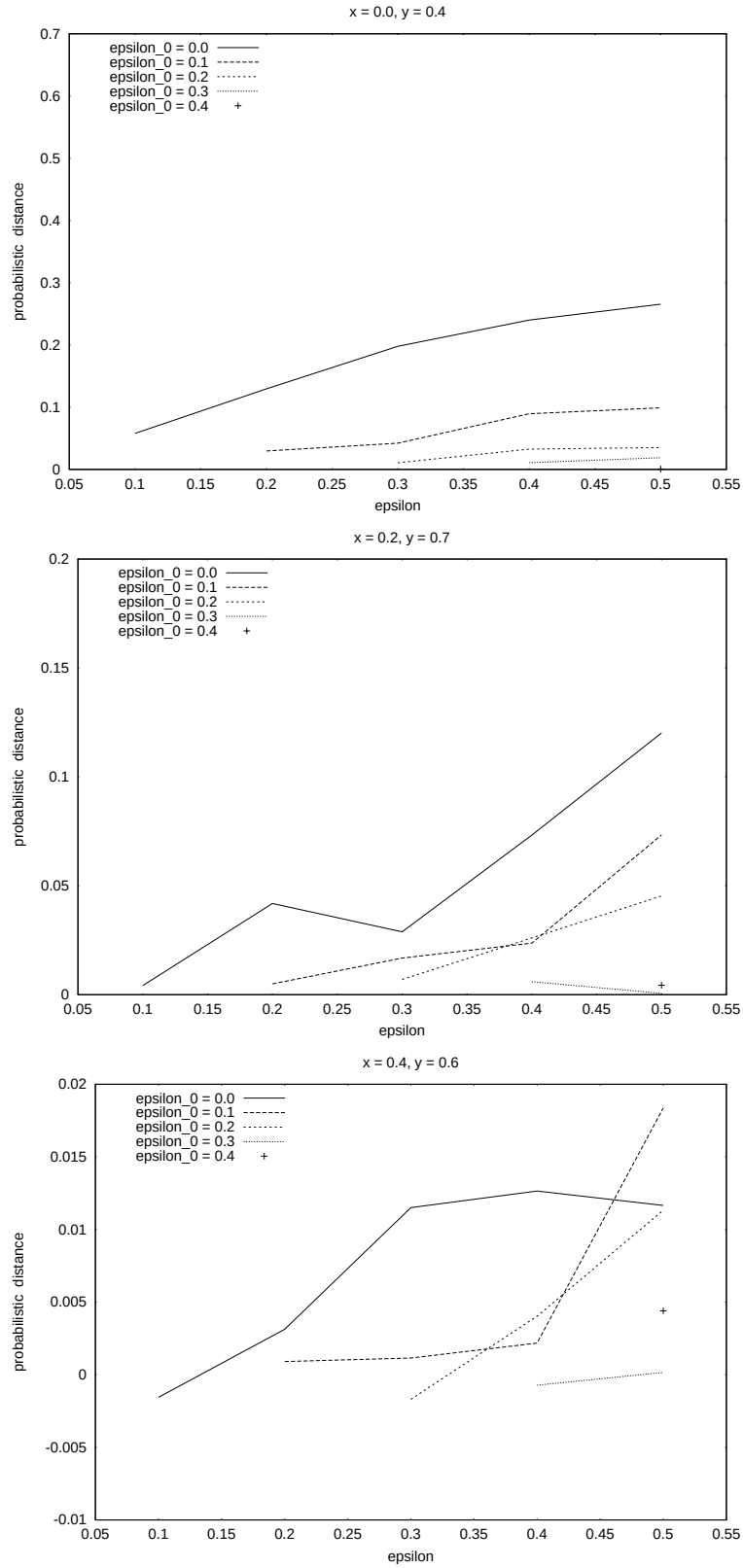

Figure 5.3: *To investigate how the measure of distinguishability compares to the probabilistic distance, we computed the latter for various x and y values. The graphs display the distance between the 2-coin Models* $\lambda(x, y, \varepsilon_0)$ *and* $\lambda(x, y, \varepsilon)$ *for* $x = 0.0, y = 0.4$ *(top),* $x = 0.2, y = 0.7$ *and* $x = 0.4, y = 0.6$ *(bottom).*

5.3 Reliability of relative k-tail Frequencies

To obtain insight into the statistical reliability of the relative k-tail frequencies — in dependence of window length and k-tail depth and sequence length — we performed the following experiment. We picked an HMM and generated a number of sequences of various lengths. Starting from this artificial sequence data, we computed prefix trees for a number of choices of w, the window length, and k, the depth of the k-tails. This yielded, for a fixed w and k, a number of prefix trees, PT_j, in which we collected the relative k-tails frequencies $r(x;k)$ corresponding to the same vertex v_x. Subsequently, we computed for each $1 \leq i \leq |\Sigma|^k$ the average and the standard deviation over $\{r(x;k)_i\}_{PT_j}$. This yielded for the trees PT_j a set of standard deviations

$$\{\sigma(x,i) | v_x \in PT_{\varepsilon,w-k}, 1 \leq i \leq |\Sigma|^k, \tag{5.37}$$

whose maximum value we show in Table 5.1.

When comparing entries in Table 5.1, recall that two prefix trees have the same number of vertices only if the respective differences $w-k$ are equal, and, qualitatively speaking, that relative k-tail frequencies go to zero exponentially as k increases. There are the following observations to be made:

- Doubling the sequence length is apparently reducing the standard deviation by a factor of approximately $\sqrt{2}$.
- When increasing w from six to ten for $k = 3$ fixed, an increase of sequence length by an eight-fold is necessary to obtain a similar standard deviation.

Conclusively, the relative k-counts seem to be sufficiently reliable. Naturally, the counts with a higher degree of variation are to be found deeper in the tree. Note, that we observed comparable behavior for other models (not shown).

5.4 Results for the 2-Coin Family of HMMs

To evaluate the performance on the family of 2-coin HMMs, we repeated the following experiment, both for single-link clustering (SLC) and weighted average hierarchical clustering (WAH), for $x = 0.0, 0.1, \ldots, 0.9$, $y = x, \ldots, 0.9$ and $e = 0.0, 0.05, \ldots, 0.4$. For each of the combinations of the window length w and tail depth k — $(w = 4, k = 2)$, $(w = 5, k = 2)$, $(w = 5, k = 3)$, $(w = 6, k = 2)$, $(w = 6, k = 3)$, $(w = 7, k = 3)$, $(w = 7, k = 4)$, $(w = 8, k = 3)$, $(w = 8, k = 4)$, $(w = 8, k = 4)$, $(w = 9, k = 4)$, and $(w = 9, k = 4)$ — we used $\lambda(x, y, e)$ to generate a random sequence of length $T = 250$ respectively $T = 1000$. For each observation sequence, we inferred a HMM using the relative k-tail frequencies as features and using one of the following distance functions between features: $\mathcal{D}$, $\mathcal{D}^+$, $\mathcal{D}^{50}$, $\mathcal{D}^L$, $\mathcal{D}^{BR}_1$, $\mathcal{D}^{BR}_{10}$, and $\mathcal{D}^{BR}_{50}$.

Since we wanted to compare several distance functions, choosing a prior on the distances proved problematic, as the ranges of the functions and their qualitative behavior are not comparable. To circumvent the danger of biasing against or in favor of particular distance functions, we decided to use a prior on the number of states directly. We do think, that this is an inferior

		Maximal standard deviation				
w	k	$T = 250$	T=500	$T = 1000$	$T = 2000$	$T = 4000$
6	2	0.00910	0.00465	0.00141	0.00117	0.00050
6	3	0.00178	0.00098	0.00026	0.07747	0.00033
7	3	0.00287	0.00234	0.00059	0.00069	0.00037
7	4	0.00130	0.00062	0.00013	0.00014	0.00007
8	3	0.00556	0.00476	0.00142	0.00088	0.00062
8	4	0.00127	0.00060	0.00025	0.00067	0.00025
10	3	0.02440	0.01583	0.00907	0.00539	0.00249
10	4	0.00390	0.00305	0.00113	0.00073	0.00058
10	5	0.00080	0.00040	0.00028	0.00033	0.00022

		Maximal standard deviation				
w	k	$T = 250$	T=500	$T = 1000$	$T = 2000$	$T = 4000$
6	2	0.00964	0.00445	0.00312	0.00280	0.00250
6	3	0.00321	0.00189	0.00131	0.00101	0.00080
7	3	0.00528	0.00291	0.00187	0.00138	0.00112
7	4	0.00124	0.00055	0.00023	0.00017	0.00010
8	3	0.00928	0.00454	0.00234	0.00175	0.00128
8	4	0.00282	0.00125	0.00074	0.00056	0.00047
10	3	0.03023	0.01191	0.00635	0.00338	0.00207
10	4	0.00629	0.00323	0.00164	0.00119	0.00081
10	5	0.00193	0.00110	0.00058	0.00032	0.00023

$$A = \begin{pmatrix} 0.8 & 0.2 \\ 0.1 & 0.9 \end{pmatrix} \qquad B = \begin{pmatrix} 0.75 & 0.25 \\ 0.25 & 0.75 \end{pmatrix}$$

Table 5.1: *2-coin HMMs: We generated 10 (top), respectively 100 (middle) random binary observation sequences of length $T = 250, 500, 1000, 2000, 4000$ to investigate the reliability of the relative k-tail frequencies. For fixed k, w and T we computed the mean and standard deviation of corresponding k-tail frequencies for the corresponding 10, respectively 100, prefix tree vertices. The values in the matrix represent the maximal standard deviation observed. The HMM used to generate the observation sequences is the two-coin model depicted at the bottom. Other models showed similar behavior (not shown).*

handle on the generalization capabilities. Nevertheless, for the sake of the comparison we had no other choice. We used a discrete probability distribution,

$$sp(m)_i := \max\{1 - \frac{|m-i|}{8}, 0\}, \tag{5.38}$$

scaled appropriately to yield a stochastic vector.

In all of the evaluation we used the difference in the likelihood of the training data between true and inferred model, denoted by *rl*. The values thus obtained are proportional to the probabilistic distance measure defined earlier, but are scaled to percent as to facilitate comparisons between different models. The co-emission distance was not used due to computational inefficiency for fully connected HMMs; the matrix distance was shown to be inferior to the probabilistic distance measure.

As far as distance function are concerned, we found $\mathcal{D}_A^{BR}$ and $\mathcal{D}^L$ to clearly outperform the other distance functions.

5.4.1 Comparing Clustering Algorithms

As we can see from comparing Fig. 5.6 and Fig. 5.9 the weighted average hierarchical shows a somewhat better performance.

5.5 Results for the 3-Coin Family of HMMs

We evaluated the performance of the inference algorithm on a number of 3-coin models (see Table 5.2 for a complete list of parameters) using both SLC and WAH clustering and the following distance functions between features $\mathcal{D}$, $\mathcal{D}^+$, $\mathcal{D}^{50}$, $\mathcal{D}^L$, $\mathcal{D}_1^{BR}$, $\mathcal{D}_{10}^{BR}$, and $\mathcal{D}_{50}^{BR}$. As for the 2-coin models, the evaluation was performed for the following combinations of the window length w and tail depth k: $(w = 4, k = 2)$, $(w = 5, k = 2)$, $(w = 5, k = 3)$, $(w = 6, k = 2)$, $(w = 6, k = 3)$, $(w = 7, k = 3)$, $(w = 7, k = 4)$, $(w = 8, k = 3)$, $(w = 8, k = 4)$, $(w = 8, k = 4)$, $(w = 9, k = 4)$, and $(w = 9, k = 4)$.

The results were consistent with the tests on the 2-coin models, as far as better performance of the $\mathcal{D}_A^{BR}$ distance functions, followed by $\mathcal{D}^L$, over the remaining distance functions was concerned. Again, we observed an advantage of WAH clustering over SLC (not shown). The difficulties for source models with not quite maximal $MD(\cdot)$ persisted, cf. the peak around $MD(\lambda) = 0.35$ in the graphs in Fig. 5.10.

The following example is typical for the output from the inference algorithm. To reiterate a point made earlier, it is unlikely to recover the exact topology of the source, as there are usually many models consistent with the input. The source model in this case was the 3-coin model with transition matrix A_0 and emission matrix B_0,

$$A_0 = \begin{pmatrix} 0.9 & 0.05 & 0.05 \\ 0.05 & 0.9 & 0.05 \\ 0.05 & 0.05 & 0.9 \end{pmatrix}, \qquad B_0 = \begin{pmatrix} 0.30 & 0.70 \\ 0.50 & 0.50 \\ 0.80 & 0.20 \end{pmatrix}.$$

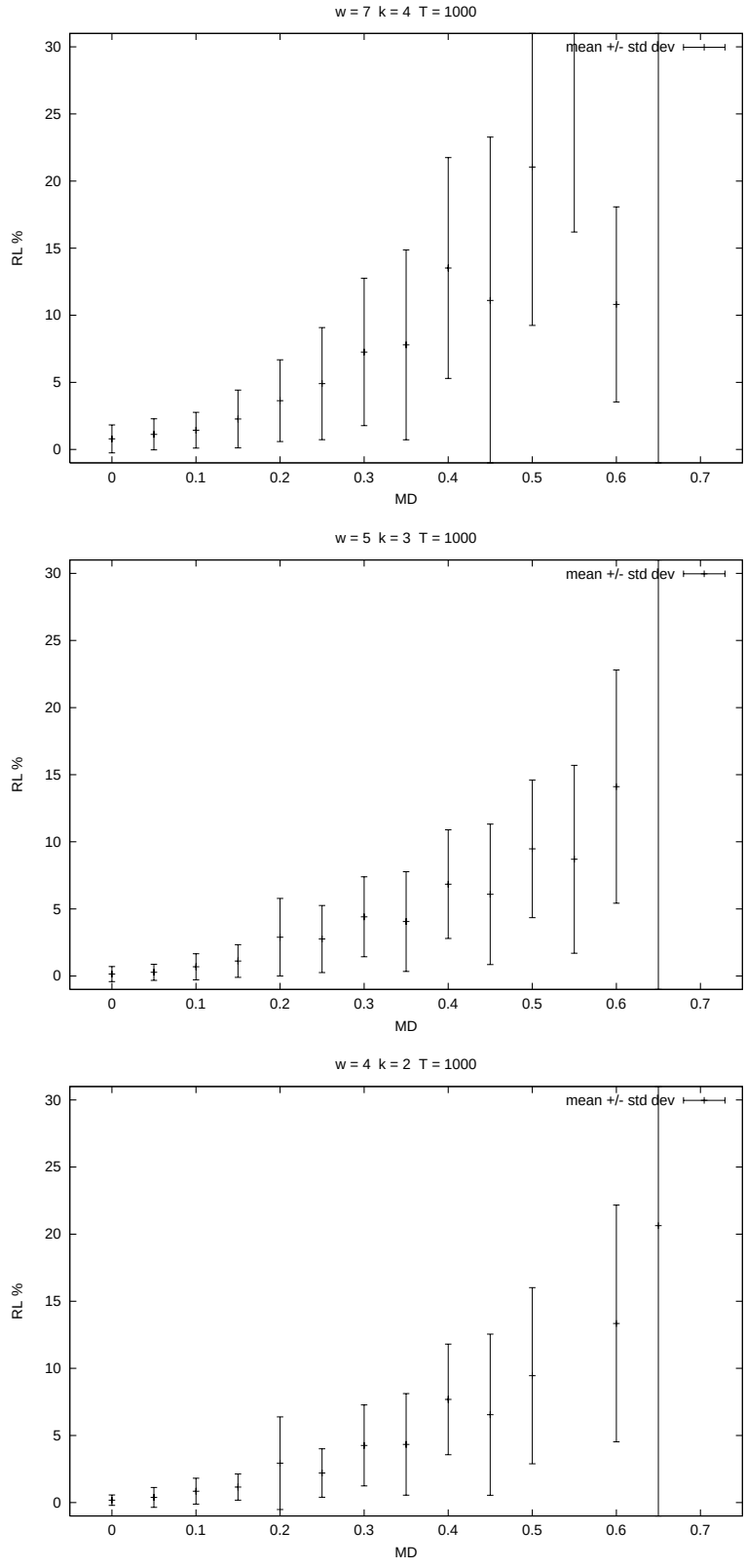

Figure 5.4: *2-coin HMMs, SLC: The influence of w, k and feature on the relative difference in likelihood vs. MD(λ) — binned with a width of 0.05 — is depicted for $\mathcal{D}$ (top), $\mathcal{D}^+$ and $\mathcal{D}^{50}$ (bottom).*

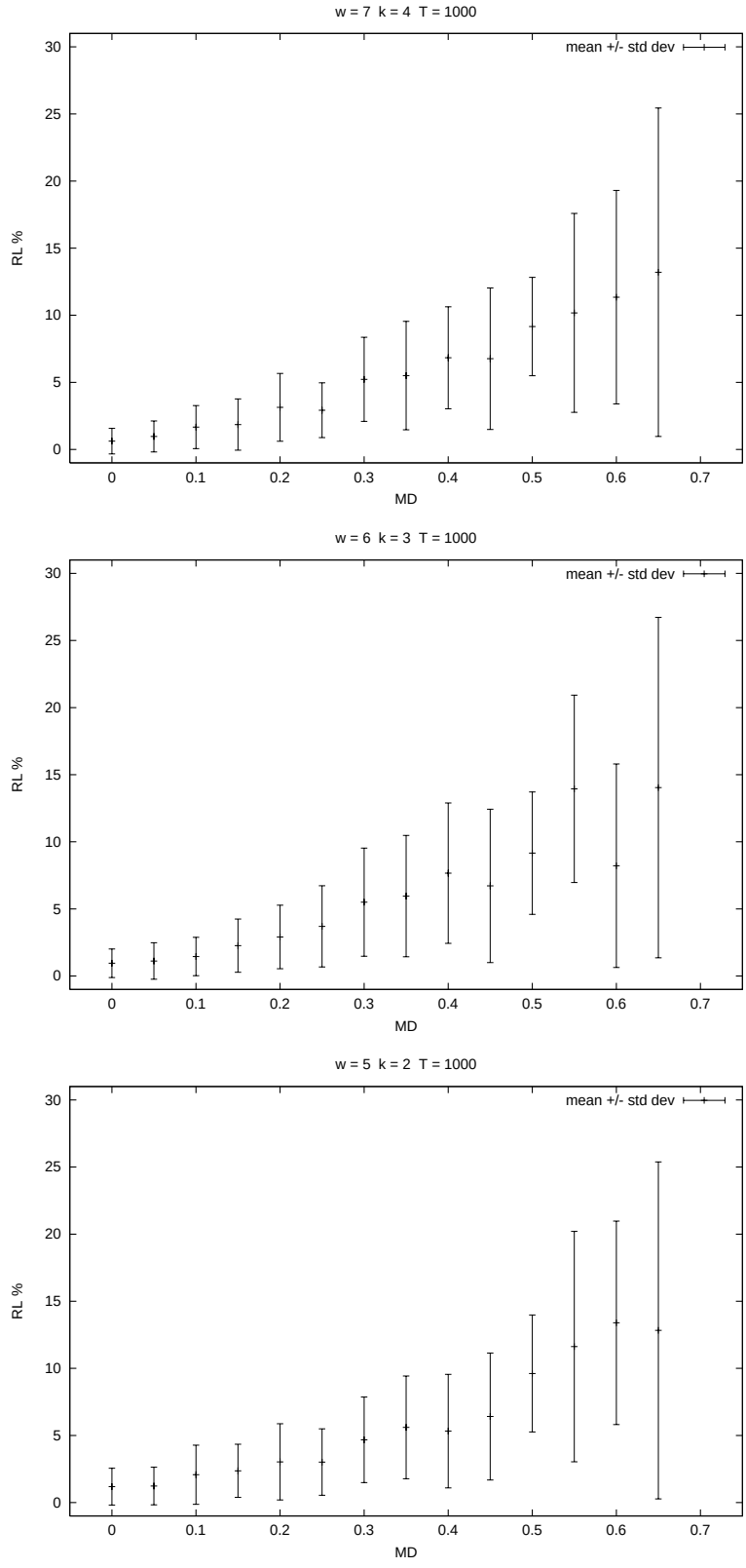

Figure 5.5: *2-coin HMMs, SLC: The influence of w, k and feature on the relative difference in likelihood vs. MD(λ) — binned with a width of 0.05 — is depicted for $\mathcal{D}^{L}$ (top), $\mathcal{D}^{BR}_{1.0}$ and $\mathcal{D}^{BR}_{50.0}$ (bottom).*

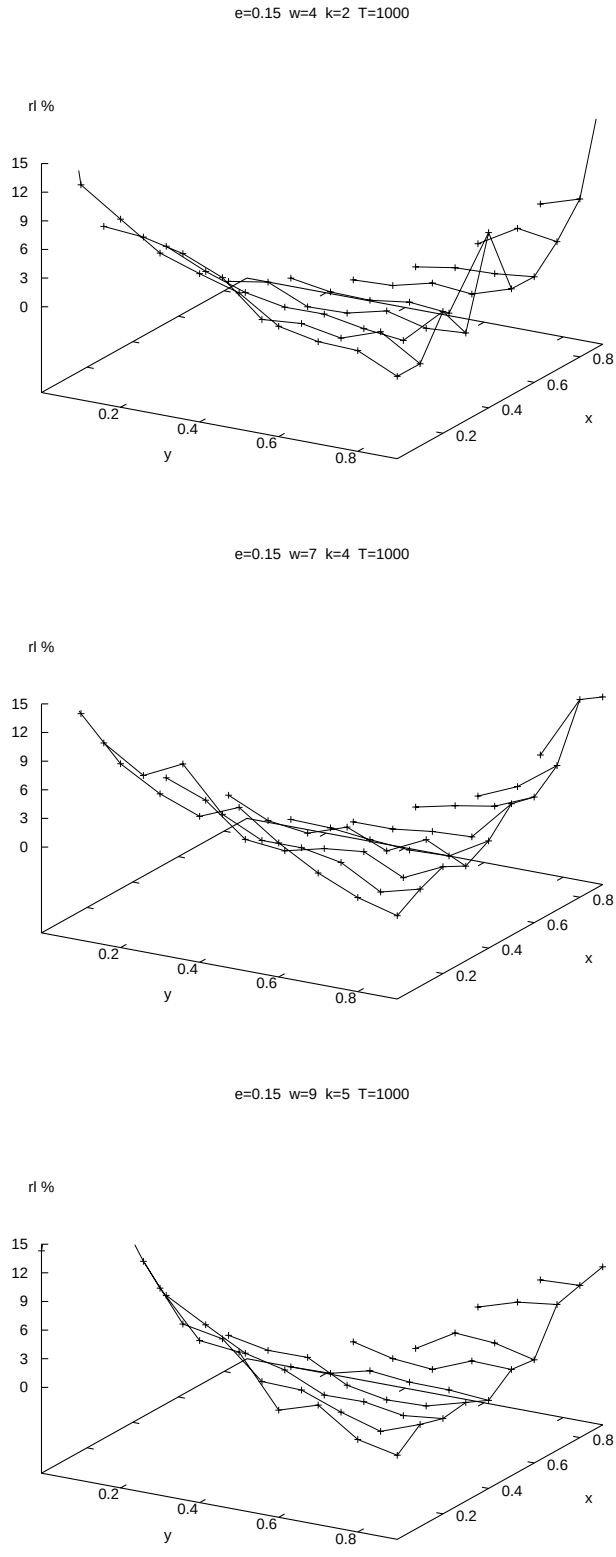

Figure 5.6: *2-coin HMMs, SLC: The influence of w, k the relative difference in likelihood is depicted in dependence on x and y, using $\mathcal{D}_{10}^{BR}$ for $w = 4, k = 2$ (top), $w = 7, k = 4$, and $w = 9, k = 5$ (bottom). The emission parameter ε is fixed at 0.15.*

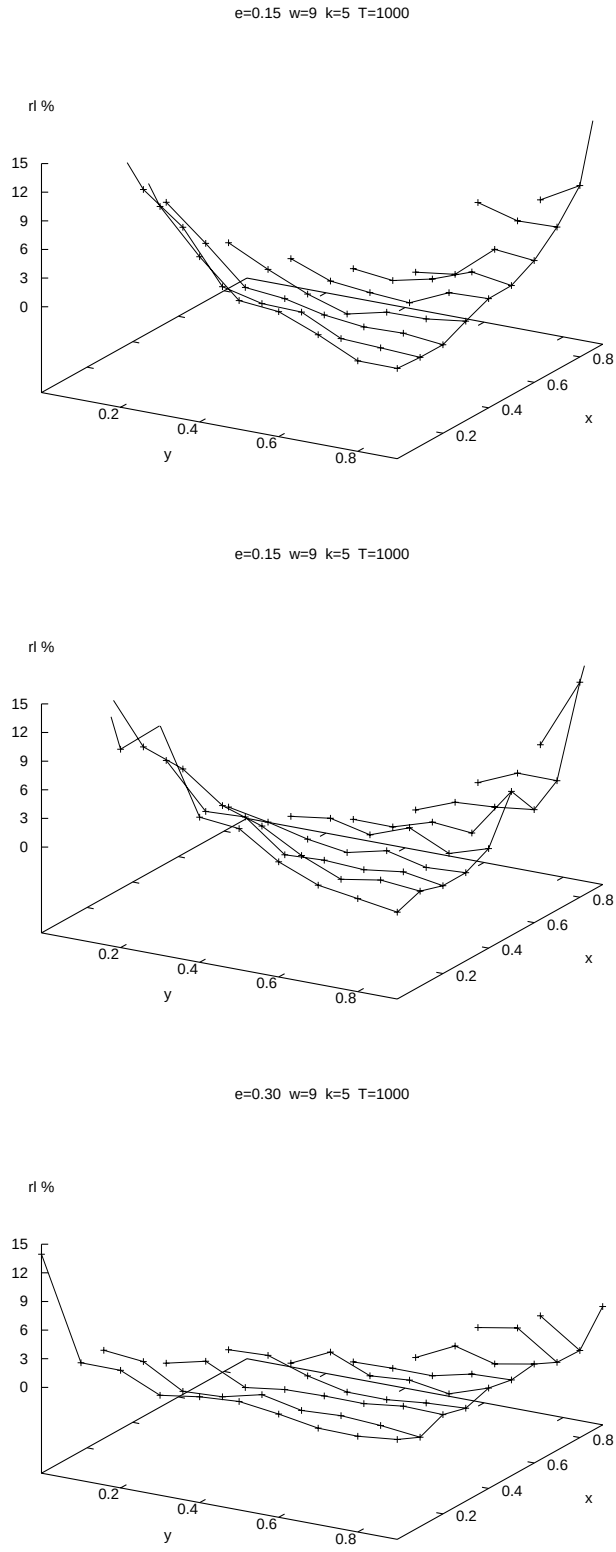

Figure 5.7: *2-coin HMMs, SLC: We show the same type of graphs as in Fig. 5.6, for* $\varepsilon = 0.15$ *using* $\mathcal{D}$ *(top) and* $\mathcal{D}^L$ *(middle). At the bottom we depict the graph resulting for* $\varepsilon = 0.3$ *using* $\mathcal{D}^{BR}_{10}$*. Note,* $w = 9, k = 5$ *in all three graphs.*

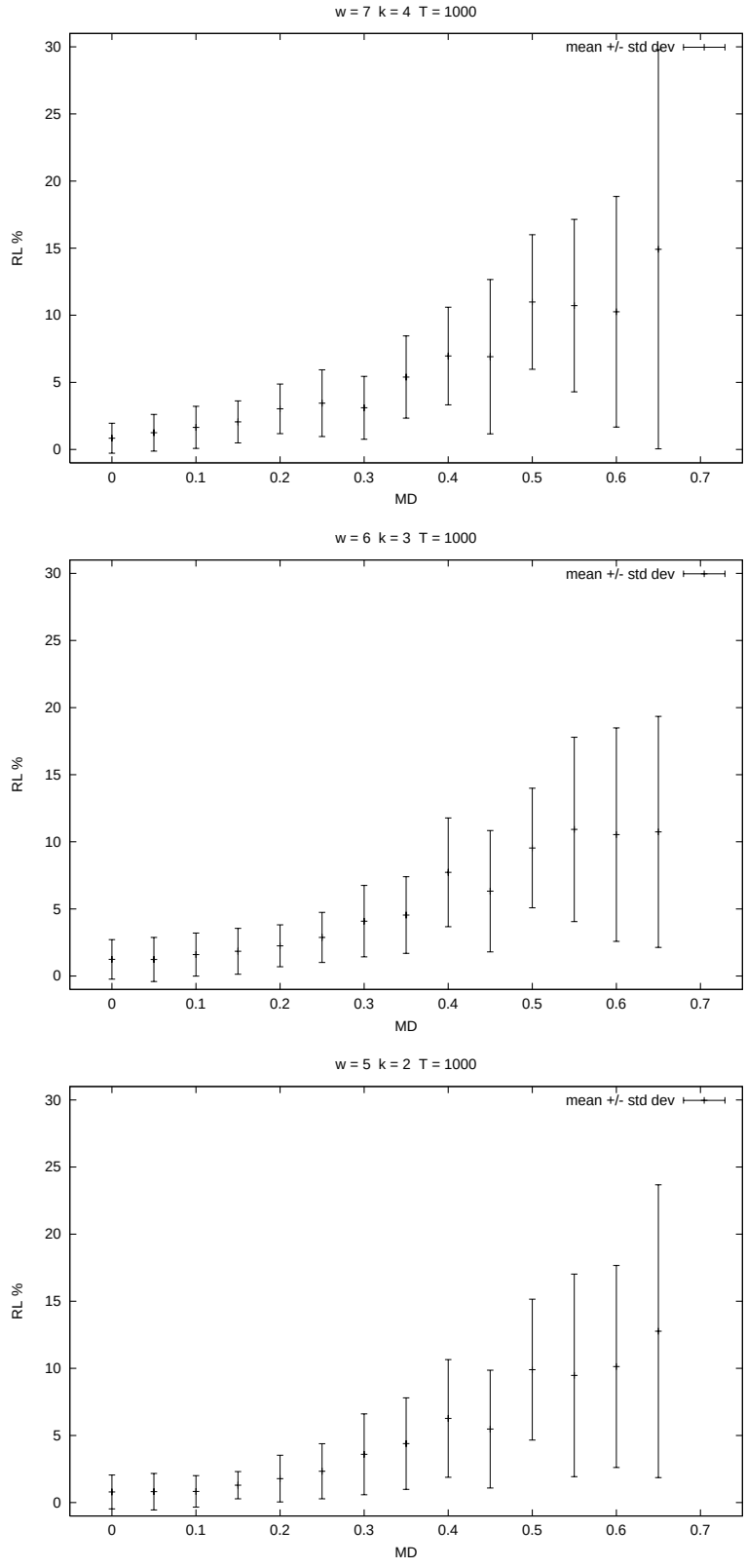

Figure 5.8: *2-coin HMMs, WAH: The influence of w, k and feature on the relative difference in likelihood vs. MD(λ) — binned with a width of 0.05 — is depicted for $\mathcal{D}^{L}$ (top), $\mathcal{D}^{BR}_{1.0}$ and $\mathcal{D}^{BR}_{50.0}$ (bottom).*

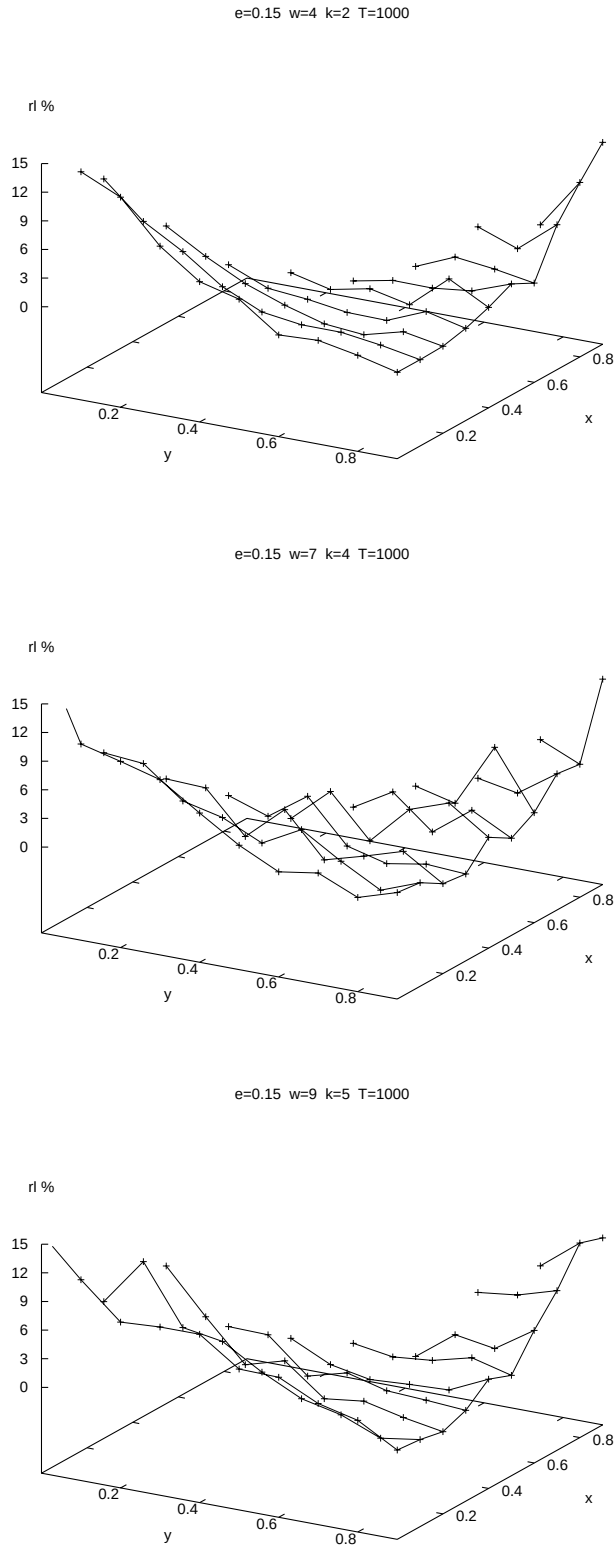

Figure 5.9: *2-coin HMMs, WAH: The influence of w, k on the relative difference in likelihood is depicted in dependence on x and y, using* $\mathcal{D}_{10}^{BR}$ *for* $w = 4, k = 2$ *(top),* $w = 7, k = 4$*, and* $w = 9, k = 5$ *(bottom). The emission parameter* ε *is fixed at 0.15.*

Transitions						Emissions		
a_{12}	a_{13}	a_{21}	a_{23}	a_{31}	a_{32}	b_1	b_2	b_3
0.05	0.05	0.05	0.05	0.05	0.05	0.0	0.5	1.0
0.10	0.10	0.10	0.10	0.10	0.10	0.1	0.2	0.9
0.20	0.20	0.20	0.20	0.20	0.20	0.1	0.3	0.9
0.30	0.30	0.30	0.30	0.30	0.30	0.1	0.4	0.9
0.40	0.40	0.40	0.40	0.40	0.40	0.1	0.5	0.9
0.45	0.45	0.45	0.45	0.45	0.45	0.1	0.6	0.9
0.10	0.10	0.10	0.10	0.20	0.20	0.1	0.7	0.9
0.10	0.10	0.10	0.10	0.30	0.30	0.1	0.8	0.9
0.10	0.10	0.10	0.10	0.40	0.40	0.2	0.3	0.8
0.20	0.20	0.20	0.20	0.10	0.10	0.2	0.4	0.8
0.20	0.20	0.20	0.20	0.30	0.30	0.2	0.5	0.8
0.20	0.20	0.20	0.20	0.40	0.40	0.2	0.6	0.8
0.20	0.60	0.60	0.20	0.25	0.25	0.2	0.7	0.8
0.60	0.10	0.60	0.10	0.25	0.25	0.3	0.4	0.8
0.60	0.10	0.10	0.60	0.60	0.10	0.3	0.5	0.8
0.70	0.10	0.10	0.70	0.70	0.10	0.3	0.6	0.8
0.80	0.10	0.10	0.80	0.80	0.10	0.3	0.7	0.8
0.80	0.10	0.10	0.80	0.35	0.35	0.4	0.5	0.7
0.90	0.0	0.0	0.90	0.0	0.90	0.4	0.6	0.7
0.90	0.10	0.10	0.90	0.50	0.30			
0.10	0.10	0.30	0.40	0.40	0.20			

Table 5.2: *Parameters of 3-coin HMMs for which the inference algorithm was tested. Note, that all combinations consisting of a set of transition parameters and a set of emission parameters were tested.*

The inferred model, with transition and emission matrix A_1 and B_1 respectively,

$$A_1 = \begin{pmatrix} 0.66 & 0.0 & 0.34 \\ 0.0 & 0.58 & 0.42 \\ 0.42 & 0.43 & 0.15 \end{pmatrix}, \qquad B_1 = \begin{pmatrix} 0.66 & 0.34 \\ 0.42 & 0.58 \\ 0.57 & 0.43 \end{pmatrix},$$

yields a log-likelihood of –689.488 instead of –667.232 for the true model. The probabilistic distance between the two models amounts to 0.022. State 2 in the original model, with its uniform emission probabilities and the high self-transition probability of 0.9, seem to cause all the emissions in the inferred model to spread.

The average number of states was 3.01 with a standard deviation of 0.16 for the prior analogously chosen to the previous section, only with the highest weight on 3.

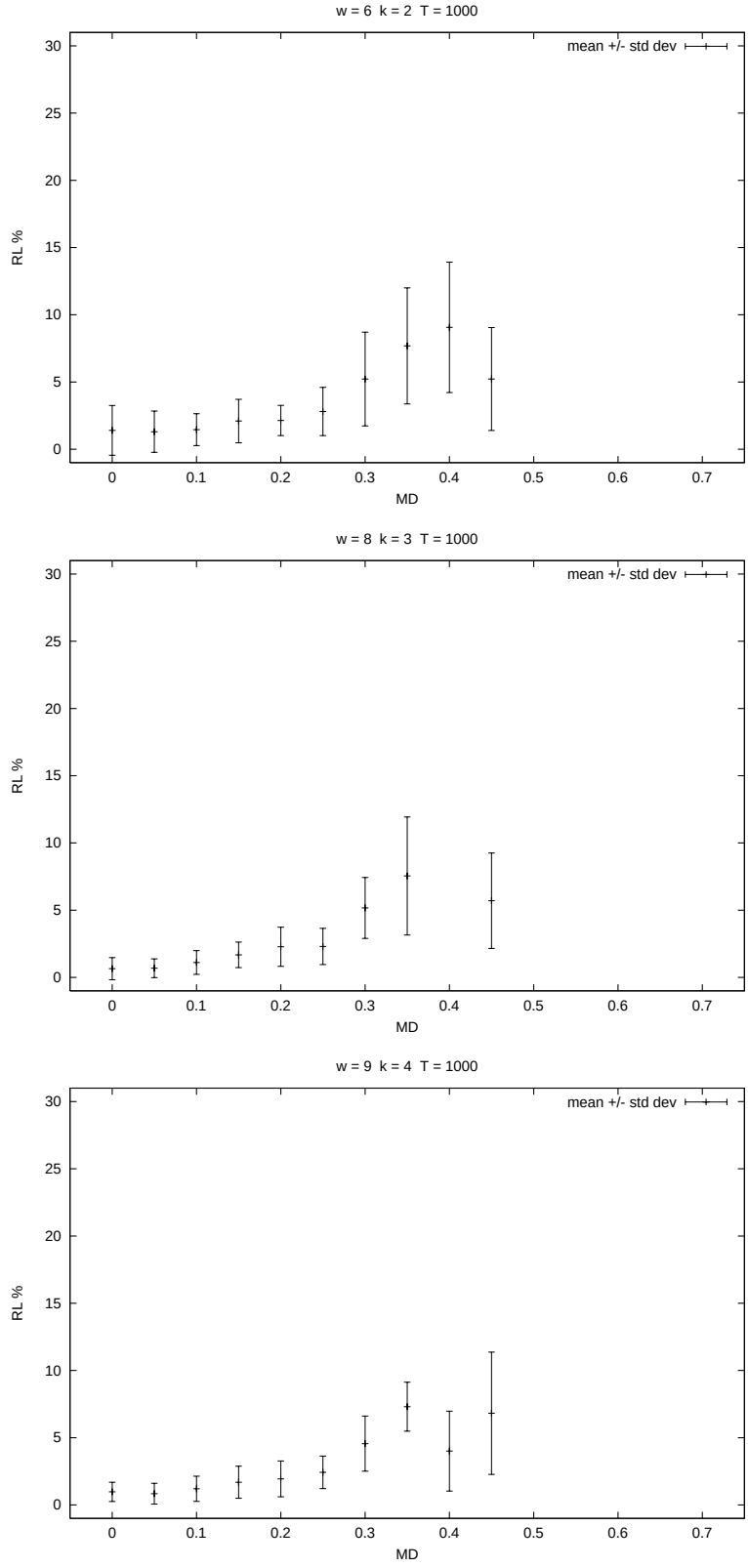

Figure 5.10: *3-coin HMMs, WAH: The influence of w, k and feature on the relative difference in likelihood vs. MD(λ) — binned with a width of 0.05 — is depicted for* $\mathcal{D}^{BR}_{50.0}$.

5.6 Results for Selected HMMs

We also designed a number of HMMs based on specific topological features present, with the goal to investigate for which choice of parameters we could reliably reconstruct an approximation to the original topology. This was motivated by the fact that in biological sequences, *stochastic* regions are often interrupted or structured by *deterministic* signals. Examples are start and stop codons, or the TATA boxes characteristic for promoter sequences in eukaryotic DNA.

In the following A_0 and B_0 will denote the transition and emission matrices for the true model, likewise A_1 and B_1 for the inferred.

Example 5.18 *Here the source transition graph is a 3-cycle. The model was inferred using* $w = 9$, $k = 4$, $\mathcal{D}_1^{BR}$ *and for* $T = 1000$.

$$A_0 = \begin{pmatrix} 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \\ 1.0 & 0.0 & 0.0 \end{pmatrix}, \qquad B_0 = \begin{pmatrix} 0.80 & 0.20 \\ 0.10 & 0.90 \\ 0.30 & 0.80 \end{pmatrix},$$

$$A_1 = \begin{pmatrix} 0.0 & 0.77 & 0.23 \\ 0.0 & 0.16 & 0.84 \\ 0.69 & 0.23 & 0.08 \end{pmatrix}, \qquad B_1 = \begin{pmatrix} 0.66 & 0.34 \\ 0.24 & 0.76 \\ 0.40 & 0.60 \end{pmatrix}.$$

Example 5.19 *Here the source transition graph is a complete directed graph on 2 nodes with a cycle of length 3 attached to one of the nodes. The model was inferred using* $w = 8$, $k = 4$, $\mathcal{D}_{10}^{BR}$ *and for* $T = 1000$.

$$A_0 = \begin{pmatrix} 0.5 & 0.5 & 0.0 & 0.0 \\ 0.1 & 0.4 & 0.5 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \end{pmatrix}, \qquad B_0 = \begin{pmatrix} 0.80 & 0.20 & 0.0 \\ 0.10 & 0.90 & 0.0 \\ 0.0 & 0.2 & 0.80 \\ 0.2 & 0.0 & 0.80 \end{pmatrix},$$

$$A_1 = \begin{pmatrix} 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.951 & 0.049 \\ 0.003 & 0.0 & 0.637 & 0.360 \\ 0.0 & 0.718 & 0.282 & 0.0 \end{pmatrix}, \qquad B_1 = \begin{pmatrix} 0.269 & 0.0 & 0.731 \\ 0.169 & 0.781 & 0.050 \\ 0.183 & 0.447 & 0.370 \\ 0.175 & 0.273 & 0.552 \end{pmatrix}.$$

Example 5.20 *Here the source transition graph is a complete directed graph on 2 nodes with a path of length 3 joining the two nodes. The model was inferred using* $w = 8$, $k = 4$, $\mathcal{D}_{10}^{BR}$ *and for* $T = 1000$.

$$A_0 = \begin{pmatrix} 0.5 & 0.5 & 0.0 & 0.0 \\ 0.1 & 0.4 & 0.5 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \\ 1.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}, \qquad B_0 = \begin{pmatrix} 0.80 & 0.20 & 0.0 \\ 0.10 & 0.90 & 0.0 \\ 0.0 & 0.2 & 0.80 \\ 0.2 & 0.0 & 0.80 \end{pmatrix},$$

$$A_1 = \begin{pmatrix} 0.44 & 0.0 & 0.18 & 0.37 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.53 & 0.0 & 0.43 & 0.04 \\ 0.0 & 0.71 & 0.29 & 0.0 \end{pmatrix}, \qquad B_1 = \begin{pmatrix} 0.18 & 0.45 & 0.37 \\ 0.77 & 0.23 & 0.0 \\ 0.43 & 0.53 & 0.04 \\ 0.39 & 0.05 & 0.56 \end{pmatrix}.$$

Example 5.21 *Here the source transition graph is a complete directed graph on 2 nodes with a path of length 4 joining the two nodes. The model was inferred using* $w = 7$, $k = 3$, $\mathcal{D}_{11}^{BR}$ *and for* $T = 1000$.

$$A_0 = \begin{pmatrix} 0.6 & 0.4 & 0.0 & 0.0 & 0.0 \\ 0.2 & 0.5 & 0.3 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \\ 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}, \qquad B_0 = \begin{pmatrix} 0.70 & 0.30 & 0.0 \\ 0.10 & 0.90 & 0.0 \\ 0.0 & 0.20 & 0.80 \\ 0.20 & 0.0 & 0.80 \\ 0.0 & 0.40 & 0.60 \end{pmatrix},$$

$$A_1 = \begin{pmatrix} 0.0 & 0.074 & 0.005 & 0.921 & 0.0 \\ 0.0 & 0.124 & 0.876 & 0.0 & 0.0 \\ 0.107 & 0.004 & 0.874 & 0.0 & 0.015 \\ 0.0 & 0.452 & 0.548 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.511 & 0.489 & 0.0 \end{pmatrix}, \qquad B_1 = \begin{pmatrix} 0.154 & 0.079 & 0.767 \\ 0.742 & 0.25 & 0.0 \\ 0.340 & 0.534 & 0.126 \\ 0.206 & 0.348 & 0.446 \\ 0.304 & 0.207 & 0.489 \end{pmatrix}.$$

The examples demonstrate that the inference algorithm is able to reconstruct peculiarities of the topology of the source model at least partially. The examples shown are for the minimal w and k parameters at which "approximate" reconstruction occurred, as we can conclude from experiments not shown here. The intuition about the value of $cd = w - k$, which is required to detect, say, a path of length 2 is confirmed: Namely, one needs $cd > 2$ and also $k \geq 2$.

Chapter 6

Conclusion

In this thesis we have introduced a novel algorithm for learning Hidden Markov Model (HMM) topology. The algorithm employs a Bayesian approach to control the level of generalization achieved.

We have established that the proposed inference algorithm constitutes a feasible approach to learning HMM topology. In most instances the difference in likelihood is within the range typically observed in multiple runs of the Baum-Welch algorithm starting from random initial models. Note, that the initial model prescribes the topology in the latter case. Also, we could show on examples, that the inference algorithm is able to recognize specific topological features of the source, such as deterministic paths through states.

Nevertheless, there are some problems with the method, which need to be resolved. On one hand, we observed that inferred models can have absorbing states, which is clearly undesirable, even if the probability of reaching those states is low. On the other hand, there were surprising problems with models, which were nearly Markov chains. That is, there were problems with models which have a large $MD(\cdot)$ value. Markov chains on the other hand, caused no problems in the inference process. Also, the computational effort is unacceptable for stochastic sources with medium range memory, respective interactions.

Some of the open theoretical problems — not exclusively concerning HMMs — which are of relevance for achieving this goal are:

- Extension to transient HMMs: Simply sliding a window over individual sequences and dealing with the resulting overlapping w-sequences as before would obscure the transient nature of the source process. A combination of feature and distance function able to deal with arbitrary subtrees of depth k instead of k-tails is required.
- Establishing a comprehensive distance function between HMMs: As we have shown, the probabilistic distance measure and differences in $MD(\cdot)$ capture different aspects of models. A combined distance function, taking both aspects into account, is of value with respect to distinguishing between HMMs in evaluating learning algorithms and also from a purely theoretical point of view.
- Minimality of HMMs: Similarly to existing algorithms in automata theory, an efficient and robust procedure for obtaining a minimal size model consistent with the probability distribution over all strings from $\Sigma^\star$ induced by a source model is desirable. This would

further reduce ambiguity in the evaluation on artificial test data. For the problem of *exact* identifiability and *fully* equivalent minimal representations some results are known [4,32].

- Sample size sensitive distance measure: While we have used a sample size sensitive clustering method (cf. 4.9), the distance functions we used do not take the size of a sample — in other words, the accuracy of the vectors they are comparing — into account.
- Simpler background models: To allow use of $\mathcal{D}_A^{BR}$ in situations where the $br(k)_i$ are zero due to limitations of the data set, an investigation on using $br(l)_i$, with $l << k$, instead could prove beneficial. The truncated l-tail background frequencies could, for example, be used as priors for the relative k-tail frequencies.
- Speeding up the weighted average clustering algorithm: The most time consuming step in the algorithm is the re-computation of the distances when two clusters are merged. Note, that the actual pairwise distances are irrelevant, as we only need to identify the pair of clusters at minimal distance. Hence, a iterative process using 1-tails, 2-tails etc. and the appropriate distance on the so defined relative frequencies corresponding to these tails can be used to successively prune out pairs of clusters at large distance, leaving just few candidates to select for merging.
- Investigate alternative clustering algorithms: As we have seen, using weighted average hierarchical clustering results in an improvement of performance compared to single-link clustering. It is conceivable, that more advanced clustering methods, tailored to the specific problem at hand, will yield even more favorable results and can circumvent the problems mentioned above.
- Additional priors: While the algorithm, as predicted, learns Markov chains (MC) very well, there are some artifacts to be observed for near-MC stochastic sources which might be removed by use of a prior on the *MD*-value of the inferred model.
- Tree pruning using variable window lengths: To assure reliability of the statistics used to identify nodes, subtrees can be pruned — which is effectively equivalent to reducing the window length — whenever counts $c(v_x)$ remain under some prescribed threshold. The additional computation effort in the, overall, uncritical prefix tree building phase, would speed up the time-critical distance computation and clustering phase, if the pruning leads to a reduction of nodes. Note, that this would also require a distance measure capable of dealing with relative frequencies of arbitrary collections of strings instead of all k-strings.

Bibliography

[1] A. V. AHO, R. SETHI, AND J. D. ULLMAN, *Compilers: Principles, Techniques and Tools*, Addison Wesley, Reading, MA, 1986.

[2] K. ASAI, T. YADA, AND K. ITOU, *Finding genes by hidden Markov models with a protein motif dictionary*, Proceedings, (1996).

[3] A. BAIROCH AND R. APWEILER, *The swiss-prot protein sequence data bank and its supplement trembl in 1999.*, Nucleic Acids Res, 27 (1999), pp. 49–54.

[4] V. BALASUBRAMANIAN, *Equivalence and reduction of hidden Markov models.* Master's Thesis, MIT, January 1993. A.I. Technical Report No.1370.

[5] P. BALDI AND S. BRUNAK, *Bioinformatics: The Machine Learning Approach.*, MIT Press, 1998.

[6] P. BALDI, S. BRUNAK, Y. CHAUVIN, J. ENGELBRECHT, AND A. KROGH, *Hidden Markov Models for human genes*, in Advances in Neural Information Processing Systems, J. D. Cowan, G. Tesauro, and J. Alspector, eds., vol. 6, Morgan Kaufmann Publishers, Inc., 1994, pp. 761–768.

[7] L. E. BAUM, *An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes,* in Inequalities, III, Academic Press, New York, 1972, pp. 1–8.

[8] L. E. BAUM, T. PETRIE, G. SOULES, AND N. WEISS, *A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains,* Ann. Math. Statist., 41 (1970), pp. 164–171.

[9] L. E. BAUM, T. PETRIE, G. SOULES, AND N. WEISS, *A maximization technique occuring in the statistical analysis of probabilistic functions of markov chains.,* Ann. Maths. Stats., 41 (1972), pp. 164–171. BAUM72.

[10] D. BECKER, J. HONERKAMP, J. HIRSCH, U. FRÖBE, E. SCHLATTER, AND R. GREGER, *Analysing ion channels with hidden markov models*, Plügers Archiv - European J. of Physiology, 426 (1994), pp. 328–332.

[11] J. A. BILMES, *A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models*, Tech. Rep. TR-97-021, International Computer Science Institute, Berkeley, CA, 1998.

[12] D. BLACKWELL AND L. KOOPMANS, *On the identifiability problem for functions of finite Markov chains.*, Ann. Math. Statist., 28 (1957), pp. 1011–1015.

[13] M. P. BROWN, R. HUGHEY, A. KROGH, I. S. MIAN, K. SJÖLANDER, AND D. HAUSSLER, *Using Dirichlet mixture priors to derive hidden Markov models for protein families*, in Proc. of First Int. Conf. on Intelligent Systems for Molecular Biology, L. Hunter, D. Searls, and J. Shavlik, eds., Menlo Park, CA, July 1993, AAAI/MIT Press, pp. 47–55.

[14] C. J. BURKE AND M. ROSENBLATT, *A markovian function of a markov chain*, Ann. math. stat., 29 (1958), pp. 1112–1120.

[15] G. A. CHURCHILL AND B. LAZAREVA, *Bayesian restoration of a hidden markov chain with applications to dna sequencing*, Journal of Computational Biology, 6 (1999), pp. 261–277.

[16] *Compaq extended mathematics library*. URL http://www.compaq.com/math/.

[17] A. P. DEMPSTER, N. M. LAIRD, AND D. B. RUBIN, *Maximum likelihood from incomplete data via the EM algorithm*, J. Roy. Statist. Soc. Ser. B, 39 (1977), pp. 1–38.

[18] V. DI FRANCESCO, J. GARNIER, AND P. J. MUNSON, *Protein topology recognition from secondary structure sequences: application of the hidden markov models to the alpha class proteins.*, J Mol Biol, 267 (1997), pp. 446–63.

[19] R. DURBIN, S. EDDY, A. KROGH, AND G. MITCHISON, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.*, CUP, 1998.

[20] S. R. EDDY, *Multiple alignment using hidden markov models.*, in Proc. Third Int. Symp. on Int. Sys. for Molec. Biol., pp. 114–120.

[21] S. R. EDDY, *Hidden Markov models*, Current Opinion in Structural Biology, (1996), pp. 361–365.

[22] B. EVERITT, *Cluster Analysis*, Edward Arnold, London, 1993.

[23] K. FUKUNAGA, *Introduction to Statistical Pattern Recognition*, Academic Press, Boston, MA, 1990.

[24] E. J. GILBERT, *On the identifiability problem for functions of finite Markov chains.*, Ann. Math. Statist., 30 (1959), pp. 688–697.

[25] L. GRATE, R. HUGHEY, K. KARPLUS, AND K. SJOLANDER, *Tutorial: Stochastic modelling techniques: Understanding and using hidden markov models*, Tech. Rep. UCSC-CRL-?????, University of California at Santa Cruz, Department of Computer and Information Sciences.

[26] W. N. GRUNDY, T. L. BAILEY, C. P. ELKAN, AND M. E. BAKER, *Meta-meme: Motif-based hidden Markov models of protein families*, Tech. Rep. UCSD-CS-96-514, University of California at San Diego, Department of Computer Science and Engineering, Jan. 1997.

[27] D. GUSFIELD, *Algorithms on strings, trees, and sequences*, Cambridge University Press, Cambridge, 1997. Computer science and computational biology.

[28] J. HARTUNG, *Statistik: Lehr- und Handbuch der angewandten Statistik*, R. Oldenbourg Verlag, München, 1982.

[29] A. HELLER, *On stochastic processes derived from Markov chains*, Ann. Math. Statist., 36 (1965), pp. 1286–1291.

[30] R. HERWIG, A. J. POUSTKA, C. MULLER, C. BULL, H. LEHRACH, AND J. O'BRIEN, *Large-scale clustering of cdna-fingerprinting data.*, Genome Res, 9 (1999), pp. 1093–105.

[31] D. P. HEYMAN AND M. J. SOBEL, eds., *Stochastic models*, North-Holland Publishing Co., Amsterdam, 1990.

[32] H. ITO, S. AMARI, AND K. KOBAYASHI, *Identifiability of hidden Markov information sources and their minimum degrees of freedom*, IEEE Trans. Information Theory, 38 (1992), pp. 324–333.

[33] F. JELINEK, *Statistical Methods for Speech Recognition*, The MIT Press, Cambridge, Massachusetts, 1997.

[34] B.-H. JUANG, *Maximum-likelihood estimation for mixture multivariate stochastic observations of Markov chains*, AT&T Tech. J., 64 (1985), pp. 1235–1249.

[35] B.-H. JUANG AND S. KATAGIRI, *Discriminative learning for minimum error classification*, IEEE Transactions on Signal Processing, 40 (1992), p. 3043.

[36] B. H. JUANG, S. E. LEVINSON, AND M. M. SONDHI, *Maximum-likelihood estimation for multivariate mixture observations of markov-chains*, IEEE Transactions on Information Theory, 32 (1986), pp. 307–309.

[37] B.-H. JUANG AND L. R. RABINER, *A probabilistic distance measure for hidden Markov models*, AT&T Tech. J., 64 (1985), pp. 391–408.

[38] B. H. JUANG AND L. R. RABINER, *The segmental K-means algorithm for estimating parameters of hidden markov models*, IEEE Transactions on Acoustics, Speech, and Signal Processing, 38 (1990), p. 1639.

[39] K. KARPLUS, C. BARRETT, M. CLINE, M. DIEKHANS, L. GRATE, AND R. HUGHEY, *Predicting protein structure using only sequence information*, PROTEINS: Structure, Function and Genetics Suppl., 3 (1999), pp. 121–125.

[40] K. KARPLUS, K. SJOLANDER, C. BARRET, M. CLINE, D. HAUSSLER, R. HUGHEY, L. HOLM, AND C. SANDER, *Predicting protein structure using Hidden Markov models*, Proteins: Structure, Functions and Genetics. Suppl., (1997), pp. 134–139.

[41] B. KNAB, *Erweiterungen von Hidden-Markov-Modellen zur Analyse ökonomischer Zeitreihen*, PhD thesis, Universität zu Köln, 2000.

[42] A. KROGH, *Two methods for improving performance of a HMM and their application for gene finding*, in Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology, T. Gaasterland, P. Karp, K. Karplus, C. Ouzounis, C. Sander, and A. Valencia, eds., Menlo Park, June 1997, AAAI Press, pp. 179–186.

[43] A. KROGH, M. BROWN, I. S. MIAN, K. SJOLANDER, AND D. HAUSSLER, *Hidden Markov models in computational biology: Applications to protein modelling*, Tech. Rep. UCSC-CRL-93-32, University of California at Santa Cruz, Department of Computer and Information Sciences, Aug. 1993. see Krogh94.

[44] A. KROGH, M. BROWN, I. S. MIAN, K. SJOLANDER, AND D. HAUSSLER, *Hidden markov models in computational biology: applications to protein modelling.*, J. Mol. Biol., 235 (1994), pp. 1501–1531.

[45] A. KROGH, I. S. MIAN, AND D. HAUSSLER, *A hidden Markov model that finds genes in* e. coli *DNA*, Nucleic Acids Research, 22 (1994), pp. 4768–4778.

[46] A. KROGH AND S. K. RIIS, *Prediction of BEta sheets in proteins*, in Advances in Neural Information Processing Systems, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, eds., vol. 8, The MIT Press, 1996, pp. 917–923.

[47] D. KULP, D. HAUSSLER, M. G. REESE, AND F. H. EECKMAN, *A generalized hidden Markov model for the recognition of human genes in DNA*, in Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology, D. J. States, P. Agarwal, T. Gaasterland, L. Hunter, and R. Smith, eds., Menlo Park, June12–15 1996, AAAI Press, pp. 134–142.

[48] T. LEONARD AND J. S. J. HSU, *Bayesian methods*, Cambridge University Press, Cambridge, 1999. An analysis for statisticians and interdisciplinary researchers.

[49] S. E. LEVINSON, L. R. RABINER, AND M. M. SONDHI, *An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition*, Bell System Tech. J., 62 (1983), pp. 1035–1074.

[50] L. A. LIPORACE, *Maximum likelihood estimation for multivariate observations of Markov sources*, IEEE Trans. Inform. Theory, 28 (1982), pp. 729–734.

[51] M. LOTHAIRE, *Combinatorics on words*, Cambridge University Press, Cambridge, 1997. With a foreword by Roger Lyndon and a preface by Dominique Perrin, Corrected reprint of the 1983 original, with a new preface by Perrin.

[52] R. B. LYNGSØ, C. PEDERSEN, X. SHI, AND D. J. STATES, *Comparison of hidden Markov models*, tech. rep., BRICS, 2000.

[53] R. B. LYNGSØ, C. N. S. PEDERSEN, AND H. NIELSEN, *Metrics and similarity measures for hidden markov models*, in Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology (ISMB-99), T. Lengauer, R. Schneider, P. Bork, D. Brutlag, J. Glasgow, H.-W. Mewes, and R. Zimmer, eds., Menlo Park, CA, Aug. 6–10 1999, AAAI Press, pp. 178–186.

[54] I. L. MACDONALD AND W. ZUCCHINI, *Hidden Markov and other models for discrete-valued time series*, Chapman & Hall, London, 1997.

[55] G. MCLACHLAN AND T. KRISHNAN, *The EM Algorithm and Extensions*, John Wiley & Sons, Inc, New York, 1997.

[56] C. NILUBOL, R. M. MERSEREAU, AND M. J. T. SMITH, *An improved hidden markov model classifier for sar images*, in Proc. Of the SPIE Conference on Signal Processing, Sensor Fusion, and Target Recognition VII, 1999.

[57] C. NILUBOL, F. MUJICA, R. MURENZI, R. M. MERSEREAU, AND M. J. T. SMITH, *A hidden markov model classifier for anti-tank guided missiles*, in Proc. Of the Defense Applications Signal Processing, 1999.

[58] C. NILUBOL, Q. H. PHAM, R. M. MERSEREAU, AND M. J. T. SMITH, *Translational and rotational invariant hidden markov models for automatic target recognition*, in Proc. Of the SPIE Conference on Signal Processing, Sensor Fusion, and Target Recognition VI, 1998.

[59] J. R. NORRIS, *Markov Chains*, Cambridge University Press, 1997.

[60] *Python numeric package website*. URL http://sourceforge.net/projects/numpy/.

[61] U. OHLER, *Polygramme und hidden markov modelle zur dna-sequenzanalyse*, Master's thesis, Friedrich-Alexander-Universität, 1995. Studienarbeit.

[62] T. PETRIE, *Probabilistic functions of finite state Markov chains*, Ann. Math. Statist, 40 (1969), pp. 97–115.

[63] P. A. PEVZNER, *Computational molecular biology*, MIT Press, Cambridge, MA, 2000. An algorithmic approach, A Bradford Book.

[64] *Python language website*. URL http://www.python.org.

[65] L. R. RABINER, *A tutorial on hidden Markov models and selected applications in speech recognition*, Proceedings of the IEEE, 77 (1989), pp. 257–285.

[66] L. R. RABINER, B.-H. JUANG, S. E. LEVINSON, AND M. M. SONDHI, *Some properties of continuous hidden Markov model representations*, AT&T Tech. J., 64 (1985), pp. 1251–1270.

[67] R. A. REDNER AND H. F. WALKER, *Mixture densities, maximum likelihood and the EM algorithm*, SIAM Review, 26 (1984), pp. 195–239.

[68] P. A. SCHRODT, *Pattern Recognition of International Crises using Hidden Markov Models*, in Non-linear Models and Methods in Political Science, D. Richards, ed., University of Michigan Press, Ann Arbor, MI, 1998.

[69] R. SEDGEWICK, *Algorithms in C*, Addison Wesley, Reading, MA, 1990.

[70] C. E. SHANNON, *A mathematical theory of communication*, Bell System Tech. J., 27 (1948), pp. 379–423, 623–656.

[71] K. SJOLANDER, K. KARPLUS, M. BROWN, R. HUGHEY, A. KROGH, I. S. MIAN, AND D. HAUSSLER, *Dirichlet mixtures: A method for improving detection of weak but significant sequence homology*, Tech. Rep. UCSC-CRL-?????, University of California at Santa Cruz, Department of Computer and Information Sciences.

[72] H. W. SORENSON AND D. L. ALSPACH, *Recursive Bayesian estimation using Gaussian sums*, Automatica—J. IFAC, 7 (1971), pp. 465–479.

[73] A. STOLCKE AND S. OMOHUNDRO, *Hidden Markov Model induction by bayesian model merging*, in Advances in Neural Information Processing Systems, S. J. Hanson, J. D. Cowan, and C. L. Giles, eds., vol. 5, Morgan Kaufmann, San Mateo, CA, 1993, pp. 11–18.

[74] A. STOLCKE AND S. M. OMOHUNDRO, *Best-first model merging for hidden markov model induction*, Tech. Rep. TR-94-003, International Computer Science Institute, Berkeley, CA, Jan. 1994.

[75] Y. T. TETSUSHI YADA, MISUTERU NAKAO AND K. NAKAI, *Modelling and predicting transcriptional units of escherichia coli genes using hidden markov models*, Bioinformatics, 15 (1999), pp. 987–993.

[76] S. THRUN, J. LANGFORD, AND D. FOX, *Monte carlo hidden markov models: Learning non-parametric models of partially observable stochastic processes*, in Proc. of ICML-99, 1999.

[77] T. YADA AND M. HIROSAWA, *Gene recognition in cyanobacterium genomic sequence data using the hidden Markov model*, in Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology, D. J. States, P. Agarwal, T. Gaasterland, L. Hunter, and R. Smith, eds., Menlo Park, June12–15 1996, AAAI Press, pp. 252–260.

[78] T. YADA, Y. TOTOKI, M. ISHIKAWA, K. ASAI, AND K. NAKAI, *Automatic extraction of motifs represented in the hidden Markov model from a number of DNA sequences*, Bioinformatics/Comp. Appl. BioSci, 14 (1998), pp. 317–325.

Erklärung

Ich versichere, daß ich die von mir vorgelegte Dissertation selbständig angefertigt, die benutzten Quellen und Hilfsmittel vollständig angegeben und die Stellen der Arbeit — einschließlich Tabellen, Karten und Abbildungen —, die anderen Werken im Wortlaut oder dem Sinn nach entnommen sind, in jedem Einzelfall als Entlehnung kenntlich gemacht habe; daß diese Dissertation noch keiner anderen Fakultät oder Universität zur Prüfung vorgelegen hat; daß sie — abgesehen von unten angegebenen Teilpublikationen — noch nicht veröffentlicht worden ist und daß ich eine solche Veröffentlichung vor Abschluß des Promotionsverfahrens nicht vornehmen werde. Die Bestimmungen dieser Promotionsordnung sind mir bekannt. Die von mir vorgelegte Dissertation ist von Professor Dr. R. Schrader betreut worden.

Teilpublikation:
Alexander Schliep. Learning Hidden Markov Model Topology. Poster Presentation. *Seventh International Conference on Intelligent Systems for Molecular Biology*. Heidelberg, August 1999.