Dissertation

zur Erlangung des Doktorgrades

der Fakultät für Angewandte Wissenschaften

an der Albert-Ludwigs-Universität Freiburg

Advanced Sequence Classification Techniques Applied to Online Handwriting Recognition

von

Claus Bahlmann

31. Oktober 2004

Dekan:	Prof. Jan G. Korvink
Prüfungskommission:	Prof. Dr. Luc De Raedt (Vorsitzender) Prof. Dr. Hans Burkhardt (Gutachter) Prof. Dr. Helge Ritter (Gutachter) Prof. Dr. Bernhard Nebel
Datum der Disputation:	10. Mai 2005

Acknowledgment

This thesis results from my work at the *Lehrstuhl für Mustererkennung und Bildverarbeitung (LMB)* at the University of Freiburg, Germany, under the supervision of Professor Hans Burkhardt. First of all, I want to express my deep gratitude to him for valuable discussions, suggestions and all his support.

Further, I am grateful to Professor Helge Ritter from the University of Bielefeld for his spontaneous agreement to co-referee this thesis.

It has been very inspiring and much fun to work in the environment of the LMB during all the years. Thanks to all my colleagues, who accounted to this unique atmosphere. In particular, I want to mention Bernard Haasdonk, Nikos Canterakis, Ralf Jüngling, Stefan Rahmann, Marc Schael, and Lothar Bergen, who contributed a great deal to my understanding of pattern recognition and computer vision. I am specifically obliged to Bernard Haasdonk, Nikos Canterakis, Ralf Jüngling, and Kai Simon for their thorough proofreading of the manuscript.

A big "Thanks" goes to my former B.S. and M.S. students Kai Simon, Dirk Bockhorn and Rudolph Triebel. All of you participated in the *frog on hand* project with your extraordinary creativity as well as with accurate and hard work, and the supervision of your work was both inspiring and fun.

Most of all, I want to thank the person, who inspired me in all non-scientific discussions and joined me in all the adventures in the last eight years: Maria, you know who I mean.

Finally, I also thank my parents Helga and Otto Bahlmann, who have always been there for me, also while 5000 miles away.

Princeton, September 2004 Claus Bahlmann

Zusammenfassung

Der Begriff Handschrifterkennung bezeichnet die Transformation einer Sprache, welche in räumlicher Form von graphischen Markierungen vorliegt, in seine symbolische Darstellung. Bei der *on-line* Handschrifterkennung wird dies gleichzeitig zum Schreibprozess durchgeführt. Diese Dissertation behandelt Mustererkennungsmethoden, welche zur Lösung der on-line Handschrifterkennung herangezogen werden.

On-line Handschriftdaten werden gewöhnlich als Vektorsequenz repräsentiert. Außer in der Handschrifterkennung tritt die Vektorsequenz in einer Reihe von weiteren Mustererkennungsproblemen auf, z.B. in der Spracherkennung, Genomverarbeitung, Wirtschafts- oder Medizinanwendungen oder Robotik. Für die Formalisierung und Lösung dieser Probleme ist eine direkte Modellierung der Vektorsequenz von entscheidendem Vorteil. In dieser Hinsicht lassen sich die vorgestellten Techniken auf diese Reihe von allgemeinen Anwendungen transferieren, deren Gemeinsamkeit der zugrundeliegende Datentyp der Vektorsequenz ist.

Die Dissertation führt zwei neuartige Methoden für die Klassifikation von Vektorsequenzen ein. Die eine, genannt CSDTW, fällt in die Kategorie des sogenannten generativen, die andere, genannt SVM-GDTW, in die des sogenannten diskriminativen Klassifikationsparadigma.

Das generative CSDTW (Cluster generative Statistical Dynamic Time Warping) ist ein skalierbares Klassifikationskonzept für Vektorsequenzen, welches die Ideen von Häufungspunktanalyse und statistischer Modellierung für Sequenzdaten zusammenführt. Gegenüber bisher bekannten Methoden werden diese zwei Aspekte in einem gemeinsamen Merkmalraum verbunden.

Von besonderer Bedeutung im Kontext der statistischen, generativen Herangehensweise des CSDTW (und verwandten Verfahren wie z.B. HMM) ist das Modellieren von sogenannten Richtungsdaten (d.h. Daten, die einer Richtung entsprechen, in 2D also einem Punkt auf dem Einheitskreis entsprechen; im Gegensatz dazu verteilen sich Lineardaten auf dem reellen Zahlenstrahl). Auch bei der on-line Handschrifterkennung treten Richtungsdaten auf, speziell in Form des Steigungswinkels des Schriftzugs. Diese Arbeit stellt ein Verfahren zur einheitlichen Modellierung von Richtungs- und Lineardaten innerhalb einer Wahrscheinlichkeitsdich-

tefunktion vor: die sogenannte “multivariate semi-wrapped Gauss” Wahrscheinlichkeitsdichtefunktion. Diese Verteilung zeigt — im Kontext der CSDTW Klassifikation — signifikante Verbesserungen in der Genauigkeit, Berechnungs- und Speicheranforderung im Vergleich zu bisher benutzten Ansätzen.

Als ein zusätzliches Konzept bei der Betrachtung der CSDTW Sequenzmodellierung wird ein (Un-) Ähnlichkeitmaß zwischen zwei CSDTW-Modellen vorgestellt. Solch ein Maß kann als Konvergenzkriterium im CSDTW Training, zur schnelleren Klassifikation, als Abstandsmaß im Kontext einer Häufungspunktanalyse von CSDTW Modellen oder als Optimierungskriterium für ein diskriminatives CSDTW Training verwendet werden. Ähnlich der CSDTW Klassifikation wird das vorgestellte Ähnlichkeitsmaß mit Hilfe dynamischer Programmierung berechnet und kann folglich mit geringem Programmieraufwand in eine vorhandene Bibliothek für die Sequenzklassifikation integriert werden. Das vorgestellte (Un-) Ähnlichkeitmaß basiert auf der Bayes Fehlerwahrscheinlichkeit. Dies macht es als Werkzeug zur Interpretation von Fehlklassifikationen besonders interessant. Zu diesem Zweck durchgeführte Experimente zeigen eine hohe Korrelation von ähnlichen und häufig empirisch verwechselten Klassenpaaren. Das vorgestellte Ähnlichkeitsmaß lässt sich auf häufig verwendete Spezialfälle der “Hidden Markov Modelle” (HMM) übertragen.

Als alternativer Ansatz zu der weithin verwendeten generativen Sequenzklassifikation wird eine diskriminative Methodik vorgestellt, die das “Dynamic Time Warping” (DTW) und “Support-Vektor-Machinen” (SVM) verbindet: SVM-GDTW. Diese Verbindung wird durch die Formulierung eines neuen SVM-Kerns verwirklicht, genannt “Gauss Dynamic Time Warping” (GDTW) Kern. Da dieses Verfahren ein rein diskriminatives ist, trifft es keine Annahmen über klassenbedingte Wahrscheinlichkeitsdichten. Stattdessen werden die Klassengrenzen direkt optimimiert.

Diese Arbeit vergleicht CSDTW und SVM-GDTW in Bezug auf ihren theoretischen Hintergrund, Rechen- und Speicherkomplexität und Klassifikationsgüte. Experimente auf der Basis eines Standard-Datensatzes der Handschrifterkennung zeigen überzeugende Resultate beider Verfahren. CSDTW zeichnet sich durch eine hervorragende Klassifikationsgüte und eine Skalierbarkeit auf auch schlankere Hardware aus. SVM-GDTW erzielt eine ähnlich hohe Klassifikationsgüte, erfordert jedoch mehr Hardware-Ressourcen. Es birgt als Beispiel des relativ jungen Gebietes der diskriminativen Sequenzklassifikation sehr viel Potential für zukünftige Weiterentwicklungen.

Die praktische Relevanz der entwickelten Handschrifterkennung wird anhand einer Implementierung auf einem Linux Compaq iPAQ PDA demonstriert.

Abstract

The term *handwriting recognition* (HWR) denotes the process of transforming a language, which is represented in its spatial form of graphical marks, into its symbolic representation. *Online* HWR performs this task concurrently to the writing process. The present thesis studies high-accuracy recognition methods applied to online HWR. Those methods have been implemented within the writer independent online HWR system *frog on hand* (*f*reiburg *rec*o*g*nition of *on*-line *hand*writing).

In online HWR, data are typically represented as vector sequences. In addition to HWR, vector sequence data appear in a number of additional pattern recognition problems, for instance, in speech recognition, genome processing, financial and medical applications, and robotics. For those problems, designing classifiers that directly address the data's natural representation can greatly improve the recognition accuracy, compared to a potential pre-applied transformation to vector space data. Beside introducing novel online HWR approaches, a concern of this thesis is also to develop broadly applicable pattern recognition techniques, which are generic to this bouquet of sequence data problems.

Emphasis is placed on classification. This thesis describes two complementary classification methods, one of them (*CSDTW*) falling into the so-called generative, the other one (*SVM-GDTW*) into the so-called discriminative classification category.

The generative CSDTW (cluster generative statistical dynamic time warping) is a scalable sequence classification, which aims at holistically combining sequence cluster analysis and statistical modeling. Contrary to previous approaches, these two aspects are embedded in a single feature space and use a closely related distance measure. As will be shown, this combined modeling leads to very accurate HWR results.

Particularly interesting in the context of statistical classification, like CSDTW, is the modeling of so-called *directional data* (i.e., data which corresponds to a direction, thus, in 2D is distributed on the unit circle; opposed to directional data, *linear data* is distributed along the real line). In online HWR directional data appear as a valuable feature by means of the angular pen trace direction. This thesis describes a unified modeling of directional and linear data within

one probability density function (PDF): the *multivariate semi-wrapped Gaussian* PDF. This modeling applied to CSDTW classification shows significant improvements in recognition accuracy, computational speed and memory requirements, compared to commonly employed modeling approaches.

As an additional resource for the CSDTW sequence modeling, a (dis-) similarity measure between a pair of CSDTW models is described. Such a measure can be used as a stop criterion in the iterative CSDTW training, as a speed-up in classification, a distance measure in the context of CSDTW model clustering or as an optimization criterion for a discriminative CSDTW training. Likewise to the CSDTW scoring, this (dis-) similarity computation uses dynamic programming as algorithmic framework and can thus be easily added to a given classification implementation. It is based on the Bayes probability of error, and, hence, can be utilized as a tool to interpret misclassifications. Experiments show a high correlation of similar and frequently confused class pairs.

As a complementary approach to the widely employed generative sequence modeling, a discriminative strategy of fusing dynamic time warping (DTW) and support vector machines (SVM) is developed: SVM-GDTW. This fusion is realized by a formulation of a novel SVM kernel, called the *Gaussian dynamic time warping (GDTW)* kernel. As this sequence classification approach is a pure discriminative one, it does not assume a model for the generative class conditional densities. Instead, it addresses the direct creation of class boundaries.

This thesis compares CSDTW and SVM-GDTW in terms of theoretical background, accuracy, and computational complexity. While CSDTW being the more efficient approach, SVM-GDTW holds much potential for future research as an instance of the relatively recent SVM based sequence classification.

The practical impact of the developed handwritten character recognition is demonstrated by an implementation on a Linux Compaq iPAQ PDA environment.

Contents

Notation

The general notation follows standard mathematical conventions. Specific variable names are summarized in the following.

General mathematical and statistical notation:

$\mathbb{R}$	real numbers
$\mathbf{x} \in \mathbb{R}^F$, $\mathbf{x}^T$	(feature) vector, transpose of a (feature) vector
F	dimension of feature vector space
$p(\mathbf{x})$	probability density function (PDF) of a continuous value random (vector) variable $\mathbf{x}$
$P(\mathbf{x})$	probability of a discrete value random (vector) variable $\mathbf{x}$
$\mathcal{E}[\mathbf{x}]$	expectation value of a random (vector) variable $\mathbf{x}$
$\mathbb{X} = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}\right\}$	set of observations of a random (vector) variable $\mathbf{x}$
M	number of observations of a random variable
$\boldsymbol{\mu}_\mathbf{x}$, $\boldsymbol{\Sigma}_\mathbf{x}$	mean, covariance matrix of a random (vector) variable $\mathbf{x}$
$\mathcal{N}_{\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x}}$	Gaussian (normal) PDF with mean $\boldsymbol{\mu}_\mathbf{x}$ and covariance $\boldsymbol{\Sigma}_\mathbf{x}$
$\boldsymbol{\mathcal{X}} = [\mathbf{x}_1, \ldots, \mathbf{x}_{N_x}]$	vector sequence
$l \in \{1, \ldots, L\}$	class index in an L-class classification problem
E	error rate in a classification problem
$\mathbf{C} = [C_{l'l}]_{L \times L}$	classification confusion matrix in an L-class classification problem
$P_e(1, 2)$	Bayes error in a classification problem of two classes 1 and 2
med	median
$\arg$	phase of a complex number
erf, erfc	error function, complementary error function
$\mathrm{diag}(\lambda_1, \ldots, \lambda_M)$	matrix with diagonal elements $\lambda_1, \ldots, \lambda_M$ and 0 elsewhere
$\|\mathbf{x}\|$	L_2-norm of vector $\mathbf{x}$

$\lvert\mathbf{A}\rvert$	determinant of matrix $\mathbf{A}$

CSDTW notation:

$\mathbb{P}$	set of alignment transitions
$\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_N]$	DTW alignment path
$\boldsymbol{\Phi}^* = [\boldsymbol{\phi}_1^*, \ldots, \boldsymbol{\phi}_N^*]$	DTW Viterbi path
$\alpha_j\left(\Delta\boldsymbol{\phi}\right)$	SDTW transition probability of transition $\Delta\boldsymbol{\phi}$ reaching sequence element j
$\beta_j(\mathbf{x})$	SDTW PDF of a random vector $\mathbf{x}$ at sequence element j
$\mathcal{R} = [\mathbf{R}_1, \ldots, \mathbf{R}_{N_\mathcal{R}}]$	SDTW sequence of statistical quantities $\mathbf{R}_j$
$\mathbf{R}_j = (\alpha_j, \beta_j)$	SDTW statistics associated with sequence position j
$\mathfrak{R}^\tau$	SDTW super reference pattern
d, $\tilde{d}$, $\hat{d}$, $\bar{d}$	local distance functions in (S)DTW Viterbi search
$D_{\boldsymbol{\Phi}}[d]\left(\boldsymbol{x}, \boldsymbol{y}\right)$	DTW alignment distance of two sequences $\boldsymbol{x}$ and $\boldsymbol{y}$, based on an alignment $\boldsymbol{\Phi}$ and a local distance d
$\tilde{D}_{\boldsymbol{\Phi}}[d]\left(\boldsymbol{x}, \boldsymbol{y}\right)$	$D_{\boldsymbol{\Phi}}[d]\left(\boldsymbol{x}, \boldsymbol{y}\right)$, normalized by the length of $\boldsymbol{\Phi}$
$D^*[d]\left(\boldsymbol{x}, \boldsymbol{y}\right)$	DTW Viterbi distance of two sequences $\boldsymbol{x}$ and $\boldsymbol{y}$, based on a local distance d
$\tilde{D}^*[d]\left(\boldsymbol{x}, \boldsymbol{y}\right)$	$D^*[d]\left(\boldsymbol{x}, \boldsymbol{y}\right)$, normalized by the length of $\boldsymbol{\Phi}^*$
$k \in \{1, \ldots, K_l\}$	sub-class (allograph) index of class l in a classification problem
$\mathbb{C}^{lk}$	sub-class (allograph) cluster k of class l
$\mathcal{R}^{lk}$	sub-class (allograph) CSDTW model k of class l
A^{tot}	total number of sub-class (allograph) models in a CSDTW classifier
$A^{(l)}$	number of sub-class (allograph) models for class l in a CSDTW classifier

SVM-GDTW notation:

w_i	SVM weight
S_i	binary class label $S_i \in \{+1, -1\}$
$K\left(\cdot, \cdot\right)$	general SVM kernel
$K_{\text{GDTW}}\left(\cdot, \cdot\right)$	SVM-GDTW kernel
$\varphi\left(\mathbf{x}\right)$	kernel mapping
M_S	number of support vectors in a binary SVM

Abbreviations

HWR	Handwriting recognition
OCR	Optical character recognition
PDA	Personal digital assistant
frog on hand	*f*reiburg *reco*gnition of *on*-line *hand*writing
MAP	Maximum a-posteriori
ML	Maximum likelihood
DTW	Dynamic time warping
SDTW	Statistical DTW
CSDTW	Cluster generative SDTW
HMM	Hidden Markov model
DP	Dynamic programming
SVM	Support vector machine
GDTW	Gaussian DTW
(D)DAG	(Decision) directed acyclic graph
SVM-GDTW	(Two-class) SVM with GDTW kernel
DAG-SVM-GDTW	DAG Multi-class SVM with GDTW kernel

CHAPTER 1

Introduction

This chapter introduces online handwriting recognition (HWR). It motivates the use of HWR, explains areas of application and the typical data representation. It gives a state of the art in current research and commercial products. Finally, it states and outlines the objective of this thesis.

1.1 Motivation

During recent years computers have moved continuously towards mobility. Prominent examples are laptops, personal digital assistants (PDAs) as well as a new generation of tablet PCs and smart phones. This fundamental shift has accented the necessity for alternative input methods, compared to the commonly used, predominant keyboard. Indeed, the keyboard is a fast and unambiguous computer input instrument. It is often a preferable choice in environments that have no constraints on space and weight for the equipment. However, many mobile computing applications demand a rather small and lightweight device, requirements that are not fulfilled by the comparably spacious keyboard.

A number of innovative input approaches have been studied. A comprehensive overview is given, for instance, by MacKenzie and Soukoreff [2002]. Among those, the most natural method is handwriting. However, the interpretation of handwriting is not an unambiguous mapping of user actions to a set of ASCII characters — contrary to the keyboard entry method. It requires sophisticated solutions that map a handwriting into the domain of characters. This procedure is called handwriting recognition (HWR). One goal of the present thesis is to develop high-accuracy HWR. This task has been realized within the development of the HWR project and prototype system *frog on hand*, an abbreviation for *f*reiburg *r*ec*og*nition of *on*-line

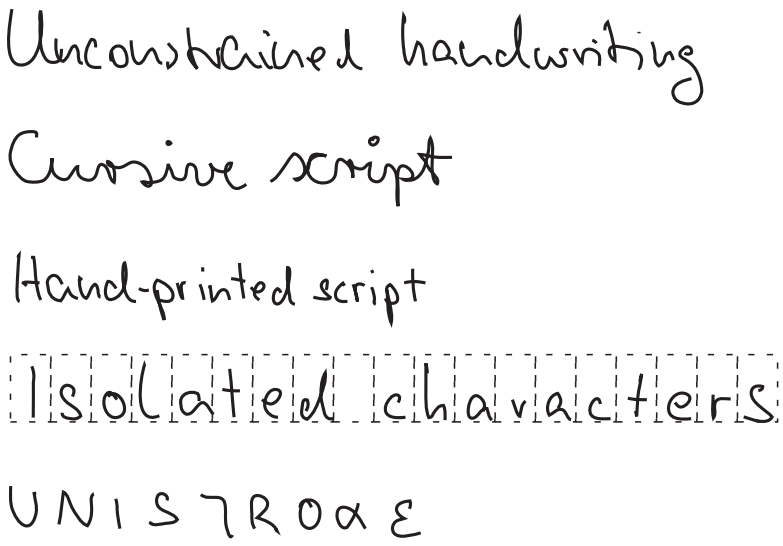


FIGURE 1.1: Types of handwriting

*hand*writing.

Many of the solutions presented are not restricted to HWR, but can be applied to a variety of additional pattern recognition problems.

A concise review of the HWR problem follows in the remainder of this chapter, with main emphasis on Latin script. Although Latin and further (e.g., Chinese, Japanese, Hangul, and Arabian) handwriting have things in common, basic differences exist that make it worthwhile analyzing handwriting with respect to the structure of each language.

Further introductory HWR literature has been published, e.g., by Plamondon and Srihari [2000], Suen et al. [2000], Plamondon et al. [1999], Guyon et al. [1997], Wakahara et al. [1992] and Tappert et al. [1990].

1.2 State of the art in handwriting recognition

Plamondon and Srihari [2000] define HWR as "the task of transforming a language represented in its spatial form of graphical marks into its symbolic representation".

The generality of this definition reflects the quality that HWR is not a single application area, but quite versatile. Indeed, there are a number of sub-domains in HWR, each of it requiring its own particular view and solution, as will be discussed subsequently.

1.2.1 Types of handwriting

Latin handwriting can appear in several contexts. In this respect, the following handwriting types exist (cf. also Figure 1.1):

Unconstrained script: This is the predominant type in everyday handwriting and constitutes the most difficult recognition problem. Without constraints, people often write with a mixture of cursive script and hand-printed characters.

One problem in unconstrained HWR is the huge variety in the writing of different people. Another extreme difficulty is the so-called *segmentation*, that is, the process of dividing an entire writing into sub-units (typically characters).

Cursive script: By definition, a cursive word is written with a single stroke. Although this type of handwriting (with a few exceptions of, e.g., "i"-dots and "t"-dashes) is taught in school, adult writers often lift the pen within a word. Thus, pure cursive script is found very rarely in practice.

Also with cursive script a recognizer has to cope with the segmentation problem.

Hand-printed script: In the context of this handwriting type, the writing still represents an entire word sequence, but succeeding characters are explicitly segmented by a pen lift.

Though, the character segmentation problem is still not solved as pen lifts can also occur within a character.

Isolated characters: In this type of handwriting, the segmentation problem is already solved, for instance by the acquisition interface and the cooperation of the writer. Graphical boxes or timeouts are common interface techniques for this purpose. Nevertheless, the recognition is still difficult due to a tremendous variability in handwriting among writers.

"Unistrokes" characters: "Unistrokes" [Goldberg and Richardson, 1993] are a rather artificial type of handwriting. "Unistrokes" characters are specifically designed symbols — one for each character — and are written within exactly one stroke. They aim to ease both problems of segmentation and writing variability by addressing the cooperation of the user.

This one-to-one correspondence between strokes and characters directly solves the character segmentation problem. Further, as each character is represented by only one representative shape, the writing variability is trimmed to a minimum. Finally, while the natural Latin alphabet of characters contains a number of ambiguities (e.g., "a" $\leftrightarrow$ "u", "u" $\leftrightarrow$ "v", "e"$\leftrightarrow$ "l", . . .), "unistrokes" characters are especially designed for an easy discrimination between them. For instance, "u" and "v" are explicitly distinguished by a special hook at the end of the "v".

The recognition difficulty decreases from the first listed type to the last. Indeed, HWR systems have a strong history in making use of this graduation in difficulty. They have aimed (and still aim) at a reasonable user satisfaction by restricting the text input to a simpler handwriting type, while simultaneously rewarding the user with recognition accuracy. In fact, the first widely

successful HWR — *Palm's Graffiti*[1] — used the simplest handwriting type of "unistrokes" characters.

1.2.2 Online vs. offline handwriting recognition

Another principal distinction is the one between online and offline HWR. The terms "online" and "offline" describe the point of time in recognition: Online HWR performs the recognition concurrently to the writing process, whereas offline HWR considerably afterwards.

This distinction has a number of consequences that further expose fundamental differences between the "online" and "offline" HWR domains. Among others, these concern data acquisition, data representation, typical application areas and the possibility of user interaction.

Data acquisition: In online HWR a typical data acquisition scenario consists of a writing pad and a pen, the combination of which is capable of capturing the trace of the pen movement. Technically, touch sensitive, electromagnetic or electrostatic sensors are often integrated into the pad. Alternatively, sensors that capture the pen dynamics can be built directly into the pen.

Offline HWR, on the other hand, acquires data typically by scanning paper documents.

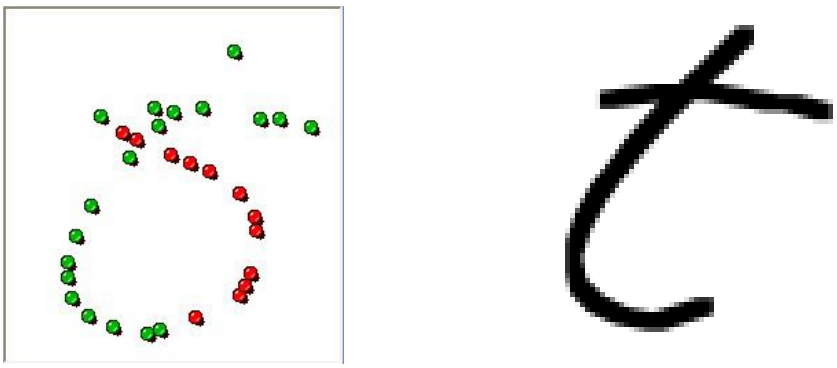
Data representation: With respect to the above described data acquisition, online HWR data is commonly a digitized representation of the pen movement. It generally contains sequential information about position, velocity, acceleration, pressure, or even angle and orientation of the pen as a function of time. Thus, it includes information about the number, order and direction of strokes, as well as the writing speed. A problem of this representation is its variety. Writings that appear similar on the "paper" can have a substantially different trace. For example, a variety of traces for the character "x" exist, or of words with "i"-dots and "t"-dashes.

Contrary to the online domain, offline data is typically represented by a 2D image matrix. Figure 1.2 illustrates the different data representation for on- and offline data.

Applications: Online HWR is used in the context of user interfaces for computing devices. In this environment, they often aim at the recognition of script. In recent years, the recognition of additional pen input like gestures, graphics [Wenyin et al., 2001], mathematical equations [Kosmala et al., 1999] or of whole page layouts [Shilman et al., 2003] have become a new focus of interest. The term *pen computing* is often used to describe an entire environment that comprises these ideas. The following list gives a number of HWR application areas and a note on the state of the art:

Input for Tablet PCs: Handwriting is the main input method for the recently introduced tablet PC. Tablet PCs are mobile laptop-sized computers with displays the

[1] `http://www.palmone.com/us/products/input/`



(a) online (b) offline

FIGURE 1.2: (a) Online handwriting data typically is a 1D sequence of coordinates. In this example green points correspond to coordinates that were sampled when the pen touched the surface. Red points were sampled while the pen was lifted. (b) Offline handwriting data typically is a 2D image matrix.

user can directly write on with a digital pen. They are designed for easy note taking in business, office, clinic and industry environments.

The accuracy of today's systems depends on many factors. Clear writing, the size of a supporting dictionary, and a large amount of representative training data are some of those. Commercial handwriting recognizers for tablet PCs are for instance included in *Microsoft Windows XP Tablet PC Edition*[2] or offered by third parties, like *Pen&Internet riteScript*[3]. Under benign conditions, word recognition rates of up to 95 % are reported.

Input for PDAs and smart phones: Character and word recognition solutions are also used in PDAs and smart phones. In addition to common HWR difficulties, the algorithms here have to cope with limited computational resources.

Commercial applications for several types of handwriting (cf. the previous section) are, for example, the already mentioned *Graffiti*, *Decuma Latin*[4] or *Microsoft Transcriber*[5]. A recognition of "unistrokes" characters achieves a recognition rate of up to 99 %. The recognition of unconstrained character and word input is still not solved to full satisfaction. A comparison of different isolated character recognition experiments from research systems is included on page 74.

Input for PCs: Handwriting can also be an appropriate input method for common personal computers. This especially concerns languages with a large alphabet, for instance, Chinese. Here, due to practical (and educational) circumstances those symbols are difficult to be entered on a keyboard.

[2] `http://www.microsoft.com/windowsxp/tabletpc/`
[3] `http://www.penandinternet.com/`
[4] `http://www.decuma.com/`
[5] `http://www.microsoft.com/windowsmobile/resources/downloads/pocketpc/transcriber.mspx`

An example for a commercial application is *Synaptics QuickStroke*.[6]

Signature verification: A variant of HWR is signature verification [Plamondon and Srihari, 2000]. Handwriting is an acknowledged and legally binding method for identification. An automatic signature verification system for biometric identification has to recognize individual characteristics of the writer and discriminate them from others. For this, many present systems rely on acceleration and pen pressure as features.

These systems are, however, several orders of magnitudes away from the very high accuracy demands that are given by, for example, financial domains.

Main applications for offline HWR are process automation for postal mail sorting, bank check processing, office automation and reading aids for blind people.

User interaction: Online HWR applications can benefit from the online acquisition environment by making use of the interaction with the user. The writer can instantly respond to the system's actions, for instance by correcting falsely recognized symbols or by choosing out of a set of possible answers, if the input was ambiguous. On the other hand, the direct user interaction imposes sharp time complexity constraints to the algorithms.

As online data describes the whole history of the writing trace it holds more information than offline data. Online recognition rates are typically higher than offline rates. It is straightforward to construct an offline image from an online trace [Manke et al., 1994]. The other way round is principally not unambiguously possible. Though, researchers have studied approaches that reconstruct the online representation out of the offline, using biophysical prior knowledge about the creation of the writing [Jäger, 1998, Lallican et al., 2000].

Notably, a merge of the traditionally independent development in on- and offline handwriting analysis methods can be observed. Online HWR approaches include "offline" image bitmaps into the feature extraction [Manke et al., 1994], others combine the results of an on- and an offline recognizer to improve overall recognition accuracy [Vinciarelli and Perrone, 2003]. Further, especially in the context of online note taking environments, image information represents a valuable source for the recognition of the document layout structure [Shilman et al., 2003].

1.3 Thesis problem statement

This thesis is concerned with the design of accurate HWR techniques. The presented methods are specifically developed to cope with the huge amount of data variations that are present in on-line handwriting. Emphasis is put on reliable recognition of isolated characters. Further,

[6] `http://www.synaptics.com/products/quickstroke.cfm`

the solutions should serve as basis for additional HWR domains, that is, hand printed words, cursive script and unconstrained handwriting (cf. Section 1.2.1).

The work of this thesis is not limited to the use within HWR, but shall also be easily transferable to general pattern recognition problems. A discussion of the specific applicable fields in pattern recognition shall be pursued in the following.

1.3.1 Thesis work in the pattern recognition context

The methods studied in this thesis are solutions to two general problems within pattern recognition: (i) the classification of sequence data and (ii) the unified statistical feature space modeling of linear and directional data.

Classification of sequence data: Many pattern recognition problems are based on *vector space data*. This means that data — or their feature representation — can be embedded in a vector space of a fixed dimension (cf. Section 2.2 for details). For vector space data, a huge collection of well-understood classification techniques exist.

However, additional problems exist, where data cannot be reasonably described by fixed dimension vectors. For those, a more complex data type may be a better representation, for example, a set, a tree, an incomplete or sparse vector — or a sequence. In this thesis, the term sequence describes a variable-size series, say $\mathcal{p}$, of observations. The observations in turn can be of arbitrary type, here they are vectors, that is $\mathcal{p} = \left[\mathbf{p}_1, \ldots, \mathbf{p}_{N_\mathcal{p}}\right]$ with $\mathbf{p}_i = (p_{i1}, \ldots, p_{iF})^T \in \mathbb{R}^F$, $i = 1, \ldots, N_\mathcal{p}$. Indeed, sequences are an ideal data type for online HWR, because the acquired data themselves are sequences (cf. Section 1.2.2).

Classifiers that are designed for vector space data cannot directly be applied to sequences (or to the other mentioned data types) in general, because they most often rely on vector space operators. For example, many classifiers use the Euclidean distance or the vector space inner product (cf. Section 2.2) — operators that are not defined for non-vector space data.

One can think of two different approaches to cope with this difficulty: One option is to neglect the respective intrinsic data structure and map the data into a vector space representation during pre-processing and feature generation. However, a drawback is the high risk of loosing substantial information through this transformation. When the loss is too severe, the alternative, more promising approach is to adopt the classifier to support the original data type directly.

This thesis aims at the second procedure by the formulation of classification approaches that are specifically designed for sequence data and their similarity relations.

The approaches are developed with application to online handwriting data. However, beside online HWR, a number of additional applications exist, where data have this sequential nature. Prominent fields are, for instance, speech recognition

[Rabiner and Juang, 1993], genome processing [Jaakkola et al., 1999], medical applications [Vullings et al., 1997] and robotics [Oates et al., 2000].

Feature space modeling of linear and directional data: Most often, patterns are represented in a feature space with *linear* variables, that is, variables that are distributed on the real line $\mathbb{R}$. For linear data, well understood methods for a statistical modeling exist. Contrary to linear data, *directional* data originate from *directions*. In two dimensions, directions may be visualized as points on the circumference of a circle and are thus inherently cyclic. Although not so well known, also for directional data techniques for a statistical modeling exist. However, no solution was so far given in case the feature space comprises *both* linear and directional data.

This thesis deals with the situation that linear and directional data occur in the context of sequences. As sequence data being a generalization of vector space data, the methods are certainly also applicable for the latter.

1.3.2 Original contributions of the thesis

The original contributions of the thesis concern solutions to the above described problems:

1. Two novel sequence classification environments are described: the generative *CSDTW* and the discriminative *SVM-GDTW* classification.

   CSDTW connects hidden Markov modeling with a scalable, hierarchical clustering. It is the first approach where clustering and hidden Markov modeling are performed in the same feature space, and the clustering gives a natural solution to the HMM topology selection and initialization. Along with CSDTW, this thesis introduces a novel (dis-) similarity measure between two generative sequence models.

   SVM-GDTW integrates the discriminative support vector machine (SVM) and the dynamic time warping (DTW) technique by the formulation of a new type of SVM kernel, called GDTW kernel. It is the first approach incorporating the powerful SVM technique into the sequence-based on-line HWR.

2. A solution for the unified statistical feature space modeling of linear and directional data is proposed by the introduction of the *approximated multivariate semi-wrapped Gaussian distribution*.

1.4 Thesis outline

The thesis is structured as follows. The following chapter begins with a review of general pattern recognition techniques. Chapter 3 explains the underlying application, that is, online HWR. It describes typical methods for data acquisition, pre-processing and feature extraction. In Chapter 4 a general and flexible generative sequence classification approach is described. It

integrates clustering and generative statistical sequence modeling, and is named *cluster generative statistical dynamic time warping (CSDTW)*. Chapter 5 discusses a solution for an issue that is often present in pattern recognition: when part of the features correspond to directions (instead of values on the real line), specific methods for a statistical modeling are required. For this, it introduces the *approximated multivariate semi-wrapped Gaussian distribution*. Chapter 6 further addresses the CSDTW classification by defining a (dis-) similarity measure for its statistical models and the related hidden Markov models. A novel discriminative sequence classification technique is introduced in Chapter 7, namely *SVM-GDTW*. This approach combines dynamic time warping (DTW) and support vector machines (SVMs) by establishing a new SVM kernel. Chapter 8 describes the implementation of one of the developed isolated character recognizers on a PDA. Chapter 9 concludes this thesis.

CHAPTER 2

Pattern recognition techniques

This chapter reviews a selection of general pattern recognition techniques. In particular, it deals with supervised and unsupervised learning methods. First, supervised learning methods where data is embedded in a vector space are reviewed. Emphasis is set on the differentiation between the generative and the discriminative classification paradigm. Second, the paradigm of supervised generative learning methods is transferred to sequence data. Dynamic time warping (DTW) and hidden Markov modeling (HMM) constitute the center of interest. Third, unsupervised learning will be reviewed by the description of an agglomerative hierarchical clustering approach. All described methods will be applied in the remainder of this thesis.

2.1 Introduction

Pattern recognition is the collective name for a number of problems, the aim of which is to assign *objects* into a set of *categories*. The objects are also referred to as *patterns,* the categories as *classes*. Humans are excellent in pattern recognition. For instance, we immediately identify faces with persons, we recognize spoken words, traffic scenes, classify documents, images and music, etc. Further, we have a distinct ability in the decoding of handwritten and printed script. In handwriting, the human's recognition system does extremely well, even in situations when a writer has taken down his notes imperfectly, illegibly, without any order and under difficult environmental conditions. The same is true in printed script, including the case of irregular, playful fonts.

With the advent of computer power in recent decades the practical importance of machine based pattern recognition systems has been raised. Attendantly, research has been consolidated and a multitude of influencing books dealing with pattern recogni-

tion published [Devijver and Kittler, 1982, Bishop, 1995, Ripley, 1996, Schürmann, 1996, Theodoridis and Koutroumbas, 1999, Duda et al., 2001].

Today, a number of pattern recognition systems are present in every day applications. Examples are: industrial computer vision systems like automated visual inspection, systems for a personal dictate and automated telephone directory enquiry using speech recognition, sequence analysis systems in bioinformatics, document and multimedia classification in the world wide web, biometric recognition for personal identification, etc. Further, HWR is used in postal and form processing and facilitates the input in electronic devices such as PDAs, smart phones and tablet PCs.

The major topic of the present thesis — online HWR — can be categorized as a pattern recognition problem. Hence, when faced to the task of the design of a HWR system, it is prosperous to take advantage of general methods developed in pattern recognition. This section describes issues that are very common and universal in pattern recognition and of considerable interest also in online HWR and the present thesis. It is beyond the scope of the thesis to give a complete illustration of these techniques. Instead, the focus lies on a sub-set of techniques that will find application in the presented work.

In particular, main topics of this section are:

- the classifier design for vector space data (Section 2.2),
- the classifier design for sequence data (Section 2.3) and
- clustering (Section 2.4).

Further, additional important pattern recognition issues like pre-processing and feature extraction are not included in this general overview, but will be addressed instead in the context of online handwriting later in Section 3.

For a further reading the above listed pattern recognition books are excellent literature. Additionally, monographs [Rabiner and Juang, 1993, Schukat-Talamazzini, 1995] and review papers [Jain et al., 2000] provide valuable information.

2.2 Classification of vector space data

Classification is part of almost every pattern recognition system. It denotes the step in the recognition process which maps the feature representation of a pattern into one, say $\hat{l}$, of a set $\{1, \ldots, L\}$ of categories. The following deliberations assume a situation where the classification takes place in a real valued vector space, that is, the feature representation $\mathbf{x} \in \mathcal{X}$ of a pattern is an element in an F-dimensional vector space of the real numbers: $\mathcal{X} = \mathbb{R}^F$. Further, it shall be assumed that a set $\mathbb{X} = \left\{\mathbb{X}^{(1)}, \ldots, \mathbb{X}^{(L)}\right\}$ of *labeled training data* is available, where $\mathbb{X}^{(l)} = \left\{\mathbf{x}^{(l,1)}, \ldots, \mathbf{x}^{(l,M_l)}\right\} \subset \mathcal{X}$ is the training set of class l.

2.2.1 The Bayes classifier

This section reviews a classifier that has a rather theoretical impact. It will turn out to be optimal in a particular sense and is called the *Bayes classifier*.

We will approach a mathematical formulation of classification and assume that available feature vectors have been generated by a double stochastic process. The first stochastic sub-process produces a discrete class decision l with respect to fixed prior class probabilities $P(l)$, $l \in \{1, \ldots, L\}$. Standard stochastic constraints $P(l) \geq 0$ and $\sum_{l=1}^{L} P(l) = 1$ are satisfied. Following, the second stochastic sub-process produces a vector observation $\mathbf{t} \in \mathbb{R}^F$, dependent on the class decision l and the respective element of a set $\{p(\mathbf{x}|1), \ldots, p(\mathbf{x}|L)\}$, $\mathbf{x} \in \mathbb{R}^F$ of *class conditional probability density functions (PDFs)*, each of which is associated with a particular class l.

In the context described above, classification is the solution to the task of revealing the unobserved class label l with only the knowledge of $\mathbf{t}$. A variety of classifiers tackles this task with help of a set $\{u_1(\mathbf{x}), \ldots, u_L(\mathbf{x})\}$ of real-valued functions called *discriminants*

$$u_l : \mathbb{R}^F \to \mathbb{R}, \quad l = 1, \ldots, L. \tag{2.1}$$

The role of the discriminants in a classifier is the following. The classification of $\mathbf{t}$ (which shall be an acronym for a test vector) is performed by choosing that class $\hat{l}$, the discriminant evaluation of which is maximal:

$$\hat{l} = \underset{l}{\operatorname{argmax}} \{u_l(\mathbf{t})\} \tag{2.2}$$

In this respect, the design of a pattern classifier consists of the formulation of $\{u_1(\mathbf{x}), \ldots, u_L(\mathbf{x})\}$.

2.2.1.1 The Bayes decision rule

Studies in decision theory specify a solution to this goal which is optimal with respect to a principle that is called *classification risk minimization*. It can be proven that $u_l(\mathbf{x})$ should be chosen as the *a-posteriori probability*

$$u_l(\mathbf{x}) = P(l|\mathbf{x}), \tag{2.3}$$

in order to achieve a minimal classification risk (which in many situations is equivalent to the probability of a classification error P_e). A connection of the a-posteriori probability to the prior probability and the PDF is given by the *Bayes rule*

$$P(l|\mathbf{x}) = \frac{P(l)\, p(\mathbf{x}|l)}{p(\mathbf{x})}. \tag{2.4}$$

The denominator in Equation (2.4) can be decomposed as

$$p(\mathbf{x}) = \sum_{l=1}^{L} p(\mathbf{x}, l) = \sum_{l=1}^{L} P(l)\, p(\mathbf{x}|l), \tag{2.5}$$

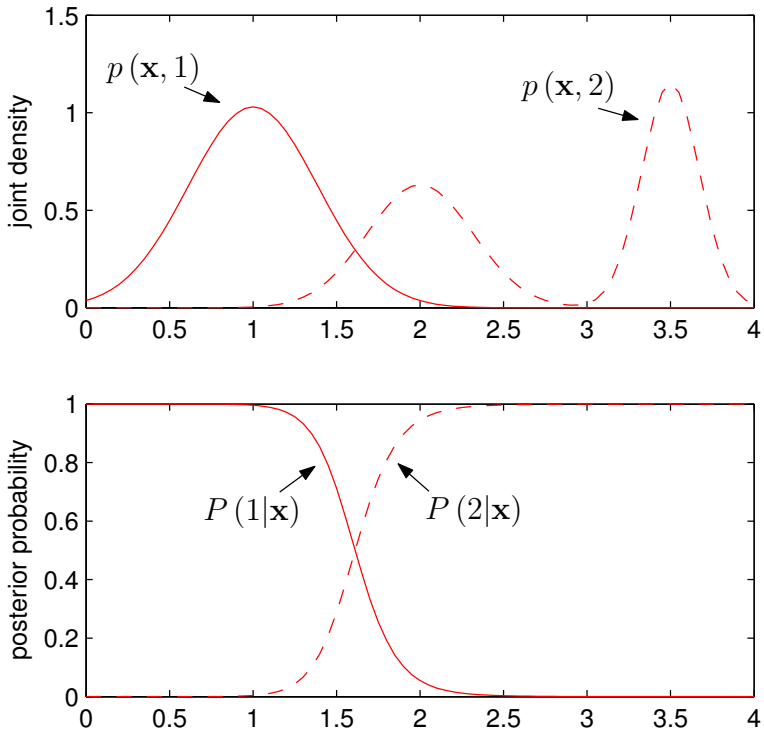


FIGURE 2.1: A two-class classification in $\mathbb{R}^1$ with the Bayes classifier. In the upper plot the densities $p(\mathbf{x}, 1) = P(1)\, p(\mathbf{x}|1)$ and $p(\mathbf{x}, 2) = P(2)\, p(\mathbf{x}|2)$ are sketched, in the lower plot the a-posteriori probabilities $P(1|\mathbf{x})$ and $P(2|\mathbf{x})$. (Figure reproduced from [Ripley, 1996].)

however, it does not depend on l and hence can ignored for the classifier's decision.

A classifier that is based on Equations (2.2) and (2.3) is optimal with respect to the classification risk minimization. Its decision rule can be simplified to

$$\hat{l} = \underset{l}{\operatorname{argmax}} \{P(l|\mathbf{x})\} = \underset{l}{\operatorname{argmax}} \{P(l)\, p(\mathbf{x}|l)\}. \tag{2.6}$$

It is usually called *maximum-a-posteriori (MAP)* or *Bayes classifier*.

In the case that prior probabilities are equal, $P(l) = 1/l$, $l = 1, \ldots, L$, the decision rule of Equation (2.6) reduces to

$$\hat{l} = \underset{l}{\operatorname{argmax}} \{p(\mathbf{x}|l)\}, \tag{2.7}$$

which is called *maximum likelihood (ML)* classification.

Details and a derivation of Equation (2.3) are presented in each of the mentioned pattern recognition books [Ripley, 1996, Duda et al., 2001, Theodoridis and Koutroumbas, 1999, Bishop, 1995, Schürmann, 1996, Devijver and Kittler, 1982].

An example for a simple Bayes classifier is illustrated in figure 2.1. The figure shows the feature space of a two-class classification problem with feature space dimension $F = 1$. In

the upper sketch of the two densities $p(\mathbf{x}, 1) = P(1)\, p(\mathbf{x}|1)$ and $p(\mathbf{x}, 2) = P(2)\, p(\mathbf{x}|2)$ are shown. The lower sketch illustrates the corresponding a-posteriori probabilities $P(1|\mathbf{x})$ and $P(2|\mathbf{x})$. According to the argumentation made previously, the Bayes classifier chooses that class, given the test pattern $\mathbf{t} \in \mathbb{R}^1$, for which $p(\mathbf{t}, l)$ or alternatively $P(l|\mathbf{t})$ is maximal. These two criteria give always the same decisions, in the example there is one decision boundary for $\mathbf{x} \approx 1.55$.

2.2.1.2 The Bayes error

An element which will be of particular interest in Chapter 6 is the already mentioned probability of a classification error P_e (also called the *Bayes probability of error*, the *Bayes error* or *Bayes risk*). It has already been mentioned that this quantity is minimized by the Bayes decision rule of Equation (2.3). However, it is worth emphasizing that P_e is not zero in general. We shall briefly illustrate the background in the context of a two-class classification problem.

Let $\mathcal{X}_l$ denote the region of the feature space where the Bayes classifier makes a decision for class l. Then, an error is made if $\mathbf{x} \in \mathcal{X}_1$ although it belongs to class 2 or if $\mathbf{x} \in \mathcal{X}_2$ although it belongs to class 1. Thus, the probability of an error $P_e(1, 2)$ for this two-class problem is

$$\begin{aligned} P_e(1,2) &= P(\mathbf{x} \in \mathcal{X}_1, 2) + P(\mathbf{x} \in \mathcal{X}_2, 1) \\ &= P(\mathbf{x} \in \mathcal{X}_1|2)\, P(2) + P(\mathbf{x} \in \mathcal{X}_2|1)\, P(1) \\ &= P(2) \int_{\mathcal{X}_1} p(\mathbf{x}|2)\, d\mathbf{x} + P(1) \int_{\mathcal{X}_2} p(\mathbf{x}|1)\, d\mathbf{x} \\ &= \int_{\mathcal{X}} \min\left\{P(1)\, p(\mathbf{x}|1), P(2)\, p(\mathbf{x}|2)\right\} d\mathbf{x} \\ &= \int_{\mathcal{X}} \min\left\{p(\mathbf{x}, 1), p(\mathbf{x}, 2)\right\} d\mathbf{x}. \end{aligned} \tag{2.8}$$

A geometrical interpretation of $P_e(1, 2)$ can be taken from Figure 2.2. Due to the integration of the minimum of the two functions $p(\mathbf{x}, 1)$ and $p(\mathbf{x}, 2)$ in Equation (2.8), $P_e(1, 2)$ corresponds to the area of overlap of $p(\mathbf{x}, 1)$ and $p(\mathbf{x}, 2)$, which is the dark shaded area in the figure.

2.2.1.3 Practical impact of the Bayes classifier

To come back to the general formulation of the classification task: at this stage it is solved in theory by Equation (2.6). However, the essential point for a real world classification problem is that the Bayes classifier is founded on the knowledge of the a-posteriori probabilities $P(l|\mathbf{x})$ or alternatively the class conditional PDFs $p(\mathbf{x}|l)$ and prior probabilities $P(l)$. Unfortunately, these are not available in the majority of problems. Mostly, only a finite set $\mathbb{X}^{(l)} = \left\{\mathbf{x}^{(l,1)}, \ldots, \mathbf{x}^{(l,M_l)}\right\}$ of labeled observations is known. These can be used to obtain at best *estimates* $\hat{P}(l|\mathbf{x})$ or alternatively $\hat{p}(\mathbf{x}|l)$ and $\hat{P}(l)$ of the quantities of interest. The quality of the estimates depends on many factors, like the structure of $\mathbb{X}$, the complexity of the problem and modeling assumptions.

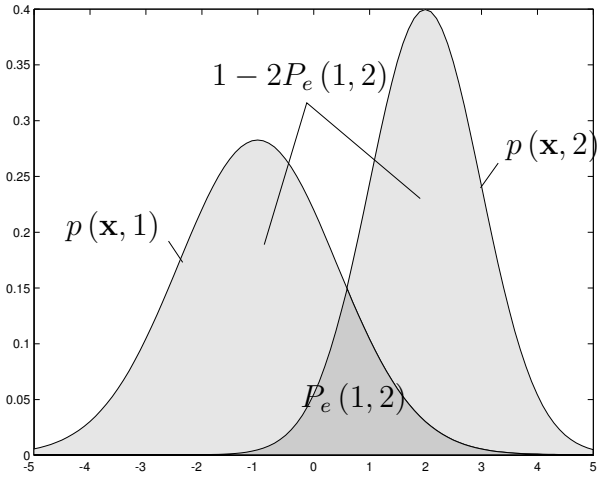


FIGURE 2.2: The Bayes probability of error $P_e\,(1,2)$ equals the area of overlap, i.e., the dark shaded area.

After all, the conclusion that can be drawn here is that in real world problems one can only get an approximation of the Bayes classifier. Approaches to this task mainly follow three distinct paradigms:

1. the *generative parametric* paradigm,
2. the *discriminative parametric* paradigm or
3. the *non-parametric* paradigm.

In the present thesis two main handwriting (and general sequence) classification approaches are studied. They address the first and second paradigm. In the following, some explanations will be given to clarify the ideas and accent the differences of the generative and the discriminative philosophies. The description of the third type, the non-parametric paradigm, will be rather concise.

2.2.2 The generative classification paradigm

The *generative classification approach* solves the discriminant function approximation from the perspective of the double stochastic, generative process, as it has been introduced in the beginning of Section 2.2.1. It assumes that the prior probabilities $P\,(l)$ and class conditional PDFs $p\,(\mathbf{x}|l)$ can be estimated reliably. In this respect, a realistic assumption about a specific function class for $p\,(\mathbf{x}|l)$ and a sufficiently sized and representative sample set $\mathbb{X}$ are required.

The Bayes rule is consulted for the transformation of $\hat{p}\,(\mathbf{x}|l)$ and $\hat{P}\,(l)$ into $\hat{P}\,(l|\mathbf{x})$. The MAP classifier becomes in the context of the probability and PDF estimates

$$\hat{l} = \underset{l}{\operatorname{argmax}} \left\{ \hat{P}\,(l)\,\hat{p}\,(\mathbf{x}|l) \right\}, \tag{2.9}$$

and the ML classifier becomes

$$\hat{l} = \underset{l}{\operatorname{argmax}} \left\{ \hat{p} \left(\mathbf{x} | l \right) \right\}. \tag{2.10}$$

A common method to estimate the PDFs is to assume a parametric function class. The most widespread assumption about the underlying function class is the Gaussian (or normal) PDF

$$p \left(\mathbf{x} | l \right) = \mathcal{N}_{\boldsymbol{\mu}_{\mathbf{x}}^{(l)}, \boldsymbol{\Sigma}_{\mathbf{x}}^{(l)}} \left(\mathbf{x} \right) = \left(\left| 2\pi \boldsymbol{\Sigma}_{\mathbf{x}}^{(l)} \right| \exp \left(\left(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}}^{(l)} \right)^T \left(\boldsymbol{\Sigma}_{\mathbf{x}}^{(l)} \right)^{-1} \left(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}}^{(l)} \right) \right) \right)^{-1/2}. \tag{2.11}$$

With this particular choice the modeling of the density falls back to the estimation of the two Gaussian parameters $\boldsymbol{\mu}_{\mathbf{x}}^{(l)}$ and $\boldsymbol{\Sigma}_{\mathbf{x}}^{(l)}$. This is often solved with help of the well-known ML estimators

$$\hat{\boldsymbol{\mu}}_{\mathbf{x}}^{(l)} = \frac{1}{M} \sum_{m=1}^{M} \mathbf{x}^{(l,m)} \tag{2.12}$$

$$\hat{\boldsymbol{\Sigma}}_{\mathbf{x}}^{(l)} = \frac{1}{M-1} \sum_{m=1}^{M} \left(\mathbf{x}^{(l,m)} - \hat{\boldsymbol{\mu}}_{\mathbf{x}}^{(l)} \right) \left(\mathbf{x}^{(l,m)} - \hat{\boldsymbol{\mu}}_{\mathbf{x}}^{(l)} \right)^T. \tag{2.13}$$

In situations where a unimodal Gaussian function class is too restrictive, a mixture of Gaussians

$$p \left(\mathbf{x} | l \right) = \sum_{k=1}^{K} c_k \mathcal{N}_{\boldsymbol{\mu}_{\mathbf{x}}^{(l,k)}, \boldsymbol{\Sigma}_{\mathbf{x}}^{(l,k)}} \left(\mathbf{x} \right) \tag{2.14}$$

is a prominent choice. We refer to literature [Rabiner and Juang, 1993, Bishop, 1995, Schukat-Talamazzini, 1995] for a more detailed description of these types of PDFs.

2.2.3 The discriminative classification paradigm

As was shown, the generative paradigm makes strict assumptions about the parametric function class of the generative source. In some situations these appear unrealistic with respect to the questioned problem. Then, a more encouraging proceeding could be to directly address a modeling of the discriminants $\{u_1, \ldots, u_L\}$ without the detour over the Bayes rule and the PDFs. This is the aim of *discriminative classifiers*.

A few presumable benefits of the discriminative paradigm can be studied by looking at Figure 2.1. Recall that the upper part illustrates the joint densities $p(\mathbf{x}, 1)$ and $p(\mathbf{x}, 2)$ and the lower part the corresponding a-posteriori probabilities $P(1|\mathbf{x})$ and $P(2|\mathbf{x})$. The first is the modeling basis of the generative philosophy, the latter of the discriminative one. We enumerate some apparent benefits of the discriminative philosophy:

1. In the example, $p(\mathbf{x}, l)$ is a more complex function than $P(l|\mathbf{x})$. This is especially valid for the interval $\mathbf{x} \geq 3$. This observation gives reason for the assumption that an approximation of $P(l|\mathbf{x})$ might need fewer parameters compared to $p(\mathbf{x}, l)$ and is thus less parameter dependent.

2. For the classification problem the function $u_l(\mathbf{x})$ is the function of interest. It is directly addressed within the discriminative approach. A detour over $p(\mathbf{x}, l)$ may introduce unnecessary inaccuracies into the classification solution.

3. An unappealing consequence of the detour is that actually the emphasis in the generative paradigm is set on the modeling of the regions of high probability and not of high discriminative impact.

4. Most generative approaches (in particular in the context of the popular ML parameter estimation) include solely the (positive) training vectors of class l in order to estimate $p(\mathbf{x}, l)$.[1] Contrary, discriminative classifiers always refine the discriminant function using both positive and negative examples.

A predominant representative of discriminative classifiers in the 1980s and 1990s has been the *multi-layer perceptron (MLP)*, another example is the *polynomial classifier* [Schürmann, 1996].

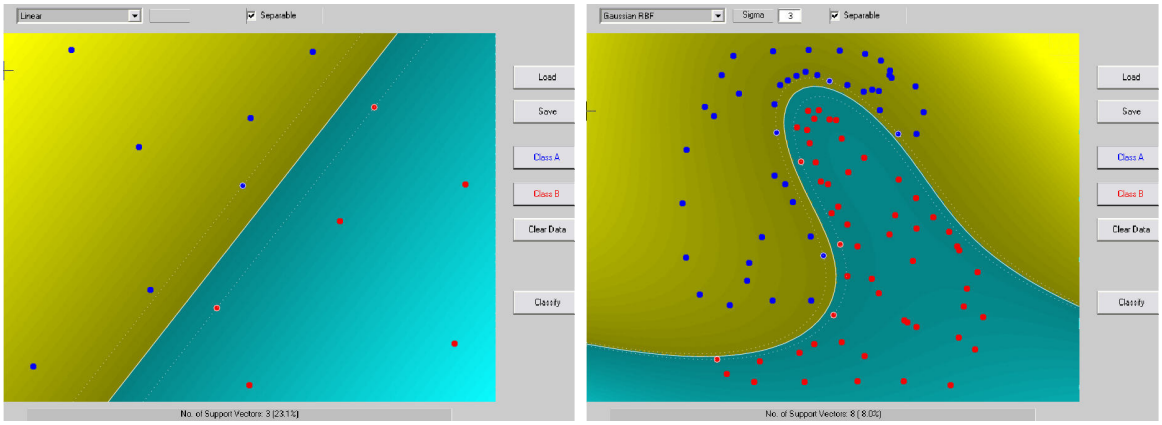
A very interesting recent development in the context of the discriminative classification paradigm is the *support vector machine (SVM)*. Many theoretical and practical arguments indicate advantages of the SVM over previous discriminative approaches. Thus, the SVM is of particular interest as an approach for discriminative handwriting classification and shall be reviewed in the following.

2.2.3.1 Support vector machine (SVM)

An SVM is basically a two-class classifier. Its concept is based on Vapnik's *structural risk minimization principle* [Vapnik, 1995]. Vapnik attributes the structural risk minimization to the maximization of a quantity that is called *margin*. In its simplest form, the margin denotes the width of the pattern-free area in the feature space around the decision boundary (cf. Figure 2.3). Given a decision boundary, the sub-set of training patterns that are closest to it is clearly determined. These training patterns are called support vectors. Only these contribute to the definition of the decision boundary and only these are used in classification. Their number is generally limited to a fraction of all training patterns.

The emphasis of this section is rather to introduce the idea of the SVM, a nomenclature and basic equations for training and classification. It abstains from a thorough treatment of the theoretical background. Excellent literature exists [Vapnik, 1995, Schölkopf, 1997, Burges, 1998, Cristianini and Shawe-Taylor, 2000, Campbell, 2000, Schölkopf and Smola, 2002] that deals with the latter issue.

[1] Although some estimators have been developed that address both positive and negative examples (e.g., maximum mutual information (MMI) or minimum classification error (MCE) [Bahl et al., 1986, Juang et al., 1997]), training for these is much more complex compared to the ML training and prohibitive in many applications.



(a) linearly separable (b) linearly non-separable

FIGURE 2.3: Principles of SVM discrimination. Part (a) shows the linearly separable case, part (b) the linearly non-separable. The discriminating region is represented by three lines. The middle line is the SVM discrimination boundary, the other two are the boundaries of the *margin*. No training vector resides in the margin. Support vectors are distinguished from the additional training vectors by white frames. (Figures produced by a *Matlab*-SVM implementation [Gunn, 1998])

In order to introduce the nomenclature, a two-class classification problem is considered. Aberrant to the previous convention, the labeled training set in this two-class problem shall here be represented as $\mathbb{X} = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}\right\}$ with its corresponding binary labels $\{S_1, \ldots, S_M\}$. The label $S_m = 1$ denotes the "positive" and $S_m = -1$ the "negative" class.

SVM classification First consider a linearly separable problem, such as the one in Figure 2.3 (a). An SVM assigns a label $\hat{S}$ to a test vector $\mathbf{t}$ by evaluating

$$f(\mathbf{t}) = \sum_i w_i S_i \left\langle \mathbf{t}, \mathbf{x}^{(i)} \right\rangle + b \quad \text{and} \quad \hat{S} = \operatorname{sign}\left(f(\mathbf{t})\right). \tag{2.15}$$

The operator $\langle \cdot, \cdot \rangle$ denotes the inner product. The *weights* w_i and the *bias* b are SVM parameters that are adopted during training (details will follow). Usually $w_i = 0$ for the majority of i and thus the summation in Equation (2.15) is limited to a subset of $\mathbb{X}$, which therefore is called the set of *support vectors*.

In the following the nonlinear case shall be described. There, per definition the vector set $\mathbb{X}$ cannot be separated by a linear formulation such as Equation (2.15). However, it may be linearly separable after being mapped to a different, usually higher dimensional (Hilbert) space, say $\mathcal{H}$, with respect to a nonlinear transformation

$$\varphi : \mathcal{X} \to \mathcal{H}. \tag{2.16}$$

When this transformation is integrated into the classification, it becomes

$$f(\mathbf{t}) = \sum_i w_i S_i \left\langle \varphi(\mathbf{t}), \varphi\left(\mathbf{x}^{(i)}\right) \right\rangle + b \quad \text{and} \quad \hat{S} = \operatorname{sign}\left(f(\mathbf{t})\right). \tag{2.17}$$

Indeed, employing a nonlinear transformation of nonlinearly separable data is a procedure which is followed as well by other classification approaches, e.g., with the polynomial classifier. The SVM approach, however, incorporates a particular understanding, which is known as the so-called *kernel trick*: Instead of explicitly evaluating the transformation φ and subsequently the term $\left\langle \varphi\left(\mathbf{t}\right), \varphi\left(\mathbf{x}^{(i)}\right)\right\rangle$ in Equation (2.17), the inner product is implicitly computed by a so-called *kernel* evaluation $K\left(\mathbf{t}, \mathbf{x}^{(i)}\right)$ that obeys

$$K\left(\mathbf{x}, \mathbf{y}\right) = \left\langle \varphi\left(\mathbf{x}\right), \varphi\left(\mathbf{y}\right)\right\rangle. \tag{2.18}$$

Hence, Equation (2.17) changes to

$$f\left(\mathbf{t}\right) = \sum_i w_i S_i K\left(\mathbf{t}, \mathbf{x}^{(i)}\right) + b \quad \text{and} \quad \hat{S} = \operatorname{sign}\left(f\left(\mathbf{t}\right)\right). \tag{2.19}$$

The short cut over $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ in place of $\varphi : \mathcal{X} \to \mathcal{H}$ and $\langle \cdot, \cdot \rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ requires only the *existence*, however not the explicit knowledge of φ and $\mathcal{H}$. Some more remarks about this issue will be pursued later in Section 7.5 on a concrete example, the Gaussian dynamic time warping kernel.

Many implementations of kernels have been proposed so far, one popular example is the *Gaussian radial basis function (RBF) kernel*

$$K\left(\mathbf{x}, \mathbf{y}\right) = \exp\left(-\gamma \left\|\mathbf{x} - \mathbf{y}\right\|^2\right). \tag{2.20}$$

others are the *sigmoid kernel*

$$K\left(\mathbf{x}, \mathbf{y}\right) = \tanh\left(\kappa \left\langle \mathbf{x}, \mathbf{y}\right\rangle - \delta\right) \tag{2.21}$$

or the *polynomial kernel*

$$K\left(\mathbf{x}, \mathbf{y}\right) = \left(\left\langle \mathbf{x}, \mathbf{y}\right\rangle + 1\right)^p. \tag{2.22}$$

SVM training For an effective use of SVM classification, the learning of the parameters w_i and b, as they appear in Equation (2.19), from the set of training observations $\left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}\right\}$ and $\{S_1, \ldots, S_M\}$ remains to be explained. The SVM framework gives a precise answer for this task. They are a unique solution to the objective of maximizing the quadratic function

$$\mathcal{L}_D = \sum_i w_i - \frac{1}{2} \sum_{i,j} w_i w_j S_i S_j K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right) \tag{2.23}$$

under the constraints

$$0 \leq w_i \leq C \quad \text{and} \quad \sum_i w_i S_i = 0, \tag{2.24}$$

with C a positive constant. The purpose of C is to weight the influence of training errors (see e.g., [Burges, 1998] for details). A solution for the w_i implies a value for b.

If the validity of Equation (2.18) can be ensured, Equations (2.23)–(2.24) define a convex quadratic optimization problem, for which the convergence towards the global optimum is guaranteed. However, obtaining this solution for real-world problems can be quite demanding and requires sophisticated optimization algorithms like *chunking*, *decomposition* or *sequential minimal optimization* [Platt, 1999, Cristianini and Shawe-Taylor, 2000].

SVM and the Bayes classifier In the explanations made above the connection of SVMs to the framework of the Bayes classifier, that is, Equations (2.1) and (2.2) is not directly obvious, since discriminants were not explicitly defined. The following argumentation shall give a hint how the SVM classification of Equation (2.19) can be identified with Equations (2.1) and (2.2).

Consider a decomposition of Equation (2.19) to

$$\begin{aligned} f(\mathbf{t}) &= \sum_{i,S_i=+1} w_i K\left(\mathbf{t}, \mathbf{x}^{(i)}\right) - \sum_{i,S_i=-1} w_i K\left(\mathbf{t}, \mathbf{x}^{(i)}\right) + b \\ &= u_{+1}(\mathbf{t}) - u_{-1}(\mathbf{t}) \end{aligned} \tag{2.25}$$

with the identifications

$$u_{+1}(\mathbf{t}) = \sum_{i,S_i=+1} w_i K\left(\mathbf{t}, \mathbf{x}^{(i)}\right) + b \tag{2.26}$$

$$u_{-1}(\mathbf{t}) = \sum_{i,S_i=-1} w_i K\left(\mathbf{t}, \mathbf{x}^{(i)}\right) \tag{2.27}$$

From this background the SVM classification rule

$$\hat{S} = \operatorname{sign}\left(f(\mathbf{t})\right) \tag{2.28}$$

is equivalent to the discriminant maximization

$$\hat{S} = \underset{S \in \{+1,-1\}}{\operatorname{argmax}} \left\{u_S(\mathbf{t})\right\}, \tag{2.29}$$

which is obviously compatible with Equation (2.2). In this context, it is worth mentioning that the SVM discriminants $u_{+1}(\mathbf{t})$ and $u_{-1}(\mathbf{t})$ do not have the semantic of the a-posteriori-probabilities $p(+1|\mathbf{t})$ and $p(-1|\mathbf{t})$, as the Bayes classifier suggests. This can indeed seen as an unfortunate situation in some cases. Also, the context of HWR on a word-level basis (in Section 7.7) it will turn out as a disadvantage.

Apart from this disadvantage, SVMs have achieved excellent recognition results in various pattern recognition applications [Cristianini and Shawe-Taylor, 2000]. Also in offline optical character recognition (OCR) they have been shown to be comparable or even superior to the generative approaches or MLPs [DeCoste and Schölkopf, 2002]. They generalize very well, as the implicit regularization of the classifier's complexity avoids overfitting. Some further properties are commonly seen as reasons for the success of SVMs in real-world problems: the optimality of the training result is guaranteed and little a-priori knowledge is required.

2.2.3.2 Multi-class SVM

The present explanations assumed a two-class classification problem. Extensions of the two-class to multi-class situations are suggested and compared in several publications [Burges, 1998, Platt et al., 2000, Hsu and Lin, 2001]. Three solution are most commonly used:

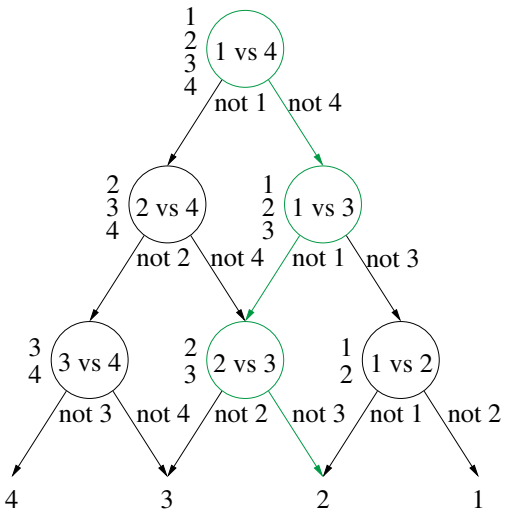


FIGURE 2.4: A DAG-SVM for an L (=4)-class classification problem. In the graph each node corresponds to a two-class SVM, indicated by the class index pair within the node. A classification of a test vector corresponds to a path through this graph. At every node one class is excluded from the set of class hypotheses by the respective two-class classifier's decision. The figure denotes the remaining classes by the list of class indices left to the node. As the path traverses $L-1$ nodes, exactly one of the L classes remains in the set of hypotheses as the final solution. (Figure reproduced from [Platt et al., 2000].)

One-against-all: For an L-class problem L two-class SVMs are trained. The positive examples of the l-th SVM are the observations of class l, the negative ones the remaining examples of *all* other classes. In classification the class with maximal $f(\mathbf{t})$ wins.

Max-wins: For an L-class problem $L \cdot (L-1)/2$ two-class SVMs are trained, one for each class pair. In classification all SVMs are evaluated and the class with the maximal number of two-class hits wins.

DAG-SVM: Training is equal to the "max-wins" approach. During classification $L-1$ two-class SVM evaluations are combined using a decision directed acyclic graph (DDAG) topology. Figure 2.4 explains details of this algorithm.

2.2.4 The non-parametric classification paradigm

The last section has argued that discriminative classifiers do not assume a particular structural constraint for the generative source. However, also in the discriminative paradigm modeling assumptions are made. This generally concerns a particular function class and parameters of the discriminants. E.g., for the SVM this is the case with the selection of a particular kernel function and the error penalty C. If also this procedure is too daring in the context of the underlying problem, a third paradigm can be pursued, the paradigm of *non-parametric classification*.

Non-parametric classifiers address the approximation of the PDFs $p(\mathbf{x}|l)$ on the basis of local accumulations in the training set $\mathbb{X}^{(l)} = \left\{\mathbf{x}^{(l,1)}, \ldots, \mathbf{x}^{(l,M_l)}\right\}$. Unlike generative classifiers they do not aim to map $\mathbb{X}^{(l)}$ to any parameterization, but they assume that the whole training set is present during classification. Given a "reasonable" dissimilarity function $D(\mathbf{t}, \mathbf{x})$ of two vector space elements $\mathbf{t}$ and $\mathbf{x}$, particular classification rules which are based on the dissimilarity of the test vector and all training patterns, are employed. A popular choice is the k-nearest-neighbor rule. For the special case of $k = 1$ this one reduces to the nearest-neighbor rule and has the form

$$\widehat{l} = \underset{l \in \{1,\ldots,L\}}{\operatorname{argmin}} \min_{m \in \{1,\ldots,M_l\}} \left\{ D\left(\mathbf{t}, \mathbf{x}^{(l,m)}\right) \right\}. \tag{2.30}$$

One obvious problem of the non-parametric paradigm in real world applications is its time and memory complexity. This unfortunate fact is due to the required omni-presence of the whole training set, as noted.

2.3 Classification of sequence data

The prerequisite of the classifier descriptions made in the previous section was that the feature representation of all patterns can mathematically be treated as an element in a (shared) vector space. In particular, this includes a common dimension F of all vectors. It has been argued in Section 1.3.1 that such an assumption does not cover all practical problems in pattern recognition. Online HWR belongs to the category of problems, where data are usually vector *sequences* of different lengths. Such problems require a special treatment.

A driving source for the development of sequence classifiers has been the research in speech recognition. Starting in the 1960s a number of approaches have been developed. Historically, speech recognition systems were firstly implemented using template based methods. In particular, this was solved by the use of an "elastic match" of two sequences and the dynamic time warping (DTW) algorithm. Later, the classifiers were formulated from a statistical perspective, which gave rise to the huge success of hidden Markov models (HMMs).

However, in the classification of sequence data the discriminative idea was not so thoroughly developed as in the case of vector data. Discriminative approaches are limited to a discriminative parameter estimation of generative models, e.g. the maximum mutual information (MMI) [Bahl et al., 1986] or minimum classification error (MCE) [Juang et al., 1997] training criteria that are especially utilized with HMMs. Indeed, one contribution of the present thesis aims to close this gap by the formulation of a sequence compatible SVM (Chapter 7).

Nevertheless, the formalisms of DTW and HMMs are very fundamental ones in sequence modeling, are also utilized in the present work and shall be described in the following.

2.3.1 Dynamic time warping (DTW)

DTW is a concept that allows an elastic match of two sequences. A definition of a distance measure is based on this match. Details about DTW are described in literature, e.g., in the

textbook of Rabiner and Juang [1993, Chapter 4.7]. Here we want to review the basic concepts.

Suppose that the following is given:

- two vector sequences $\boldsymbol{t} = [\mathbf{t}_1, \ldots, \mathbf{t}_{N_t}]$ and $\boldsymbol{r} = [\mathbf{r}_1, \ldots, \mathbf{r}_{N_r}]$ with $\mathbf{t}_i, \mathbf{r}_j \in \mathbb{R}^F$; one can think of $\boldsymbol{t}$ being a "test" and $\boldsymbol{r}$ being a "reference" sequence
- a so-called *alignment* (or *warping*) *path* $\boldsymbol{\Phi} = [\boldsymbol{\phi}(1), \ldots, \boldsymbol{\phi}(N)]$ with $\boldsymbol{\phi} = (\phi_t, \phi_r) : \{1, \ldots, N\} \to \{1, \ldots, N_t\} \times \{1, \ldots, N_r\}$, the purpose of which is to define an alignment of corresponding regions in $\boldsymbol{t}$ and $\boldsymbol{r}$
- a distance function $d : \mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}$ of two sequence elements; d shall be called *local* distance

Then, the *alignment distance* $D_{\boldsymbol{\Phi}}[d](\boldsymbol{t}, \boldsymbol{r})$ is defined as the sum of all distances between the elements of $\boldsymbol{t}$ and $\boldsymbol{r}$ with respect to d and the particular alignment path $\boldsymbol{\Phi}$

$$D_{\boldsymbol{\Phi}}[d](\boldsymbol{t}, \boldsymbol{r}) \;=\; \sum_{n=1}^{N} d\left(\mathbf{t}_{\phi_t(n)}, \mathbf{r}_{\phi_r(n)}\right). \tag{2.31}$$

Note that the parameterization of $D_{\boldsymbol{\Phi}}$ by d is quite unusual in literature, however it will prove to be practical in the remainder of this thesis.

Further, the *DTW (Viterbi) distance* $D^*[d](\boldsymbol{t}, \boldsymbol{r})$ is defined as the alignment distance according to the *Viterbi path* $\boldsymbol{\Phi}^*$. The Viterbi path is the *optimal* alignment path in the sense that it minimizes $D_{\boldsymbol{\Phi}}[d](\boldsymbol{t}, \boldsymbol{r})$:

$$D^*[d](\boldsymbol{t}, \boldsymbol{r}) = D_{\boldsymbol{\Phi}^*}[d](\boldsymbol{t}, \boldsymbol{r}) = \min_{\boldsymbol{\Phi}} \left\{D_{\boldsymbol{\Phi}}[d](\boldsymbol{t}, \boldsymbol{r})\right\}. \tag{2.32}$$

In situations when emphasis is set on a specific Viterbi path that comes from the alignment of a particular $\boldsymbol{t}$ and $\boldsymbol{r}$, the notation $\boldsymbol{\Phi}^*_{\boldsymbol{t},\boldsymbol{r}}$ instead of $\boldsymbol{\Phi}^*$ will be used.

It is convenient to model $\boldsymbol{\Phi}$ as a sequence of *transitions* from a transition set $\mathbb{P}$, i.e., $\Delta\boldsymbol{\phi}(n) = \boldsymbol{\phi}(n) - \boldsymbol{\phi}(n-1) \in \mathbb{P}$, $n = 2, \ldots, N$. We use the ones which are known as *Sakoe-Chiba transitions* in literature [Rabiner and Juang, 1993, Chapter 4.7]. These only allow forward steps of size 1 in $\boldsymbol{t}$, $\boldsymbol{r}$ or in both of them, i.e.,

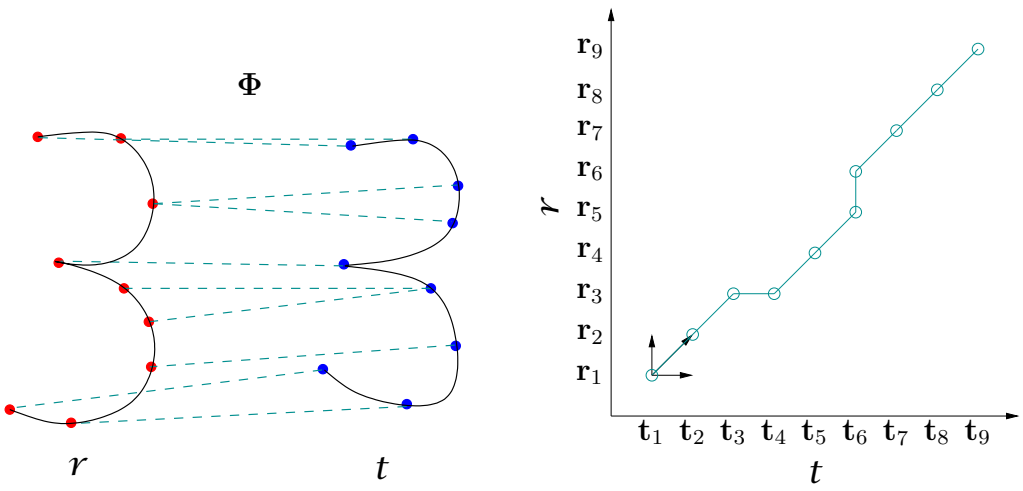
$$\mathbb{P} = \{(1,0), (0,1), (1,1)\}. \tag{2.33}$$

The alignment paths are constrained to include the endpoints of both $\boldsymbol{t}$ and $\boldsymbol{r}$, i.e., $\boldsymbol{\phi}(1) = (1,1)$ and $\boldsymbol{\phi}(N) = (N_t, N_r)$.

In literature often the Euclidean distance or its square

$$d(\mathbf{t}_i, \mathbf{r}_j) = \|\mathbf{t}_i - \mathbf{r}_j\|^2 \tag{2.34}$$

is used for the local distance of the sequence elements.



(a) (b)

FIGURE 2.5: DTW aims to align corresponding sample points in two vector sequences. Found identifications are shown by the dashed lines in part (a). Note the nonlinear alignment with the 3rd, 5th and 6th sample points in r. The correspondences found can also be illustrated through an entry in a matrix, shown by part (b). In this respect, an entry at element (i, j) means that sample i of sequence t is identified with sample j of sequence r. (Illustration of part (a) reproduced from [Tappert et al., 1990].)

Sometimes it turns out that a modification of the DTW distance

$$\tilde{D}^*[d]\,(t, r) = \frac{1}{N^*} D^*[d]\,(t, r)\,, \tag{2.35}$$

which is normalized by the Viterbi path length N^*, is more successful in classification. This is especially true if sequences that represent similar patterns have largely varying lengths. In online handwriting data such a situation occurs when data comes from a variety of sensors that have different sampling rates.

An example of the DTW (Viterbi) distance computation is given in Figure 2.5. It shows an elastic match of two patterns that both represent the digit "3".

There exist efficient solutions for the minimization of Equation (2.32), as will be pointed out in Section 2.3.3.

It is straightforward to define a (simple) sequence classifier based on the DTW distance. The nearest-neighbor rule of Equation (2.30) in combination with Equation (2.32) is a typical and historically early employed example for a non-parametric sequence classifier. However, the drawbacks mentioned for this paradigm (time and memory complexity) especially count for the rather expensive DTW distance computation.

2.3.2 Hidden Markov modeling (HMM)

Hidden Markov models (HMMs) have largely contributed to the success of many speech and HWR systems. Also in other sequence data based applications like bioinformatics, robotics,

etc., they have been widely used.

Due to a large existing literature that deals with HMMs the review given in the following is rather concise. The nomenclature is taken from a number of publications [Rabiner and Juang, 1993, 1986, Schukat-Talamazzini, 1995, Rabiner, 1989, Elms, 1996, Jäger, 1998, Jelinek, 1998]. In some situations a preparation for subsequent sections is intended and thus it was deviated from the notion of the mentioned publications.

First, *Markov chains* will be introduced. *HMMs* will follow.

2.3.2.1 Markov chains

Consider a finite set

$$\mathbb{S} = \{s_1, \ldots, s_{N_S}\} \tag{2.36}$$

of *states* and a discrete stochastic process

$$\boldsymbol{q} = [q_1, \ldots, q_N]\,,\ q_n \in \mathbb{S}, \tag{2.37}$$

the elements of which are taken from $\mathbb{S}$. At every time step n the stochastic process produces an observable state.

A *state transition* is defined as a successive tuple (q_n, q_{n+1}) of two states. It shall be assumed that the transition probabilities of the stochastic process are invariant in n and only depend on the previous state, more formally expressed by the equation

$$P\left(q_{n+1}|q_n, \ldots, q_1\right) = P\left(q_{n+1}|q_n\right). \tag{2.38}$$

With this property, the transition probabilities can be expressed as an $N_S \times N_S$ matrix

$$\mathbf{A} = [a_{ij}]_{N_S \times N_S} \text{ with } a_{ij} = P\left(q_{n+1} = s_j|q_n = s_i\right). \tag{2.39}$$

Its elements obey the standard stochastic constraints $a_{ij} \geq 0$ and $\sum_j a_{ij} = 1$.

Further, in order to provide a well defined stochastic initialization of the process, the N_S dimensional vector

$$\boldsymbol{\pi} = (\pi_1, \ldots, \pi_{N_s})^T \text{ with } \pi_i = P\left(q_1 = s_i\right) \text{ and } \sum_{i=1}^{N_S} \pi_i = 1 \tag{2.40}$$

summarizes the state probabilities at the initial time step $n = 1$.

Equation (2.38) is called *first order Markov property* and the discrete process of Equation (2.37) that satisfies the first order Markov property is called *first order Markov chain* or *Markov model*.

A first order Markov chain is entirely defined by the tuple

$$\kappa = (\boldsymbol{\pi}, \mathbf{A})\,. \tag{2.41}$$

The choice of the term "first-order" implies the presence of "second-", "third-", etc. order Markov chains. Indeed, the order number indicates the memory of the process, i.e., the number of previous states on that a transition depends with respect to Equation (2.38). However, Markov chains of a higher order than one are not considered in the present work.

2.3.2.2 Continuous density hidden Markov model

A more sophisticated model than the Markov chain is the HMM. In this section the focus shall address *continuous density* HMMs. In addition to these constructs, *semi-continuous density* and *discrete* HMMs exist, among others. The principle concept of these types, however, is very similar to the continuous case. Readers with particular interest in those models are referred to literature [Rabiner and Juang, 1993, Schukat-Talamazzini, 1995].

For the formulation of (continuous density) HMMs a second stochastic process is introduced on top of the Markov chain. Let this stochastic process produce, dependent of the currently occupied state s_i, a random vector out of $\mathbb{R}^F$. Hidden Markov modeling assumes that a hypothetic observer of the process solely perceives a sequence $\boldsymbol{t} = [\mathbf{t}_1, \ldots, \mathbf{t}_N]$, $\mathbf{t}_i \in \mathbb{R}^F$ of these random vectors and not the state sequence $\boldsymbol{q} = [q_1, \ldots, q_N]$ directly. It is further assumed that this second stochastic process is independent of the $\boldsymbol{t}$ and $\boldsymbol{q}$ history, i.e., obeys the property

$$p\left(\mathbf{t}_n | \mathbf{t}_{n-1}, \ldots, \mathbf{t}_1, q_n, \ldots, q_1\right) = p\left(\mathbf{t}_n | q_n\right). \tag{2.42}$$

Its characteristic can thus be summarized by a sequence

$$\boldsymbol{b} = [b_1, \ldots, b_{N_S}] \tag{2.43}$$

of PDFs, each element b_j corresponding to a PDF associated with state s_j

$$b_j\left(\mathbf{x}\right) = p\left(\mathbf{x} | q_n = s_j\right), \mathbf{x} \in \mathbb{R}^F, j = 1, \ldots, N_S \tag{2.44}$$

The complete double stochastic process is entirely defined by the tuple

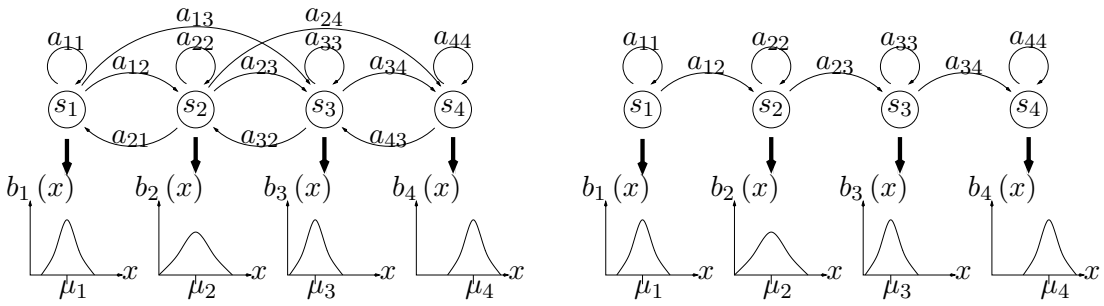
$$\lambda = \left(\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{b}\right) \tag{2.45}$$

and called *continuous density hidden Markov model*.

If all elements of $\mathbf{A}$ are non-zero, an HMM is called fully connected or *ergodic*. An ergodic HMM is illustrated in Figure 2.6 (a). For speech and HWR applications ergodic models have been shown to be too complex. A number of HMM specializations have thus been studied. These are usually *not* fully connected and include a preferential transition direction, corresponding to a rather sparse occupation of the state transition matrix $\mathbf{A}$. A typical instance is the *linear HMM*, sketched in Figure 2.6 (b). For linear HMMs only the transitions a_{jj} and $a_{j(j+1)}$ (for $j = 1, \ldots, N_S - 1$) and $a_{N_S N_S}$ have non-zero values.

2.3.2.3 The three basic problems with HMMs

So far, the notation of HMMs has been introduced. An application of the HMM concept requires some further recipes for common problems associated with HMMs. Literature distinguishes between three basic problems. We enumerate the problems and briefly sketch a solution. Detailed information is given in literature [Rabiner and Juang, 1993, Schukat-Talamazzini, 1995].



(a) ergodic HMM (b) linear HMM

FIGURE 2.6: Examples of HMMs. Part (a) shows an *ergodic* HMM. Here, every state is possibly connected with every other state. For speech and handwriting recognition applications, the ergodic topology is mostly too complex. Usually, much more sparse HMMs like the *linear* one shown in part (b) are employed.

1. *Probability scoring:* Given an observation sequence $\boldsymbol{t}$, what is the probability score $p(\boldsymbol{t}|\lambda)$ that $\boldsymbol{t}$ has been produced by an HMM λ? An answer for this problem is given by the *forward-backward algorithm* [Rabiner and Juang, 1993, Section 6.4.1].

2. *Uncovering the most probable hidden state sequence:* Given an observation sequence $\boldsymbol{t}$, what is the most likely state sequence $\boldsymbol{q}^*$ that an HMM λ may have produced? The Viterbi algorithm [Rabiner and Juang, 1993, Section 6.4.2] solves this issue. It will be described in Section 2.3.3.1. As the nomenclature indicates, $\boldsymbol{q}^*$ is the HMM counterpart to the DTW Viterbi path $\boldsymbol{\Phi}^*$.
   The so-called *Viterbi score* $p\left(\boldsymbol{t}, \boldsymbol{q}^*|\lambda\right)$ scores a test pattern $\boldsymbol{t}$ solely with respect to the most probable state sequence $\boldsymbol{q}^*$. In literature, $p\left(\boldsymbol{t}, \boldsymbol{q}^*|\lambda\right)$ is often named a "decision-directed" or "state-optimized" variant of $p(\boldsymbol{t}|\lambda)$. It has been shown that in many real world applications $p\left(\boldsymbol{t}, \boldsymbol{q}^*|\lambda\right)$ is highly correlated to the probability score $p(\boldsymbol{t}|\lambda)$ [Merhav and Ephraim, 1991]. Since it benefits from a lower computation complexity, many HMM implementations use it as a substitute for $p(\boldsymbol{t}|\lambda)$.

3. *Parameter estimation:* Given a set of observed sequences $\mathbb{X} = \left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(M)}\right\}$, what is the most likely HMM λ, which has been the source of these observations? This problem is often solved by the Baum-Welch algorithm [Rabiner and Juang, 1993, Section 6.4.3], also known as EM-algorithm. This algorithm iteratively estimates model parameters based on the ML criterion (which means that the objective $\mathcal{L}(\lambda) = \sum_{m=1}^{M} \ln P\left(\boldsymbol{x}^{(m)}|\lambda\right)$ is maximized).
   Also for parameter estimation, a decision-directed variant exists. This variant is solely based on the Viterbi state sequences $\left(\boldsymbol{q}^*\right)^{(m)}$. It maximizes $\mathcal{L}^*(\lambda) = \prod_{m=1}^{M} P\left(\boldsymbol{x}^{(m)}, \left(\boldsymbol{q}^*\right)^{(m)}|\lambda\right)$ instead of $\mathcal{L}(\lambda)$, again with the advantage of a lower complexity. This training method is known as *Viterbi training* (or *segmental K-means algorithm*) [Schukat-Talamazzini, 1995, Juang and Rabiner, 1990]) and will be described in

more detail in the context of CSDTW (cf. Chapter 4).

2.3.2.4 HMM classification and training

With a solution to the basic three problems a generative sequence classification framework can be stated.

For training, HMM parameters $\lambda_l = \left(\boldsymbol{\pi}^{(l)}, \mathbf{A}^{(l)}, \boldsymbol{\mathcal{B}}^{(l)}\right)$ are estimated from the training set $\mathbb{X}^{(l)} = \left\{\boldsymbol{X}^{(l,1)}, \ldots, \boldsymbol{X}^{(l,M_l)}\right\}$ of observations for each class l, employing either the Baum-Welch or the Viterbi training (problem 3).

For classification, the probability score $p\left(\boldsymbol{t}|\lambda_l\right)$ or substitutively the Viterbi score $p\left(\boldsymbol{t}, \boldsymbol{q}^*|\lambda\right)$ construct an ML classifier according to Equation (2.7),

$$\hat{l} = \underset{l}{\operatorname{argmax}} \left\{p\left(\boldsymbol{t}|\lambda_l\right)\right\} \tag{2.46}$$

or

$$\hat{l} = \underset{l}{\operatorname{argmax}} \left\{p\left(\boldsymbol{t}, \boldsymbol{q}^*|\lambda_l\right)\right\}, \tag{2.47}$$

respectively.

Of course, the MAP classifier can also be employed, however not always is more accurate in real world problems.

2.3.2.5 HMM — an example

Following this rather theoretical review of HMM a concrete example with an explicit link to online HWR might be illustrative. Figure 2.7 provides an example and also shows a connection to the DTW framework, in particular to Figure 2.5. In figure 2.7(a), the left hand side shows the state and transition topology. The reader can verify that the HMM chosen is a linear one. Each of the five states $s_1, \ldots, s_5$ points to a PDF b_j which in the example is a unimodal Gaussian in $\mathbb{R}^2$ (as shown by the pseudo-color plot). To the right a test sequence $\boldsymbol{t}$ is plotted. A dashed line between a sequence element $\mathbf{t}_i$ and the HMM states s_j means that $\mathbf{t}_i$ is produced by the PDF b_j with respect to the Viterbi state sequence $\boldsymbol{q}^*$. Figure 2.7(b) shows $\boldsymbol{q}^*$ in a matrix (or "trellis") representation.

When comparing this illustration with Figure 2.5, common characteristics are apparent. The reference samples $\mathbf{r}_j$ in the DTW context have counterparts in the HMM states s_j and their attached PDFs b_j. The DTW transitions $\Delta\boldsymbol{\phi}\left(n+1\right)$ have correspondences in the HMM transitions $\left(q_n, q_{n+1}\right)$. However, the transition set of the HMM is more restricted than the DTW Sakoe-Chiba transitions of Equation (2.33) in the sense that it lacks a transition $(0,1)$, as can be seen in a comparison of both figures. A further unifying study of DTW and HMM will follow in Section 4.3.1.4.

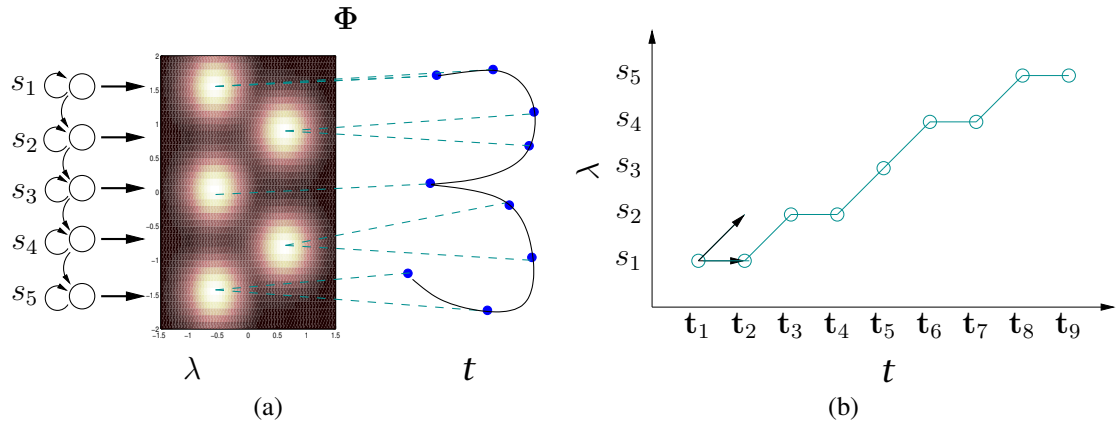


FIGURE 2.7: An example for an HMM:(a) To the left the state and transition topology of a linear HMM is shown. Each of the five states $s_1, \ldots, s_5$ points to a PDF b_j which in the example is a unimodal Gaussian in $\mathbb{R}^2$. To the right a test sequence t is plotted. A dashed line between a sequence element $\mathbf{t}_i$ and the HMM states s_j means that $\mathbf{t}_i$ is produced by the PDF b_j with respect to the Viterbi state sequence q^*. 2.7(b) q^* is shown in a matrix (or "trellis") representation. A comparison of this figure with Figure 2.5 reveals basic correspondences between HMM and DTW.

2.3.3 Search strategies for complexity reduction

When solving Equation (2.32) in the DTW or the forward-backward algorithm in the HMM context, a naive optimization procedure is prohibitive, as the complexity increases exponentially with the pattern lengths. Therefore, it is common and advantageous to apply techniques like dynamic programming and beam search in order to achieve a complexity reduction. The following explanations orientate at the DTW framework. However, in the hidden Markov modeling the situation is very similar and a transfer of the explanations to this concept is straightforward [Rabiner and Juang, 1993].

2.3.3.1 Dynamic programming

A central element in dynamic programming (DP) is the Viterbi matrix, i.e., the matrix $\mathbf{D}^* = \left[D^*_{ij}\right]_{M_t \times M_r}$ of the prefixes' Viterbi distances $D^*_{ij} = D^*[d]\left([\mathbf{t}_1, \ldots, \mathbf{t}_i], [\mathbf{r}_1, \ldots, \mathbf{r}_j]\right)$. DP's basic idea is to recursively develop $\mathbf{D}^*$ in order to compute $D^*[d](t, r)$ (cf. Figure 2.8 (a)), exploiting Bellmann's *optimality principle* [Bellmann, 1957]. In the DTW context this principle allows the computation of $\mathbf{D}^*$ with help of the following recursion:

$$D^*_{ij} = d(\mathbf{t}_i, \mathbf{r}_j) + \min \left\{ \begin{array}{c} D^*_{(i-1)j} \\ D^*_{i(j-1)} \\ D^*_{(i-1)(j-1)} \end{array} \right\}, \; i = 1, \ldots, N_t, \; j = 1, \ldots, N_r \tag{2.48}$$

and the initializations

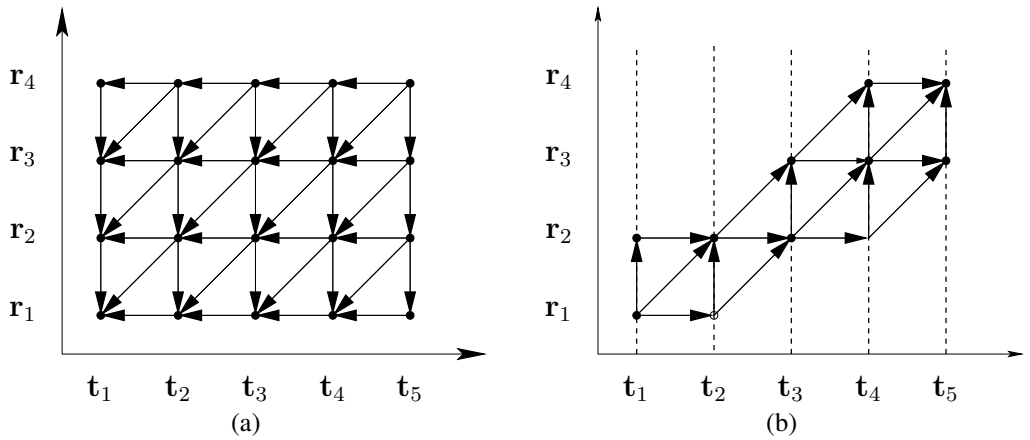


FIGURE 2.8: (a) Dynamic programming: The Viterbi matrix $\mathbf{D}^*$ is computed by the recursion scheme of Equation (2.48). (b) A typical beam search scheme: With each step, only promising hypotheses are developed, indicated by the forward arrows. Here, the search is performed frame-synchronous, i.e., concurrent hypotheses are pruned along a time frame (the dashed vertical lines).

$$\begin{aligned} D_{0,j} &= D_{i,0} = \infty,\ i = 1, \ldots, N_t,\ j = 1, \ldots, N_r \\ D_{0,0} &= 0. \end{aligned} \tag{2.49}$$

The recursion computation can practically be implemented in correspondence with various geometrical schemes, e.g., by a row-, column- or diagonal-wise expansion of $\mathbf{D}^*$. Algorithm 1 summarizes the column-wise dynamic programming algorithm.

Algorithm 1 Dynamic programming

Variables:

D^*_{ij}: *element* (i, j) *in Viterbi matrix*

1: Initialize D^*_{ij} according to Equation (2.49)
2: **for** $i = 1, \ldots, N_t$
3: **for** $j = 1, \ldots, N_r$
4: compute D^*_{ij} according to Equation (2.48)
5:
6: $D^*[d]\,(\boldsymbol{t}, \boldsymbol{r}) := D^*_{N_t N_r}$

As the resulting computational cost is proportional to the Viterbi matrix size, DP reduces the complexity of Equation (2.32) from $\mathcal{O}(|\mathbb{P}|^{\tilde{N}})$ of a naive algorithm to $\mathcal{O}(|\mathbb{P}|\,\tilde{N}^2)$, with $\tilde{N}$ the average sequence length.

2.3.3.2 Beam search

Beam search [Schukat-Talamazzini, 1995, Jelinek, 1998] denotes the strategy of eliminating path hypotheses, which are unlikely to turn out as the final, optimal solution at a specific "level" k in the DP search. A criterion for the assessment of a path hypothesis is a comparison of its accumulated distance to the best so-far developed concurrent hypothesis. In this respect, a hypothesis is eliminated if its distance

$$D > D_k + s_{\text{beam}}, \tag{2.50}$$

where D_k is the distance of the best concurrent hypothesis at level k and s_{beam} a threshold variable. The hypotheses remaining from the elimination are called *active hypotheses*. For an effective beam search algorithm, one has to define a reasonable strategy, at what stage concurrent hypotheses should be compared. Most beam search algorithms handle this issue *frame-synchronous*, i.e., they compare and eliminate hypotheses along a line $\phi_t(n) = k$ (each dashed vertical line in Figure 2.8 (b)). The most promising hypotheses are managed in terms of an entry in a list $\mathcal{A}_i$ of active states.

Algorithm 2 summarizes a typical beam search algorithm. Regard that here the list of active states is organized such that the most promising hypothesis is evaluated first, in order to get early a good estimate of D_k.

Beam search can further reduce the computational complexity of Equation (2.32) to $\mathcal{O}(|\mathbb{P}|\,\tilde{N})$, however may lead to a sub-optimal solution, if the optimal one has been eliminated due to high local distances in early regions of the sequences. Nevertheless, in practice, beam search has been shown to be a successful heuristic.

2.4 Clustering

The classification techniques described in Sections 2.2 and 2.3 can be summarized with the term *supervised learning*, because they rely on labeled data. More specifically, they assume that a classifier can be learned from data that occur together with information on the respective class membership. If these memberships are not known other techniques have to be employed. These techniques are generally summarized under the terms *unsupervised learning*. *Clustering* is one instance of unsupervised learning.

According to Theodoridis and Koutroumbas [1999, Chapters 11–16], the term clustering describes the task of " 'revealing' the organization of patterns into 'sensible' clusters (or groups)". A specification of the term "sensible" is highly dependent on factors like the underlying *pattern dissimilarity measure*, the *clustering criterion* and the *clustering algorithm*, among others. Since these factors are application dependent, it remains to the clustering designer to define them properly with respect to prior knowledge about the problem.

Mathematically speaking, a *clustering* $\mathfrak{C}$ is a partitioning of a set $\mathbb{X} = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}\right\}$ into a set $\left\{\mathbb{C}^1, \ldots, \mathbb{C}^K\right\}$ of nonempty, pairwise disjoint sets, such that $\bigcup_{k=1}^{K} \mathbb{C}^k = \mathbb{X}$. In this notation, $\mathbb{C}^k$ is the k-th cluster of $\mathfrak{C}$.

Algorithm 2 Frame-synchronous beam search

Variables:

$\mathcal{A}_k$: *list of active states in column* k
D: *new accumulated distance*
D_{ij}: *smallest distance for matrix element* (i, j)
D_k: *smallest distance for column* k
s_{beam}: *pruning threshold*

```
1:  A_1 := [1];
2:  D_ij := ∞, ∀i = 1,...,N_t, j = 1...,N_r;
3:  D_(1)(1) := D_1 := d(t_1, r_1)
4:  D_k := ∞, ∀k = 2,...,N_t;
5:  for i = 1,...,N_t      % columns
6:    for all j ∈ A_i      % active states
7:      for all Δφ ∈ ℙ      % transitions
8:        (k,l) := (i,j) + Δφ;    % goal matrix element of Δφ
9:        D := D_ij + d(t_k, r_l);    % new accumulated distance
10:       if D > D_k + s_beam      % distance exceeds beam criterion
11:         continue;     % with next Δφ
12:       if (l ∈ A_k) ∧ (D > D_k)      % another path hypothesis has smaller distance
13:         continue;     % with next Δφ
14:       D_kl := D;     % D is new smallest distance for matrix element (k,l)
15:       if D < D_k      % D is new smallest distance for column k
16:         D_k := D;
17:       if (l ∉ A_k)
18:         if (D < D_k)
19:           A_k := [l] ++ A_k;     % prepend l to list of active states of column k
20:         else
21:           A_k := A_k ++ [l];     % append l to list of active states of column k
22:
23: D*[d](t, r) := D_{N_t};
```

A number of different clustering algorithms have been developed in pattern recognition. Theodoridis and Koutroumbas [1999, Section 12.2] primarily discriminate between the following types:

1. *Sequential algorithms*: Observations are presented to the algorithm in a sequence and the clustering result depends an the order of the sequence. These types produce a single clustering.

2. *Hierarchical algorithms*: Here, a sequence of clusterings is produced. The granularity of the clustering, i.e., the number of clusters, can be flexibly specified by the selection

of one particular clustering among the sequence. An appealing property of some of the hierarchical algorithms is that they only depend on distance relationships of the patterns and not on the patterns itself.

3. *Algorithms based on cost function optimization*: In these schemes, a cost function is to be optimized. One popular representative hereof is the k-means algorithm. The number of clusters is here mostly kept fixed.

One idea in this thesis (Chapter 4) will benefit from the utilization of the agglomerative hierarchical clustering. It will be described in the following.

2.4.1 The agglomerative hierarchical clustering algorithm

A *hierarchical clustering* algorithm produces a hierarchy of *nested* clusterings $[\mathfrak{C}_1, \ldots, \mathfrak{C}_M]$. The term "nested" defines the property that each cluster in $\mathfrak{C}_{m-1}$ is a sub-set of a cluster in $\mathfrak{C}_m$ and at least one of these sub-sets is a proper one. The hierarchy is generated iteratively in M steps, a clustering at step m is obtained from the clustering produced at the previous step $m-1$.

An *agglomerative hierarchical clustering* algorithm starts the iteration with a fine granularity of M clusters and ends with one cluster, i.e., $\mathfrak{C}_1 = \left\{\left\{\mathbf{x}^{(1)}\right\}, \ldots, \left\{\mathbf{x}^{(M)}\right\}\right\}$ divides $\mathbb{X}$ into the trivial set of M clusters, $\mathfrak{C}_2$ into $M-1$ clusters, etc., until $\mathfrak{C}_M = \left\{\left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}\right\}\right\}$. Given a dissimilarity function $D\left(\mathbb{C}^k, \mathbb{C}^{k'}\right)$ of two clusters, it uses the general approach, summarized in Algorithm 3.

Algorithm 3 Hierarchical agglomerative clustering [Theodoridis and Koutroumbas, 1999].

Variables:

$\mathfrak{C}_m$: *clustering at iteration* m
$\mathbb{C}^k$: *cluster* k
$D\left(\mathbb{C}^k, \mathbb{C}^{k'}\right)$: *dissimilarity between cluster* k *and* k'

1: Initialize the clustering $\mathfrak{C}_1 = \left\{\left\{\mathbf{x}^{(1)}\right\}, \ldots, \left\{\mathbf{x}^{(M)}\right\}\right\}$
2: **for** $m = 2, \ldots, M$
3: $\quad (k_{\min}, k'_{\min}) := \operatorname{argmin}_{(k,k'),k\neq k'} D\left(\mathbb{C}^k, \mathbb{C}^{k'}\right)$
4: $\quad$ obtain the new clustering $\mathfrak{C}_m$ by merging clusters $\mathbb{C}^{k_{\min}}$ and $\mathbb{C}^{k'_{\min}}$ of $\mathfrak{C}_{m-1}$

2.4.2 Cluster and point dissimilarities

The hierarchical clustering algorithm relies on *cluster distances (or dissimilarities)* $D\left(\mathbb{C}^k, \mathbb{C}^{k'}\right)$. One possibility to define these is to use the dissimilarity of its elements $D\left(\mathbf{x}^{(k)}, \mathbf{x}^{(k')}\right)$ $\mathbf{x}^{(k)} \in \mathbb{C}^k$, $\mathbf{x}^{(k')} \in \mathbb{C}^{k'}$. More specifically, Theodoridis and Koutroumbas

[1999, Chapter 11] suggest several particular combination methods like *min*, *max* or *average dissimilarity*. The *average dissimilarity*

$$D\left(\mathbb{C}^k, \mathbb{C}^{k'}\right) = \frac{1}{O^k \cdot O^{k'}} \sum_{\mathbf{x}^k \in \mathbb{C}^k} \sum_{\mathbf{x}^{k'} \in \mathbb{C}^{k'}} D\left(\mathbf{x}^{(k)}, \mathbf{x}^{(k')}\right), \tag{2.51}$$

gives a favor to compact clusters. Here, O^k denotes the cardinality of $\mathbb{C}^k$.

If data are embedded in a vector space, one popular choice for *point dissimilarity* $D\left(\mathbf{x}^{(k)}, \mathbf{x}^{(k')}\right)$ is the Euclidean distance or its square

$$D\left(\mathbf{x}^{(k)}, \mathbf{x}^{(k')}\right) = \left\|\mathbf{x}^{(k)} - \mathbf{x}^{(k')}\right\|^2. \tag{2.52}$$

2.4.3 Cluster representatives

In many cases, it is useful to define a cluster representative. In general, a cluster representative can be a point in the cluster space, a hyperplane or a hyperspherical representative. A suitable choice for compact clusters are *point representatives* $\tilde{\mathbf{x}}^{(k)}$. Theodoridis and Koutroumbas [1999] describe different criteria for selecting point representatives, however favor the *median center* for the case that the dissimilarity of two points is not a metric. The median center $\tilde{\mathbf{x}}^{(k)} \in \mathbb{C}^k$ is defined by the property that it is the element with the smallest median distance with respect to the remaining cluster elements, i.e.,

$$\underset{\mathbf{x} \in \mathbb{C}^k, \mathbf{x} \neq \tilde{\mathbf{x}}^{(k)}}{\operatorname{med}} (D(\tilde{\mathbf{x}}^{(k)}, \mathbf{x})) \leq \underset{\mathbf{x} \in \mathbb{C}^k, \mathbf{x} \neq \mathbf{x}'}{\operatorname{med}} (D(\mathbf{x}', \mathbf{x})),\ \forall \mathbf{x}' \in \mathbb{C}^k. \tag{2.53}$$

2.4.4 Determining the number of clusters

The result of an agglomerative hierarchical clustering algorithm can be visualized by a binary tree, a so-called *dissimilarity dendrogram*. Figure 2.9 shows an example of a dissimilarity dendrogram for the clustering of five two dimensional vectors $\left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(5)}\right\}$ in the context of the squared Euclidean distance and the average dissimilarity.

A closer look at Figure 2.9 reveals a strategy to determine a proper number of clusters. Instead of iterating all M steps in Algorithm 3, the merging can be stopped, when the cluster dissimilarity exceeds a threshold $D_{\max}$, i.e., $D\left(\mathbb{C}^{k_{\min}}, \mathbb{C}^{k'_{\min}}\right) > D_{\max}$. Metaphorically, this approach cuts the dissimilarity dendrogram at the height $D_{\max}$ and uses the clustering assignment obtained from below that level.

2.5 Summary

This chapter has reviewed important, general techniques in pattern recognition. First, the supervised learning of both vector data and sequences has been described.

Particular emphasis was put onto the differentiation between the generative and the discriminative classification paradigms. The generative classifier was explained with respect

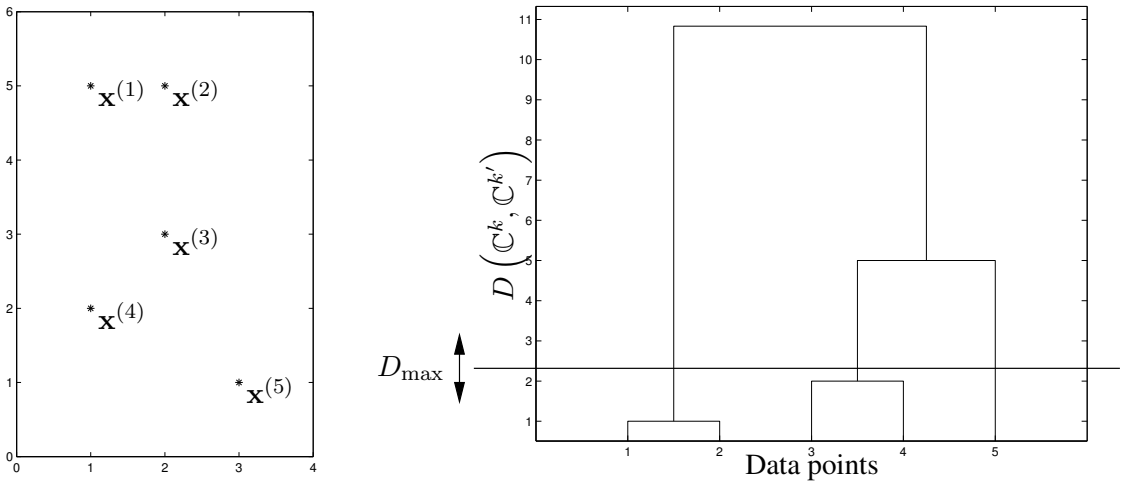


FIGURE 2.9: An example for a hierarchical clustering. (a): Five data points $\left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(5)}\right\}$ in $\mathbb{R}^2$ are illustrated. (b) The clustering is represented with help of the dissimilarity dendrogram with respect to $\left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(5)}\right\}$, using the squared Euclidean distance (Equation (2.52)) and average dissimilarity. Each leaf in the tree represents the lower subscripted vector; a node denotes a merge step of two particular clusters in algorithm 3. The level of the node on the ordinate corresponds to the cluster dissimilarity $D\left(\mathbb{C}^k, \mathbb{C}^{k'}\right)$ during merging.

to a Gaussian distribution classifier, the discriminative one with respect to a support vector machine (SVM) classifier. Benefits of both fundamental classification philosophies were opposed. Summarized, benefits of the generative approach are:

Modularity of the training: Each class is trained individually, thus the training complexity is usually smaller.

Flexibility: As each class is trained individually, new classes can be added spontaneously without re-training the whole classifier.

Probabilistic framework: The discriminants always have the semantic of a PDF or an a-posteriori probability. Hence, one can benefit from a probabilistic post-processing. In this respect, many speech and handwriting recognition systems integrate a generative classifier into probabilistic language modeling.

Advantages of the discriminative approach can be seen as the following:

Smaller complexity of the addressed function: The class boundaries are usually less complex than class conditional PDFs. In this sense, a discriminative classifier can be less wasteful with parameters.

Direct modeling of discriminants: The discriminative approach *directly* models the quantity of interest, i.e., the discriminants. Its focus concerns the modeling of the decision boundaries, not of the class regions.

Further, supervised (generative) classification methods were transferred to situations where data can not be embedded in a feature space of fixed dimension, but are sequences of vectors. In particular, the concepts of DTW and HMM have been introduced.

Finally, we have reviewed unsupervised learning. One instance of a clustering method, the agglomerative hierarchical clustering was introduced, preparatory for later chapters.

All of the methods will be useful in the remainder of the thesis. The generative classification paradigm will be picked up again and combined with an adapted sequence clustering in Chapters 4 and 6. The discriminative classification paradigm will be pursued by establishing a sequence compatible SVM kernel in Chapter 7.

CHAPTER 3

Data, pre-processing, and feature extraction in online HWR

This chapter describes the stages in the recognition process that are specific to the underlying application of online HWR. In particular, it addresses the online handwriting data format, data acquisition and data collections, as well as pre-processing, and feature extraction.

3.1 Introduction

As noted above, online handwriting data are typically sequences of coordinates, digitized by means of an electronic tablet and a pen. As with many pattern recognition problems, the raw data is not directly used as input for the classifier. One potential problem are noise artifacts in the raw data. Then, the raw data are not invariant with respect to simple transformations such as change of position or scale. In order to cope with these issues, a pre-processing and feature extraction are employed prior to classification.

In general, pre-processing addresses the problems of segmentation, noise reduction and normalization. Feature extraction generates a set of characteristics that shall allow for robust discrimination of patterns of one category against those of others.

In the remainder of this chapter, we start with a brief review of commonly employed pre-processing and feature extraction methods in online HWR. Afterwards, we discuss the particular concepts and techniques that are pursued in *frog on hand*: the underlying *UNIPEN* online data representation and collection, the chosen pre-processing, and feature extraction.

3.2 Literature review

Pre-processing and feature extraction for online HWR have been widely studied. Comprehensive overviews exist, for example, in a number of review articles [Plamondon et al., 1999, Guerfali and Plamondon, 1993, Tappert et al., 1990]. Further ideas can be found in the above mentioned HWR system descriptions [Hu et al., 2000, Seni et al., 1996, Kosmala, 2000, Manke et al., 1995, Schenkel et al., 1995, Guyon et al., 1991].

3.2.1 Pre-processing

Pre-processing addresses the following problems:

(Pre-) Segmentation: Segmentation is the process of dividing an entire writing into smaller pieces (i.e., segments). Segments can be, e.g., text lines, words, characters, strokes (i.e., connected lines), or atomic sub-strokes (i.e., parts of a stroke, that occur in different characters, but have a similar shape). In case the segmentation is completed entirely during the pre-processing, one speaks of "external segmentation". More often, the problem is too challenging for an external segmentation. In those situations, a number of segmentation hypotheses are generated during the pre-processing. The hypotheses are later refined in combination with the classification. However, it is also common to abstain from a pre-segmentation and solve this task implicitly during classification. In this case, one speaks of "internal segmentation".

Both, temporal as well as spatial information is used in (pre-) segmentation [Plamondon et al., 1999].

Noise reduction: In handwriting, noise is commonly present due to hardware limitations or difficult writing conditions. Frequently observed imperfections are a shaky writing, hooks at the beginning or end of characters and sporadic wild points (i.e., outliers far away from the pen trace). Further, a single stroke is sometimes incorrectly broken into two pieces during the data acquisition [Plamondon et al., 1999].

Many approaches [Hu et al., 2000, Kosmala, 2000] deal with the problem of high-frequency noise in the writing curve by employing a spline approximation. Next, the resulting functional spline curve is often re-sampled in order to obtain a smoother coordinate sequence. The problems of de-hooking, wild point correction and break correction are mostly addressed by filtering and by heuristic methods [Guerfali and Plamondon, 1993].

Normalization: Normalization is applied in order to reduce the amount of variability in the patterns prior to classification and thus to simplify the further recognition process. Typical sources for variability are the following: One can often observe a vertical drift of the imaginary baseline over time (also called *writing skew*). A normalization step re-aligns the writing with respect to an estimate of the baseline. Further, different users write with different slant and scale.

Normalization algorithms for a skew correction are based, for instance, on regression [Seni et al., 1996], the Hough transform [Hu et al., 2000] or projection histograms [Kavallieratou et al., 1999, Kosmala, 2000]. Slant correction algorithms employ, for example, an extreme point analysis [Brocklehurst and Kenward, 1988, Hu et al., 2000] or, also, projection histograms [Kavallieratou et al., 1999, Kosmala, 2000]. The writing scale and position is commonly normalized with respect to the writing's bounding box or estimated reference lines.

After all, a large amount of parameter fine-tuning is required for most parts of the preprocessing, which often follows heuristic approaches. One resulting problem is that preprocessing can irrevocably cross out valuable discriminative information. This issue is even more important when data of different quality is present. For example, an exaggerated smoothing can eliminate cusps, which often are the only distinctive marks between characters (as in the case of "u" and "v").

3.2.2 Feature extraction

Following a definition of Devijver and Kittler [1982], the term *feature extraction* in a pattern recognition problem denotes the process of "extracting from the raw data the information which is most relevant for classification purposes, in the sense of minimizing within-class pattern variability while enhancing the between-class pattern variability."

Although some general methods exist (principal component analysis, linear discriminant analysis, etc.; cf. Jain et al. [2000]), feature extraction is still a challenging problem in most pattern recognition contexts. In many cases, the system designer can improve the recognition performance by developing individual features, that include prior knowledge about the underlying application and typical data variability.

Feature extraction methods that have been specifically studied in the online HWR context can roughly be assigned to one of the following categories:

Temporally local features: *Temporally local features* are most often part of a dynamic feature representation, that is, a sequence $t = [\mathbf{t}_1, \ldots, \mathbf{t}_N]$ of feature vectors $\mathbf{t}_i$, each of it computed from the sample point i or its direct *temporal* vicinity. Temporally local features are often based on differential geometry characteristics of planar curves [Carmo, 1976]. Common choices are normalized representations of the coordinates and their first and second order derivatives [Hu et al., 2000, Manke et al., 1995].

Spatially local features: For online handwriting data, the spatial vicinity does not always coincide with the temporal vicinity. For example, at the crossings in the "t", "x" or "f" two samples can indeed cover each other spatially but come from different parts in the writing trace. Similar deliberations are true for contact points in many characters like "a", "b", "d", etc. In order to increase the influence of spatial relations, researchers [Manke et al., 1995, Kosmala, 2000] have introduced so-called context bitmaps. These

are low-resolution images of the spatial vicinity of each sample point i, and are included into the feature vector $\mathbf{t}_i$. Further, binary features that indicate the presence of a cusp, loop or crossing in the vicinity of sample point i, have been included into $\mathbf{t}_i$ [Schenkel et al., 1995, Hu et al., 2000].

Global features: A *global feature* combines long-range information about the pattern into one feature value. In online HWR, global features have been studied in the context of Fourier coefficients or the presence or numbers of ascenders, descenders or diacritical marks in the writing [Tappert et al., 1990]. Plamondon et al. [1999] use the vector between starting point and the last vertical maximum in the trajectory as a global feature. In general, it can be stated that global features are more informative than local features, but less robust.

3.3 Online handwriting data

Online handwriting data is typically a dynamic, digitized representation of the pen trace, containing sequential information about position, velocity, acceleration or pen angles as a function of time.

3.3.1 Data acquisition

Online handwriting is usually acquired with a writing pad and a pen. Two different digitizing technologies are dominant nowadays:

1. *Touch sensitive sensors* are installed beneath the surface of PDAs and smart phones. They are able to sense and digitize the contact of any pen alike object, thus they do not require a special pen.

2. *Electromagnetic sensors* are mostly employed in writing tablets and tablet PCs. They work only in combination with a special pen, and also catch the pen signal when it hovers in a range of up to approximately 1cm above the surface. In contrast to touch sensitive sensors, the user can lay his hand onto the surface without disturbing the acquisition.

3.3.2 Data format

The most prominent online handwriting data format in research is the *UNIPEN* format [Guyon et al., 1994].[1] Simply speaking, *UNIPEN* treats handwriting as a sequence $\mathcal{p} = [\mathbf{p}_1, \ldots, \mathbf{p}_N]$ of sample points $\mathbf{p}_i$ of pen tip information. The sampling may or may not be regular (in time).

[1] Currently, a new standard *InkML* (`http://www.w3.org/TR/InkML/`) is being developed and expected to replace *UNIPEN*.

Category key / Category	# data samples
1a / isolated digits	15953
1b / isolated upper case	28069
1c / isolated lower case	61351
1d / symbols (punctuation, etc.)	17286
2 / isolated characters, mixed case	122628
3 / isolated characters in context of words or text	67352
4 / isolated printed words, not mixed with digits or symbols	0
5 / isolated printed words, full character set	0
6 / isolated cursive or mixed-style words (w/o digits or symbols)	75529
7 / isolated words, any style, full character set	85213
8 / free text of minimum two words	14544

TABLE 3.1: *UNIPEN* categories and their sizes in the "Train-R01/V07" data collection.

In this thesis, $\mathbf{p}_i = (x_i, y_i, s_i)^T$ comprises the plane coordinates (x_i, y_i) and the pen status $s_i \in \{0, 1\}$. The latter is a boolean value indicating if the pen touches the pad ($s_i = 1$) or is lifted ($s_i = 0$) (see Figure 3.2(a)). The term *component* is used to denote a connected sample point sequence with equal pen status.

Additional information like velocity, acceleration and pen angles — although included in the *UNIPEN* data format — is generally present only in a few samples of common databases. As it is not clear how much it contributes to a successful recognition, and in order to have access to the largest amount of data, it is not taken into account in this work.

3.3.3 Data collections

The name *UNIPEN* is also associated with an online handwriting data collection. Parts of it have become the most popular non-proprietary database in online handwriting research. It is publicly available within the "Train-R01/V07" database, distributed by the INTERNATIONAL UNIPEN FOUNDATION.[2] Additional *UNIPEN* databases exist, but are not publicly available. Table 3.1 summarizes different handwriting categories and their sizes in "Train-R01/V07".

3.4 Data pre-processing

It has been argued in Section 3.2.1 that pre-processing is very sensitive to the underlying data, among other things its quality. It is further true that the viability of individual pre-processing steps strongly depends on the handwriting type. For example, skew slant and reference line estimation are very unreliable for isolated characters, as they would be based on only a few

[2] `http://unipen.nici.kun.nl/cdroms/`

data points. In fact, experiments have shown [Simon, 2002] that an extensive pre-processing leads to a degraded recognition accuracy in the character recognition context.

In this respect, the pre-processing in *frog on hand* is treated differently for the various handwriting types. In the present work, HWR is being applied to isolated characters and words. A detailed description of the character pre-processing will follow in the next section. As the word classification will be treated rather marginally, the description of word pre-processing methods will be more concise. A more detailed description of word preprocessing is given by Simon [2003].

3.4.1 Data pre-processing for isolated characters

Pre-processing for isolated characters consists of two steps: elimination of spurious sample points and normalization.

Removal of pen-up components: The writing's pen-up samples $\mathbf{p}_i$ (samples with $s_i = 0$) are eliminated. The remaining pen-down samples (i.e., samples with $s_i = 1$) are concatenated to a single coordinate sequence $\mathcal{p}' = [\mathbf{p}'_1, \ldots, \mathbf{p}'_{N'}]$.

Duplicate removal: The *UNIPEN* database includes a number of writings that contain repetitions of coordinates, that is, $\mathbf{p}'_i = \mathbf{p}'_{i+1} = \ldots = \mathbf{p}'_{i+d}$ for some i and d. When computing differential features, these co-occurrences can cause singularities. Thus, the extra occurrences $\mathbf{p}'_{i+1}, \ldots, \mathbf{p}'_{i+d}$ are removed. The new representation shall be named $\mathcal{p}'' = [\mathbf{p}''_1, \ldots, \mathbf{p}''_{N''}]$

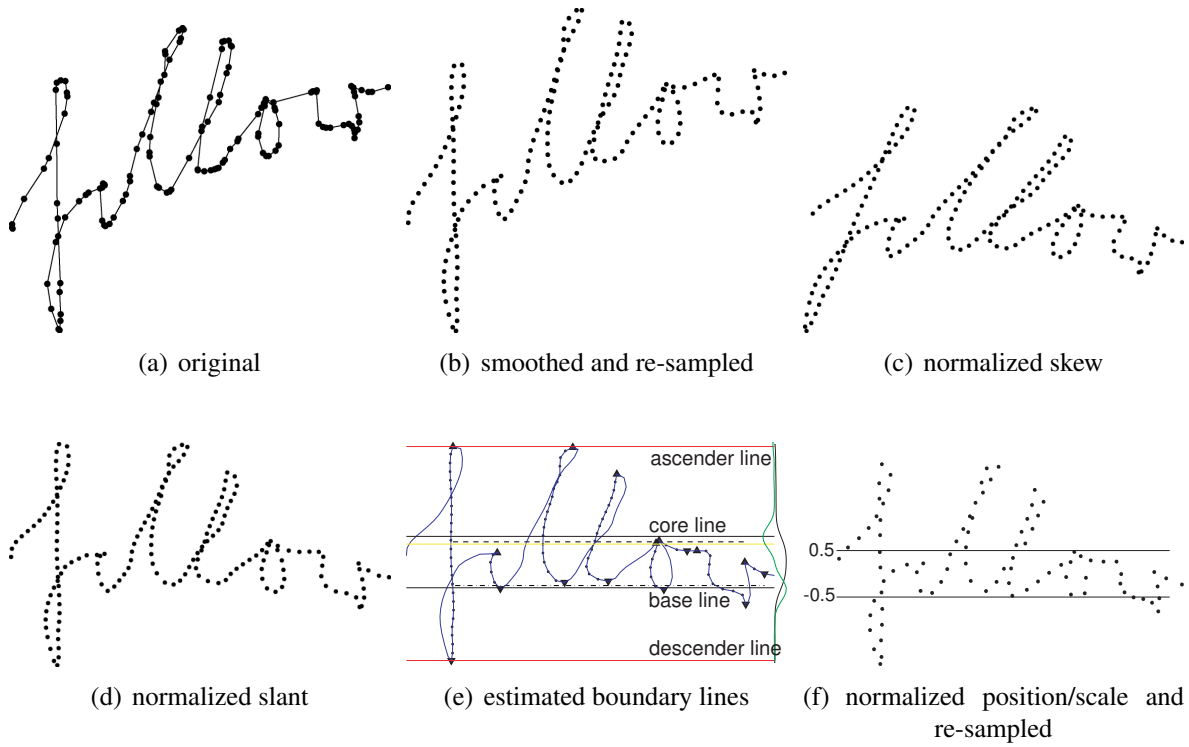
Position and scale normalization: The writing's position and scale are normalized by

$$\tilde{x}_i = \frac{x_i - \mu_x}{r} \tag{3.1}$$

$$\tilde{y}_i = \frac{y_i - \mu_y}{r} \tag{3.2}$$

Here, $\boldsymbol{\mu} = (\mu_x, \mu_y)^T = \frac{1}{N''} \sum_{i=1}^{N''} \mathbf{p}''_i$ denotes the sample mean and $r = \sigma_y = \sqrt{\frac{1}{N''-1} \sum_{i=1}^{N''} (\mu_y - y_i)^2}$ the (vertical) y standard deviation of the character's sample points.

Further character pre-processing has also been studied [Simon, 2002]. In these studies the writing was approximated by a cubic spline, re-parameterized by the arc length and equidistantly re-sampled. Simon [2002] found that the recognition accuracy decreases with this pre-processing. Hence, this further, expensive pre-processing step is skipped in the isolated character context. Instead, the classifier copes with the associated variations.



(a) original (b) smoothed and re-sampled (c) normalized skew

(d) normalized slant (e) estimated boundary lines (f) normalized position/scale and re-sampled

FIGURE 3.1: Pre-processing steps for handwritten words. (Figures taken from [Simon, 2003].)

3.4.2 Data pre-processing for words

If handwriting data correspond to entire words, a more complex preprocessing is worthwhile, because:

1. Compared to isolated characters, the skew, slant and the reference lines of a handwritten word can be estimated more accurately. This suggests to make use of those parameters during normalization.

2. Empirical studies indicate that reference lines can be estimated even more accurately, if the writing is available at a higher sampling rate and with equidistant arc length. Thus, the writing should be re-sampled prior to the reference line estimation.

Word pre-processing for *frog on hand* has been developed by Simon [2003]. A concise review of all steps is given in the following:

Removal of pen-up components and duplicates: These procedures are the same as for isolated characters (cf. previous section).

Smoothing and re-sampling: As indicated above, smoothing and re-sampling are employed mainly with the aim of a more accurate reference line estimation (cf. Figure 3.1 (c)). The procedure comprises the following steps:

1. Cusps are detected by thresholding the difference of two consecutive angles.
2. Each stroke, that is, a line segment between two cusps (or the starting/end point of a component), is approximated by a cubic B-spline [deBoor, 1978]. This approximation results in a moderate smoothing.
3. The writing is re-parametrized by the arc length and equidistantly re-sampled.

A smoothed and re-sampled writing, named $\boldsymbol{p}'''$, is shown in Figure 3.1 (b).

Skew correction: The aim of the skew correction is to estimate the angle $\hat{\alpha}^{\text{skew}}$ of the main writing direction and to rotate it according to $\hat{\alpha}^{\text{skew}}$. The employed solution uses projection histograms and its rough idea is as follows:

1. Rotate the re-sampled writing $\boldsymbol{p}'''$ by $\alpha \in \{-45°, -44°, \ldots, 44°, 45°\}$. Denote an α-rotated writing $\boldsymbol{p}^{(\alpha)}$.
2. Compute the y-histogram of all $\boldsymbol{p}^{(\alpha)}$ and low-pass filter it. A filtered y-histogram is named $P_y^{(\alpha)}$.
3. It can be argued that the y-histogram of a horizontally oriented writing has minimum entropy. Thus, compute the entropy $H^{(\alpha)}$ for each $P_y^{(\alpha)}$ and estimate $\hat{\alpha}^{\text{skew}} = \text{argmin}_\alpha \left\{H^{(\alpha)}\right\}$.
4. Correct the writing skew by rotating $\boldsymbol{p}'''$ about $-\hat{\alpha}^{\text{skew}}$.

Slant correction: The aim of the slant correction is to remove the main slant of the writing. A similar algorithm as the one for the skew correction can be used. Two main differences address: (1) an x-histogram is used instead of the y-histogram and (2) the writing is *sheared* by the angles $\alpha \in \{-45°, -44°, \ldots, 44°, 45°\}$ instead of *rotated*. A slant-corrected writing is shown in Figure 3.1 (d).

Reference line estimation: Latin handwriting extends across three horizontal zones. Ideally, those zones are limited by four parallel reference lines: the *descender line*, *base line*, *core line* and *ascender line* (cf. Figure 3.1 (e)). These lines, especially base and core line, contain valuable information for position and scale normalization. A simplified variant of the algorithm proposed by [Simon, 2003] works as follows:

1. Compute the writing's y-histogram and low-pass filter it. Denote the filtered y-histogram P_y.
2. Compute the first derivative $\frac{\partial}{\partial y} P_y$.

3. Compute the height of the base line, $\hat{y}_{\text{base}}$, and of the core line, $\hat{y}_{\text{core}}$, from the extreme points of $\frac{\partial}{\partial y}P_y$, respectively:

$$\hat{y}_{\text{core}} = \operatorname*{argmin}_y \left\{ \frac{\partial}{\partial y} P_y \right\} \tag{3.3}$$

$$\hat{y}_{\text{base}} = \operatorname*{argmax}_y \left\{ \frac{\partial}{\partial y} P_y \right\} \tag{3.4}$$

Position and scale normalization: The writing's position and scale are normalized by the coordinate transformations

$$\tilde{x}'_i = \frac{x_i}{r} \tag{3.5}$$

$$\tilde{y}'_i = \frac{y_i - y_{\text{base}}}{r} - \frac{1}{2}. \tag{3.6}$$

Here, x_i and y_i denote the coordinates of the so far pre-processed writing. The scaling factor $r = y_{\text{core}} - y_{\text{base}}$ is the height of the so called *core zone*. The normalization is illustrated in Figure 3.1 (f).

Re-sampling: The writing is re-sampled a second time to facilitate classification: A lower sampling rate reduced the number of sample points. A scale-normalized and re-sampled writing is shown in Figure 3.1 (f).

3.5 Feature extraction

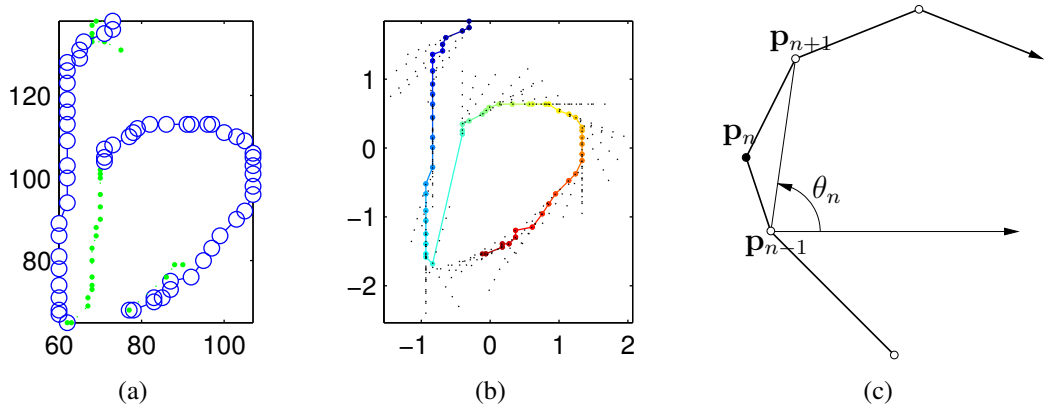
The feature extraction used in this work is based on temporally local features and transforms $\boldsymbol{p}$ into a dynamic representation, that is, a sequence $\boldsymbol{t} = [\mathbf{t}_1, \ldots, \mathbf{t}_N]$ of feature vectors $\mathbf{t}_i = (t_{i1}, \ldots, t_{iF})^T \in \mathbb{R}^F$.

Like the pre-processing, the feature extraction is different for isolated characters and for words.

In what follows $\boldsymbol{p} = [\mathbf{p}_1, \ldots, \mathbf{p}_N]$ denotes the pre-processed writing.

3.5.1 Feature extraction for isolated characters

Several features have been studied with *frog on hand* [Simon, 2002], inspired by recent publications [Guyon et al., 1991, Schenkel et al., 1995, Seni et al., 1996, Hu et al., 2000, Jäger et al., 2000]. Those studies include a normalized representation of the coordinates, a representation of the tangent slope angle, a normalized curvature, the ratio of tangents, etc. The best character recognition rates — in combination with the classification described in Chapter 4 — have been observed with the following simple features:



(a) (b) (c)

FIGURE 3.2: (a) A sample of a *UNIPEN* character p of class "b". Coordinates are illustrated by a circle for pen-down and by a point for pen-up components. (b) Illustration of a feature sequence t and the features $\tilde{x}_i$, $\tilde{y}_i$ and θ_i: $\tilde{x}_i$ and $\tilde{y}_i$ are plotted according to their value in the $\tilde{x}$-$\tilde{y}$-plane. The dashed lines illustrate θ_i by the direction of the tangent. (c) The tangent slope angle is approximated by the secant, approximating the neighboring sample points.

t_{i1}, t_{i2} **— normalized horizontal and vertical coordinates.** With

$$t_{i1} = \tilde{x}_i \tag{3.7}$$
$$t_{i2} = \tilde{y}_i, \tag{3.8}$$

as defined in Section 3.4.1, a normalized representation of the plane coordinates is incorporated into the feature vector. Both features are invariant with respect to position and scale.

t_{i3} **— tangent slope angle.** The feature

$$t_{i3} = \theta_i = \arg\left((x_{i+1} - x_{i-1}) + J \cdot (y_{i+1} - y_{i-1})\right), \tag{3.9}$$

with $J^2 = -1$ the imaginary unit and "arg" the complex phase, is an approximation of the tangent slope angle at point i (cf. Figure 3.2 (c)). θ_i is also invariant with respect to position and scale. Results about the relevance of θ_i in the context of $\tilde{x}_i$ and $\tilde{y}_i$ will be given in Section 5.6.

Since θ_i is a directional quantity, a special treatment for the computation of probabilities, statistics and distances is necessary. A detailed discussion of this issue is provided in Chapter 5.

To summarize, a writing's feature representation is defined as $t = [\mathbf{t}_1, \ldots, \mathbf{t}_N]$, where $\mathbf{t}_i = (\tilde{x}_i, \tilde{y}_i, \theta_i)^T$ for $i = 1, \ldots, N$. Figure 3.2 (b) illustrates the feature representation for isolated characters.

3.5.2 Feature extraction for words

In the context of an unsegmented writing, no information about the horizontal character boundaries is available. In this respect, no straightforward (position and scale) normalization of the x-coordinate exists.

The situation is different for the y-coordinate. With accurate estimates of the reference lines, there are parameters for a normalization of the vertical coordinate available. The variable $\tilde{y}'_i$, as defined in Equation (3.6), is invariant with respect to position and scale. Further, the tangent slope angle θ_i is invariant with respect to position and scale.

In summary, each feature vector $\mathbf{t}_i$ of the vector sequence $t = [\mathbf{t}_1, \ldots, \mathbf{t}_N]$ comprises two elements in the context of words:

t_{i1} **— normalized vertical coordinate.** $t_{i1} = \tilde{y}'_i$, as defined in Section 3.4.2

t_{i2} **— tangent slope angle.** $t_{i2} = \theta_i$, as defined in Section 3.5.1.

3.6 Summary

Data acquisition, pre-processing and feature extraction are the first steps in nearly every pattern recognition system. This chapter has addressed these issues for online HWR. First, typical online HWR pre-processing and feature extraction methods have been reviewed.

Following, the data, pre-processing and feature extraction, as employed in the context of *frog on hand*, have been described. It has been argued that it is worthwhile to optimize both pre-processing, and feature extraction for the respective handwriting type, either isolated characters or words.

Nevertheless, both procedures have in common that the resulting feature representation is given as a sequences of vectors, $t = [\mathbf{t}_1, \ldots, \mathbf{t}_N]$. This enables the use of uniform classification approaches for characters and words.

CHAPTER 4

CSDTW — A generative sequence classification framework

This chapter describes the generative classification framework cluster generative statistical dynamic time warping (CSDTW). *CSDTW is a general, scalable approach for variable-sized, sequential data that holistically combines cluster analysis and statistical sequence modeling. It can handle general classification problems that rely on this sequential type of data. Contrary to previous attempts to a common clustering and statistical modeling combination these two issues are embedded in a single feature space and use a closely related distance measure. Character recognition experiments with CSDTW indicate that the recognition accuracy is significantly higher than reported results of other handwriting recognition systems.*

4.1 Introduction

The difficulty of designing a writer independent HWR system is commonly explained as follows. First, typical target devices like PDAs and smart phones have limitations in computational power and memory size, which is cumbersome in the system design for these devices. Second, for a writer independent solution the system has to discriminate between a large variety of different writing styles which are present in the target group of users. Even more difficult for online recognition, a writing which looks similar in a graphical (i.e., offline) representation, can have a different sequential (i.e., online) representation.

Thus there is demand for a HWR system which is efficient, scalable to the device's capability, accurate and which can deal with the natural handwriting of a wide range of different writers and writing styles.

Cluster generative statistical dynamic time warping (CSDTW) [Bahlmann and Burkhardt,

2004] is a generative, holistic approach that aims at coping with these difficulties. Like all generative approaches (cf. Section 2.2.2) it models the class conditional PDFs for each character class. As the difficulties with generative methods is often the complexity of the generative functions (cf. Figure 2.1), CSDTW aims to decompose classes into smaller sub-classes. For this purpose it employs a hierarchical clustering (cf. Section 2.4). A beneficial characteristic of CSDTW's procedure over previous methods is that the combination of cluster analysis and generative statistical sequence modeling is treated *holistically*, i.e., a closely related dissimilarity measure is utilized in both stages.

Another attractive property of CSDTW is its scalability. By adjusting particular parameters the system designer can straightforwardly find a compromise between the classifier size and the recognition accuracy.

4.2 Literature review

In order to argue the particular advance by CSDTW over previous attempts to the combination of clustering and statistical modeling in the context of sequential data, the philosophy of those shall be reviewed. Previous methods follow mainly two different philosophies:

1. A powerful classifier (e.g., a hidden Markov model, HMM) uses cluster information which has been revealed in a different feature space and with a different assumption about (dis-) similarity [Connell and Jain, 1994, Oates et al., 1999, Lee et al., 2000, Smyth, 1997]. Hence, those algorithms cannot be certain that the clusters found in the cluster space correspond to well-formed clusters in the classifier space.

2. Clustering and classification are performed in the same feature space, however the classifier uses simple, limited techniques like template matching [Vuurpijl and Schomaker, 1997, Prevost and Milgram, 1999, Vuori et al., 2001].

A deviation from these two philosophies exists. Perrone and Connell [2000] embed a clustering/HMM hybrid in the single feature space of the HMM parameters. However, this approach lacks a generic quality, since their (iterative) clustering relies on a reasonable initialization, which the authors perform by a manual adjustment.

4.3 CSDTW modeling

This section explains the general concept of CSDTW. In the context of HWR, the philosophy of CSDTW is to model each different character writing style by a distinct generative statistical model. In literature the terms *allograph* [Vuurpijl and Schomaker, 1997, Prevost and Milgram, 1999, Plamondon et al., 1999, Perrone and Connell, 2000, Schomaker, 1993, Bercu and Lorette, 1993] or *lexeme* [Connell and Jain, 1994] have been evolved to describe such a distinct character style. With CSDTW, the scope of the term allograph can be

flexibly implemented. Adaptable to the classification background, CSDTW includes a mechanism to specify whether just large data variations (e.g., culturally conditioned) or additionally smaller variations (e.g., due to different writer habits or the word context in that a character is written) shall be regarded for the generation of the distinct allograph models.

In the proposed CSDTW framework, both clustering and statistical modeling are embedded in a single feature space — the space of the training samples. As will be shown, they assume a closely related distance measure, while retaining a powerful, HMM-alike classification scheme. Additionally, the CSDTW approach includes a strategy to an open question in statistical sequence modeling research — the problem of dimensioning and initializing the statistical models in the context of the commonly used iterative classifier training. Thus, no manual dimensioning or initialization has to be employed.

4.3.1 CSDTW classification

The aim of this section is to describe the classification part of CSDTW. At this point it shall be assumed that a classifier has already been trained from a set $\mathbb{X} = \left\{\mathbb{X}^{(1)}, \ldots, \mathbb{X}^{(L)}\right\}$ of labeled characters, where $\mathbb{X}^{(l)} = \left\{\boldsymbol{\mathcal{X}}^{(l,1)}, \ldots, \boldsymbol{\mathcal{X}}^{(l,M_l)}\right\}$ is the training set of class l. The DTW and HMM concepts, as introduced in Section 2.3, will play a central role in these deliberations.

4.3.1.1 DTW distance

DTW has been introduced in Section 2.3.1 with the squared Euclidean distance $d\left(\mathbf{t}_i, \mathbf{r}_j\right)$ as the local distance measure. In the context of CSDTW a different choice of local distance is more profitable. We anticipate our choice at this point, however motivate it later in Section 4.3.1.4. So, instead of the squared Euclidean distance an adaptation of it

$$\begin{aligned} \tilde{d}\left(\mathbf{t}_i, \mathbf{r}_j\right) &= \frac{1}{2}\left(\ln\left(|2\pi\boldsymbol{\Sigma}|\right) + \left(\mathbf{t}_i - \mathbf{r}_j\right)^T \boldsymbol{\Sigma}^{-1}\left(\mathbf{t}_i - \mathbf{r}_j\right)\right) \\ &\quad + \ln\left(|\mathbb{P}|\right) \end{aligned} \tag{4.1}$$

will be utilized in the CSDTW context (in particular in the clustering, Section 4.3.2.1). Here, $|\mathbb{P}|$ denotes the cardinality of the transition set $\mathbb{P}$ and $\boldsymbol{\Sigma}$ a global covariance matrix of dimension $F \times F$. $\boldsymbol{\Sigma}$ can be used to model prior knowledge, e.g., about expected variances σ_f^2 of the feature dimension f, i.e., $\boldsymbol{\Sigma} = \operatorname{diag}\left(\sigma_1^2, \ldots, \sigma_F^2\right)$. Alternatively, if no prior knowledge is given, it can be set to unity, i.e., $\boldsymbol{\Sigma} = \boldsymbol{I}$, or a scalar multiple of it.

At this point we can benefit from the parameterization of $D^*[d]$ by the local distance function d. In this respect, $D^*[\tilde{d}]$ describes the DTW distance with $\tilde{d}$ as local distance.

4.3.1.2 Statistical DTW (SDTW) distance

In the SDTW modeling, a character reference will not be represented by a sequence $\boldsymbol{\mathcal{r}} = [\mathbf{r}_1, \ldots, \mathbf{r}_{N_r}]$ of *feature vectors*, as with DTW, but by a sequence $\boldsymbol{\mathcal{R}} = [\mathbf{R}_1, \ldots, \mathbf{R}_{N_{\mathcal{R}}}]$ of a tuple $\mathbf{R}_j$ of *statistical quantities*.

This comprises, similar to HMMs:

1. discrete probabilities, say $\alpha_j : \mathbb{P} \rightarrow [0, 1]$, for the statistical modeling of the transitions $\Delta\boldsymbol{\phi} \in \mathbb{P}$ reaching the sequence's sample point j, and
2. a continuous PDF $\beta_j : \mathbb{R}^F \rightarrow \mathbb{R}$, that models the feature distribution $\mathbf{x} \in \mathbb{R}^F$ at the sequence's sample point j,

thus $\mathbf{R}_j = (\alpha_j, \beta_j)$.

The choice of the variable names α_j and β_j as the Greek variations of the HMM elements a_{ij} and b_j emphasize the similarity of the SDTW to the HMM framework (cf. Section 2.3.2). Section 4.3.1.4 will provide a unified analysis of SDTW and HMM.

In the next paragraphs, a few details about α_j and β_j will follow. The state transition probabilities α_j are defined by

$$\alpha_j\left(\Delta\boldsymbol{\phi}\right) = P\left(\Delta\boldsymbol{\phi}\left(n\right) = \boldsymbol{\phi}\left(n\right) - \boldsymbol{\phi}\left(n-1\right) | \phi_{\mathcal{R}}\left(n\right) = j\right), \quad \forall \Delta\boldsymbol{\phi} \in \mathbb{P}, \tag{4.2}$$

where the special case for $n = j = 1$ is

$$P\left(\Delta\boldsymbol{\phi}\left(1\right) = \boldsymbol{\phi}\left(1\right) - \boldsymbol{\phi}\left(0\right) | \phi_{\mathcal{R}}\left(1\right) = 1\right) = 1 \tag{4.3}$$

by definition.

In some situations, the probability $\alpha'_j\left(\Delta\boldsymbol{\phi}\right)$ that a transition $\Delta\boldsymbol{\phi}$ emerges from (instead of reaches) the sequence's sample point j, is of interest. $\alpha'_j\left(\Delta\boldsymbol{\phi}\right)$ fulfills the standard stochastic constraints

$$\sum_{\Delta\boldsymbol{\phi} \in \mathbb{P}} \alpha'_j\left(\Delta\boldsymbol{\phi}\right) = 1 \tag{4.4}$$

and is related to $\alpha_j\left(\Delta\boldsymbol{\phi}\right)$ by the equation

$$\alpha_j\left(\Delta\boldsymbol{\phi}\right) = \alpha'_{j-\Delta\phi_{\mathcal{R}}}\left(\Delta\boldsymbol{\phi}\right), \tag{4.5}$$

with $\Delta\phi_{\mathcal{R}}$ the $\mathcal{R}$-component of $\Delta\boldsymbol{\phi}$.

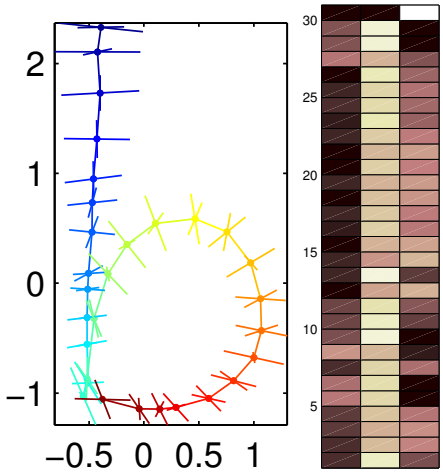
The described SDTW implementation models the PDF β_j by a unimodal, multivariate Gaussian, i.e.,

$$\beta_j\left(\mathbf{x}\right) = p\left(\mathbf{x} | \phi_{\mathcal{R}}\left(n\right) = j\right) = \mathcal{N}_{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j}\left(\mathbf{x}\right) = \left(\left|2\pi\boldsymbol{\Sigma}_j\right| \exp\left(\left(\mathbf{x} - \boldsymbol{\mu}_j\right)^T \boldsymbol{\Sigma}_j^{-1}\left(\mathbf{x} - \boldsymbol{\mu}_j\right)\right)\right)^{-1/2}. \tag{4.6}$$

In a general context the assumption of such a type of PDF would be quite restrictive. Other HWR systems utilize Gaussian mixture or discrete probability models [Hu et al., 2000], which are more flexible than unimodal Gaussians. However, the described approach assumes that large, especially multimodal variations in the data are due to a different writing *style*. In this sense, the philosophy is to model each style by a distinct, but simple character sub-class model.

A graphical illustration for an example of $\mathcal{R}$ — a reference model of a character "b" — is shown in Figure 4.1.

With the new representation of the reference models with respect to DTW — i.e., the utilization of $\mathcal{R}$ instead of r — a new local distance can be adapted for the use in the SDTW



(a) β_j (b) α'_j

FIGURE 4.1: An example of an allograph reference model $\mathcal{R}$ (of character class "b"). In part (a) the reader can see a connected sequence of dots, each one with two additional lines attached. As a reference model is represented by a sequence of Gaussian PDFs, each dot illustrates the mean of the Gaussian and the lines the covariance matrix. The direction of the two lines match the projection of the first two eigenvectors onto the $\tilde{x}$-$\tilde{y}$-plane, their length the square root of the corresponding eigenvalues. Thus, the lines indicate the orientation and width of the projected Gaussian. Note that the model is represented by means of the features $(\tilde{x}_i, \tilde{y}_i)^T \in \mathbb{R}^2$, which is a projection of the chosen feature vector $(\tilde{x}_i, \tilde{y}_i, \theta_i)^T \in \mathbb{R}^3$. Part (b) illustrates the transition probabilities $\alpha'_j(\Delta\boldsymbol{\phi})$ as a color plot. The three columns correspond to the transitions $(0,1)$, $(1,1)$ and $(1,0)$, from left to right. The rows corresponds to the reference model index j, from bottom to top. Light colors denote high values for $\alpha'_j(\Delta\boldsymbol{\phi})$, dark colors low values. Note that $\alpha'_{N_\mathcal{R}}((0,1)) = \alpha'_{N_\mathcal{R}}((1,1)) = 0$, as $N_\mathcal{R}$ is the last sample point in $\mathcal{R}$.

context. The proposed adaptation naturally emerges from a ML classification criterion and can be derived from the following proposition. Initial deliberations have been studied by Bockhorn [2000]. The proposition starts with a summary of the modeling assumptions.

Proposition 4.1. *Assume that the following prerequisites for a vector sequence observation $\mathbf{t}$, an SDTW model $\mathcal{R}$ and an alignment function $\boldsymbol{\Phi}$ are valid:*

1. *Markov property for $\boldsymbol{\Phi}$: The alignment $\boldsymbol{\Phi} = [\boldsymbol{\phi}(1), \ldots, \boldsymbol{\phi}(N)]$ can be decomposed into a sequence of transitions*

$$\Delta\boldsymbol{\phi}(n) = \boldsymbol{\phi}(n) - \boldsymbol{\phi}(n-1) \in \mathbb{P},\ n = 2, \ldots, N \tag{4.7}$$

and the transitions are sequentially independent and solely depend on the reference position $\phi_\mathcal{R}(n)$:

$$P(\Delta\boldsymbol{\phi}(n) | \boldsymbol{\phi}(n), \ldots, \boldsymbol{\phi}(1)) = P(\Delta\boldsymbol{\phi}(n) | \phi_\mathcal{R}(n)),\ n = 1, \ldots, N. \tag{4.8}$$

2. *The PDFs for an observed sequence* $\boldsymbol{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_{N_x}]$ *are sequentially independent and solely depend on the position* $\phi_{\mathcal{R}}(n)$*:*

$$p\left(\mathbf{x}_{\phi_x(n)} | \mathbf{x}_{\phi_x(n-1)}, \ldots, \mathbf{x}_{\phi_x(1)}, \boldsymbol{\phi}(n), \ldots, \boldsymbol{\phi}(1)\right) = p\left(\mathbf{x}_{\phi_x(n)} | \phi_{\mathcal{R}}(n)\right),\ n = 1, \ldots, N. \tag{4.9}$$

3. *The PDFs* $\beta_j(\mathbf{x})$ *are Gaussians* $\mathcal{N}_{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j}(\mathbf{x})$*, parameterized by* $(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ *for all* $j = 1, \ldots, N_{\mathcal{R}}$*.*

Then, the SDTW (Viterbi) distance $D^*[\hat{d}](\boldsymbol{t}, \mathcal{R})$ *that is defined analogous to Equations (2.31) and (2.32) by*

$$D_{\boldsymbol{\Phi}}[\hat{d}](\boldsymbol{t}, \mathcal{R}) = \sum_{n=1}^{N} \hat{d}\left(\mathbf{t}_{\phi_t(n)}, \Delta\boldsymbol{\phi}(n), \mathbf{R}_{\phi_{\mathcal{R}}(n)}\right) \tag{4.10}$$

$$D^*[\hat{d}](\boldsymbol{t}, \mathcal{R}) = D_{\boldsymbol{\Phi}^*}[\hat{d}](\boldsymbol{t}, \mathcal{R}) = \min_{\boldsymbol{\Phi}} \left\{ D_{\boldsymbol{\Phi}}[\hat{d}](\boldsymbol{t}, \mathcal{R}) \right\} \tag{4.11}$$

and the local distance $\hat{d}$ *is given as*

$$\begin{aligned} \hat{d}(\mathbf{t}_i, \Delta\boldsymbol{\phi}, \mathbf{R}_j) = {} & \frac{1}{2}\left(\ln\left(|2\pi\boldsymbol{\Sigma}_j|\right) + (\mathbf{t}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\mathbf{t}_i - \boldsymbol{\mu}_j)\right) \\ & - \ln\left(\alpha_j(\Delta\boldsymbol{\phi})\right), \end{aligned} \tag{4.12}$$

is related to the Viterbi-path optimized likelihood $p(\boldsymbol{t}, \boldsymbol{\Phi}^* | \mathcal{R})$ *by the equation*

$$D^*[\hat{d}](\boldsymbol{t}, \mathcal{R}) = -\ln\left(p(\boldsymbol{t}, \boldsymbol{\Phi}^* | \mathcal{R})\right). \tag{4.13}$$

Proof. Due to prerequisites 1 and 2 we can write

$$P(\boldsymbol{\Phi} | \mathcal{R}) = \prod_{n=1}^{N} P\left(\Delta\boldsymbol{\phi}(n) | \boldsymbol{\phi}_{\mathcal{R}}(n)\right) = \prod_{n=1}^{N} \alpha_{\phi_{\mathcal{R}}(n)}\left(\Delta\boldsymbol{\phi}(n)\right) \tag{4.14}$$

with $\alpha_j(\Delta\boldsymbol{\phi})$ defined as in Equation (4.2) and

$$p(\boldsymbol{x} | \boldsymbol{\Phi}, \mathcal{R}) = \prod_{n=1}^{N} p\left(\mathbf{x}_{\phi_x(n)} | \phi_{\mathcal{R}}(n)\right) = \prod_{n=1}^{N} \beta_{\phi_{\mathcal{R}}(n)}\left(\mathbf{x}_{\phi_x(n)}\right), \tag{4.15}$$

respectively, with $\beta_j(\mathbf{x})$ defined as in Equation (4.6).

The likelihood $p(\boldsymbol{t}, \boldsymbol{\Phi} | \mathcal{R})$ can be developed to

$$\begin{aligned} p(\boldsymbol{t}, \boldsymbol{\Phi} | \mathcal{R}) &= p(\boldsymbol{t} | \boldsymbol{\Phi}, \mathcal{R})\, P(\boldsymbol{\Phi} | \mathcal{R}) \\ &= \prod_{n=1}^{N} \beta_{\phi_{\mathcal{R}}(n)}\left(\mathbf{t}_{\phi_t(n)}\right) \prod_{n=1}^{N} \alpha_{\phi_{\mathcal{R}}(n)}\left(\Delta\boldsymbol{\phi}(n)\right) \\ &= \prod_{n=1}^{N} \mathcal{N}_{\boldsymbol{\mu}_{\phi_{\mathcal{R}}(n)}, \boldsymbol{\Sigma}_{\phi_{\mathcal{R}}(n)}}\left(\mathbf{t}_{\phi_t(n)}\right) \alpha_{\phi_{\mathcal{R}}(n)}\left(\Delta\boldsymbol{\phi}(n)\right). \end{aligned} \tag{4.16}$$

In this remodeling, the first equation is due to the definition of the conditional probability, the second due to the assumption to the Markov property (prerequisite 1 and Equation (4.14)) and the sequential independence of the PDFs (prerequisite 2 and Equation (4.15)) and the third equality refers to the Gaussian assumption (prerequisite 3).

By further taking the negative natural logarithm we get

$$
\begin{aligned}
&-\ln\left(p\left(\boldsymbol{t},\boldsymbol{\Phi}|\mathcal{R}\right)\right)\\
&\quad=-\ln\left(\prod_{n=1}^{N}\mathcal{N}_{\boldsymbol{\mu}_{\phi_{\mathcal{R}}(n)},\boldsymbol{\Sigma}_{\phi_{\mathcal{R}}(n)}}\left(\mathbf{t}_{\phi_t(n)}\right)\alpha_{\phi_{\mathcal{R}}(n)}\left(\Delta\boldsymbol{\phi}\left(n\right)\right)\right)\\
&\quad=\sum_{n=1}^{N}-\ln\left(\mathcal{N}_{\boldsymbol{\mu}_{\phi_{\mathcal{R}}(n)},\boldsymbol{\Sigma}_{\phi_{\mathcal{R}}(n)}}\left(\mathbf{t}_{\phi_t(n)}\right)\right)-\ln\left(\alpha_{\phi_{\mathcal{R}}(n)}\left(\Delta\boldsymbol{\phi}\left(n\right)\right)\right)\\
&\quad=\sum_{n=1}^{N}-\ln\left(\left(\left|\left(2\pi\right)\boldsymbol{\Sigma}_{\phi_{\mathcal{R}}(n)}\right|\exp\left(\left(\mathbf{x}-\boldsymbol{\mu}_{\phi_{\mathcal{R}}(n)}\right)^{T}\boldsymbol{\Sigma}_{\phi_{\mathcal{R}}(n)}^{-1}\left(\mathbf{x}-\boldsymbol{\mu}_{\phi_{\mathcal{R}}(n)}\right)\right)\right)^{-1/2}\right)\\
&\qquad-\ln\left(\alpha_{\phi_{\mathcal{R}}(n)}\left(\Delta\boldsymbol{\phi}\left(n\right)\right)\right)\\
&\quad=\sum_{n=1}^{N}\frac{1}{2}\left(\ln\left(2\pi\left|\boldsymbol{\Sigma}_{\phi_{\mathcal{R}}(n)}\right|\right)+\left(\mathbf{t}_{\phi_t(n)}-\boldsymbol{\mu}_{\phi_{\mathcal{R}}(n)}\right)^{T}\boldsymbol{\Sigma}_{\phi_{\mathcal{R}}(n)}^{-1}\left(\mathbf{t}_{\phi_t(n)}-\boldsymbol{\mu}_{\phi_{\mathcal{R}}(n)}\right)\right)\\
&\qquad-\ln\left(\alpha_{\phi_{\mathcal{R}}(n)}\left(\Delta\boldsymbol{\phi}\left(n\right)\right)\right)
\end{aligned}
\tag{4.17}
$$

The terms in the sum can be summarized with Equation (4.12) and then Equation (4.10) to

$$
\begin{aligned}
-\ln\left(p\left(\boldsymbol{t},\boldsymbol{\Phi}|\mathcal{R}\right)\right) &= \sum_{n=1}^{N}\hat{d}\left(\mathbf{t}_{\phi_t(n)},\Delta\boldsymbol{\phi}\left(n\right),\mathbf{R}_{\phi_{\mathcal{R}}(n)}\right)\\
&= D_{\boldsymbol{\Phi}}[\hat{d}]\left(\boldsymbol{t},\mathcal{R}\right).
\end{aligned}
\tag{4.18}
$$

In the case of $\boldsymbol{\Phi}$ being the Viterbi path $\boldsymbol{\Phi}^*$ the equality to Equation (4.13) is obvious.

□

As a result, one can draw the conclusion that a minimum $D^*[\hat{d}]\left(\boldsymbol{t},\mathcal{R}\right)$ corresponds to a maximum likelihood $p\left(\boldsymbol{t},\boldsymbol{\Phi}^*|\mathcal{R}\right)$.

In fact, the prerequisites 1–3 that are made in this proposition are contestable for general real world applications and thus can be seen as weak points of this generative modeling approach. However, they are the common assumptions pursued in sequence modeling. In particular, they are also presumed with HMMs [Rabiner and Juang, 1993, Schukat-Talamazzini, 1995]. On the other hand, if the prerequisites are accepted, they allow a concise formulation of sequence classification in terms of statistics by Equations (4.10)–(4.12).

To summarize the benefit of the SDTW compared to the DTW distance, it adds value by a statistical modeling of a particular tuple of parameters.

4.3.1.3 Cluster generative SDTW (CSDTW) classification

Finally, this section introduces the CSDTW classification formulation. It shall be assumed that a set $\Re = \left\{\mathcal{R}^{lk}\right\}_{l\in\{1,\ldots,L\},k\in\{1,\ldots,K_l\}}$ of SDTW sub-class models with $\mathcal{R}^{lk} = \left[\mathbf{R}_1^{lk},\ldots,\mathbf{R}_{N_{\mathcal{R}^{lk}}}^{lk}\right]$ and a test pattern $\boldsymbol{t}$ (that corresponds to an isolated character in the HWR context) are given. Each $\mathcal{R}^{lk}$ is aimed to be a generative model for one "compact" cluster in the feature space of class l, hence representing an allograph in the context of HWR.

According to Proposition 4.1 and Equation (4.13), a Viterbi path-optimized ML classification principle is employed, when the classifier decision $\widehat{l}$ is given by

$$\widehat{l} = \underset{l\in\{1,\ldots,L\}}{\operatorname{argmin}} \min_{k\in\{1,\ldots,K_l\}} \left\{ D^*[\hat{d}]\left(\boldsymbol{t},\mathcal{R}^{lk}\right)\right\}. \tag{4.19}$$

A Viterbi path-optimized MAP classification would include prior probabilities $P\left(\mathcal{R}^{lk}\right)$ and maximize $P\left(\boldsymbol{t},\boldsymbol{\Phi}^*|\mathcal{R}^{lk}\right)P\left(\mathcal{R}^{lk}\right)$. It can easily be shown that the classification rule becomes with this background

$$\widehat{l} = \underset{l\in\{1,\ldots,L\}}{\operatorname{argmin}} \min_{k\in\{1,\ldots,K_l\}} \left\{ D^*[\hat{d}]\left(\boldsymbol{t},\mathcal{R}^{lk}\right) - \ln P\left(\mathcal{R}^{lk}\right)\right\}. \tag{4.20}$$

However, experiments have shown that for the present online HWR data the ML criterion achieves higher recognition rates than the MAP criterion.

Like in the vector data case the true parameters α_j^{lk}, β_j^{lk} and $P\left(\mathcal{R}^{lk}\right)$ are generally not known in real world applications, but have to be estimated with help of a set of labeled training observations $\mathbb{X}$. Section 4.3.2 describes a solution to this problem.

As noted in Section 2.3.1, in some situations (e.g. if data comes from sensors with different sampling rates) it is favorable to normalize the score $D^*[\hat{d}]\left(\boldsymbol{t},\mathcal{R}^{lk}\right)$ by the Viterbi path length N^*. Indeed, this strategy has been shown to be lucrative also in CSDTW classification, hence instead of Equation (4.19) the variation

$$\widehat{l} = \underset{l\in\{1,\ldots,L\}}{\operatorname{argmin}} \min_{k\in\{1,\ldots,K_l\}} \left\{ \tilde{D}^*[\hat{d}]\left(\boldsymbol{t},\mathcal{R}^{lk}\right)\right\} \tag{4.21}$$

is the basis of the CSDTW classification.

In this respect, Equation (4.21) in combination with Equations (2.33), (4.10), (4.11) and (4.12) define the framework for the CSDTW classification. Figure 4.2 shows a graphical illustration of a sample character classification.

A notable connection to the 1-NN classifier (Section 2.2.4) is obvious by a comparison of Equations (4.21) and (2.30). In this sense, the allograph models of the CSDTW-classification correspond to the training examples in the 1-NN classifier. Thus, CSDTW can be seen to condense a cluster of training observations into one single statistical model.

4.3.1.4 DTW, SDTW and HMM — a unifying view

The concepts of DTW, SDTW and HMM share common aspects, which will be pointed out in the following.

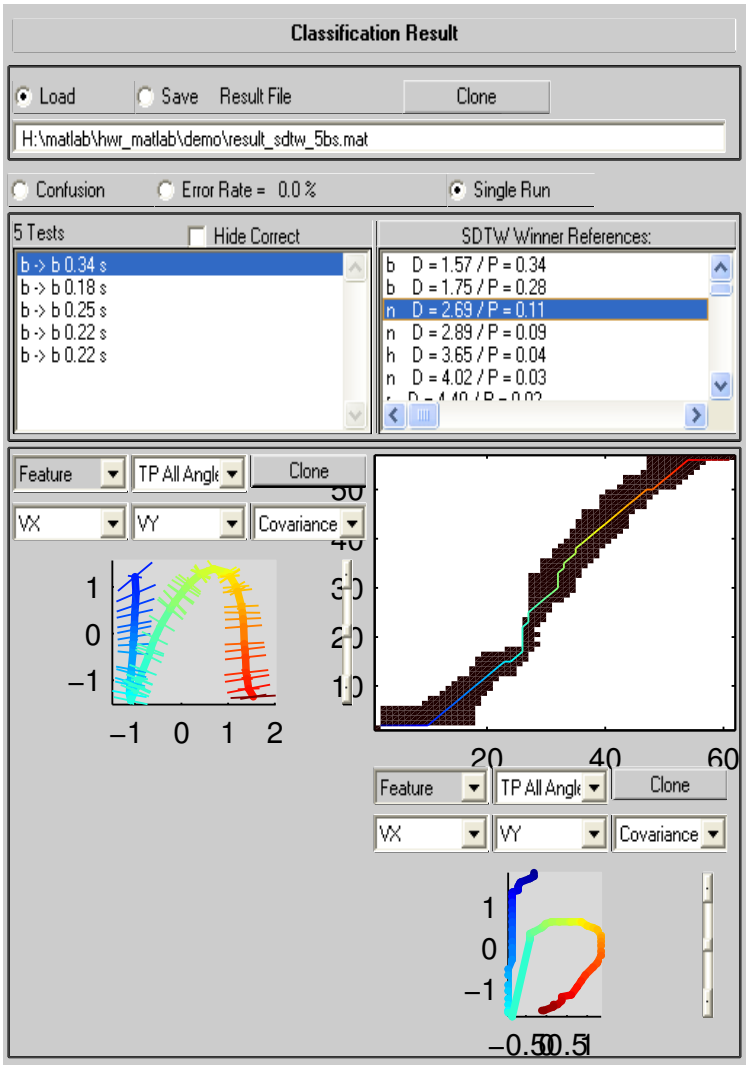


FIGURE 4.2: The classification of a test pattern t (of class "b"; illustrated in the lower right corner). The list on the right shows labels of reference models $\mathcal{R}^{lk}$, sorted by the CSDTW Viterbi distance $\tilde{D}^*[\hat{d}]\left(t, \mathcal{R}^{lk}\right)$. Since a "b" is on top of the list, t is correctly classified. The rectangular area below the list illustrates the alignment of t with the third best match reference (a reference for the character "n", illustrated to the left in a similar way as in Figure 4.1). White areas indicate regions, which were pruned by a beam search criterion (cf. Section 4.3.1.5). The Viterbi alignment path is illustrated by the sketched line: It traverses all aligned sample point pairs. Corresponding points in the "b", "n" and the Viterbi path are coded in the same color. The reader can see apparent temporal distortions at the beginning, the center and the end region of the Viterbi alignment. The CSDTW Viterbi distance is $\tilde{D}^*[\hat{d}]\left(t, \mathcal{R}^{lk}\right) = 2.69$ for this particular alignment.

DTW and SDTW A valuable insight concerning DTW and SDTW is given by the observation that $\tilde{D}^*[\tilde{d}]\left(\boldsymbol{t}, \boldsymbol{r}\right)$ is an exact specialization of $\tilde{D}^*[\hat{d}]\left(\boldsymbol{t}, \mathcal{R}\right)$.[1] With the identifications

$$\mathbf{r}_j = \boldsymbol{\mu}_j, \tag{4.22}$$

$$\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_j \tag{4.23}$$

and

$$\alpha_j\left(\Delta\boldsymbol{\phi}\right) = 1/\left|\mathbb{P}\right| \tag{4.24}$$

the matching of the local distances of Equations (4.1) and (4.12) can easily be verified. Further, the warping capabilities defined by $\mathbb{P}$ do not differ between DTW and SDTW, resulting in an exact match of $\tilde{D}^*[\tilde{d}]\left(\boldsymbol{t}, \boldsymbol{r}\right)$ and $\tilde{D}^*[\hat{d}]\left(\boldsymbol{t}, \mathcal{R}\right)$ in the case of the identifications seen above. The only difference of $\tilde{D}^*[\tilde{d}]\left(\boldsymbol{t}, \boldsymbol{r}\right)$ and $\tilde{D}^*[\hat{d}]\left(\boldsymbol{t}, \mathcal{R}\right)$ is the statistical treatment of the feature space $\mathbb{R}^F$ and the transitions $\mathbb{P}$ for each sample point with SDTW.

SDTW and HMM HMMs can be formulated equivalently to SDTW. In the context of the HMM nomenclature introduced in Section 2.3.2 the HMM *states* q_j, *state transitions* (q_n, q_{n+1}), *Viterbi state sequence* $\boldsymbol{q}^*$, *state transition probabilities* a_{ij} and *observation functions* $b_j\left(\mathbf{x}\right)$ have correspondences in the index j of the reference sequence, $\Delta\boldsymbol{\phi}\left(n+1\right)$, $\boldsymbol{\Phi}^*$, $\alpha_j\left(\Delta\boldsymbol{\phi}\right)$ and $\beta_j\left(\mathbf{x}\right)$, respectively. The starting point constraint $\boldsymbol{\phi}\left(1\right) = \left(1,1\right)$ (cf. Section 2.3.1) corresponds to the particular *HMM state prior probabilities* $\boldsymbol{\pi} = \left(1,0,0,\ldots,0\right)^T$. Both concepts calculate the corresponding likelihoods $p\left(\boldsymbol{t}, \boldsymbol{q}^*|\lambda\right)$ and $p\left(\boldsymbol{t}, \boldsymbol{\Phi}^*|\mathcal{R}\right)$, with respect to the underlying modeling assumptions.

However, to achieve full equivalence of SDTW and HMM, the concept of *null transitions* must be included into the HMM framework. Null transitions in HMMs allow a step in the state sequence without the emission of an observation. Hence, they correspond to the transition $\Delta\boldsymbol{\phi} = \left(0,1\right)$ in the SDTW distance computation. They have been introduced in HMMs of the IBM speech recognizer [Bahl et al., 1983], but most common HMM implementations in HWR systems do not employ this concept. Instead, typically *linear* (corresponding to $\mathbb{P} = \left\{\left(1,0\right),\left(1,1\right)\right\}$) or *Bakis* (corresponding to $\mathbb{P} = \left\{\left(1,0\right),\left(1,1\right),\left(1,2\right)\right\}$) transition models [Rabiner and Juang, 1993] are assumed in these systems.

For SDTW the specific choice of the Sakoe-Chiba transitions $\mathbb{P} = \left\{\left(1,0\right),\left(0,1\right),\left(1,1\right)\right\}$ and thus a "null-transition"-like modeling is very important for the following reasons:

1. Because of the flexibility of the alignment paths, the length of the reference sequences $\mathcal{R}$ in SDTW can be *arbitrary*. This is distinct from HMMs without null transitions, e.g., with $\mathbb{P} = \left\{\left(1,0\right),\left(1,1\right)\right\}$. There, the model length must not exceed the length of the test pattern due to the compulsory step in the test sequence. The arbitrariness of the reference length will allow an automatic, data-driven dimensioning of the CSDTW reference models, as will be shown in Section 4.3.2.2.

[1] The same deliberations in this section hold for the non-normalized variants $D^*[\tilde{d}]\left(\boldsymbol{t}, \boldsymbol{r}\right)$ and $D^*[\hat{d}]\left(\boldsymbol{t}, \mathcal{R}\right)$, however, for simplicity reasons only the normalized DTW and SDTW distances will be explicitly pursued.

2. Since DTW and SDTW are defined on the same set of transitions $\mathbb{P}$, the global warping capability does not differ between them. This behavior is indispensable in a scenario where DTW and SDTW shall model a related measure of distance. In fact, CSDTW uses the DTW distance for clustering (cf. Section 4.3.2.1) and the SDTW distance during classification (cf. Section 4.3.1.3) and statistical parameter estimation (cf. Section 4.3.2.2).

3. The symmetric transitions $\Delta\boldsymbol{\phi}$ imply a *symmetric* set of possible alignments $\boldsymbol{\Phi}$ in the search space. A solution for $\boldsymbol{\Phi}^*$ is neither biased to the test nor the reference pattern. This property is also convenient for the definition of a distance measure $\tilde{D}^*[\bar{d}]\left(\mathcal{T}, \mathcal{R}\right)$ between two CSDTW reference models $\mathcal{T}$ and $\mathcal{R}$, as it will been studied in Chapter 6.

In this section the connections between SDTW and HMM have been illuminated. These considerations should help to transfer further development from one of these concepts to the other one. In particular, HMM refinements like state tying, state duration modeling, etc. [Rabiner and Juang, 1993] can straightforwardly be applied to SDTW and thus CSDTW. It is also worth noting that SDTW is not restricted to Gaussian distributions, but can also handle a mixture of densities or discrete probabilities.

So, as a SDTW model is equivalent to an HMM with null transitions, one might argue that it would be favorable to use the wide spread HMM formulation rather than to introduce a new concept by means of SDTW. An answer to this concern is as follows: The formulation of SDTW retains a direct connection to DTW. In particular, it has been shown that a specifically modified DTW distance is an incarnation of the SDTW distance. This relation will help with respect to the formulation of a holistic combination of the cluster analysis and statistical sequence modeling.

4.3.1.5 Implementation related issues

Also with the computation of the SDTW distance it is advantageous to apply DP and beam search in order to achieve a complexity reduction. Deviating from the standard beam search Algorithm 2, two modifications and extensions have been applied in the implemented system.

The first modification concerns the strategy, at what stage concurrent hypotheses are compared. Most beam search algorithms in the HMM context — also Algorithm 2 — handle this issue *frame-synchronous*, i.e., they compare and eliminate hypotheses along a line $\phi_t\left(n\right) = k$ (each dashed vertical line in Figure 2.8(b)). This strategy is well motivated by the fact that for the typical choice of transitions in HMMs such as $\mathbb{P} = \{(1,0), (1,1)\}$, all hypotheses have an equal amount of remaining transition steps to the endpoint $(N_t, N_{\mathcal{R}})$ of the path.

However, for the symmetric transitions of Equation (2.33) it is advantageous when the pruning strategy reflects the symmetry of the warping space. One solution and our choice is to compare and prune hypotheses *synchronous with respect to the matrix diagonals*, i.e., the lines $\phi_t\left(n\right) + \phi_{\mathcal{R}}\left(n\right) = k$ (compare Figure 4.3). Algorithm 4 summarizes the complete diagonal synchronous beam search algorithm.

Algorithm 4 Diagonal synchronous beam search

Variables:

$\mathcal{A}_m$: *list of active states in diagonal* m
D: *new accumulated distance*
D_{ij}: *smallest distance for matrix element* (i, j)
D_m: *smallest distance for diagonal* m
s_{beam}: *pruning threshold*

$\mathcal{A}_1 := [1]$;
$D_{ij} := \infty, \forall i = 1, \ldots, N_t, j = 1 \ldots, N_r$;
$D_{(1)(1)} := D_1 := d(\mathbf{t}_1, \mathbf{r}_1)$
$D_m := \infty, \forall m = 2, \ldots, N_t + N_r - 1$;
for $m = 1, \ldots, N_t + N_r - 2$ *% diagonals*
  for all $j \in \mathcal{A}_m$ *% active states*
    for all $\Delta\phi \in \mathbb{P}$ *% transitions*
      $(k, l) := (i, j) + \Delta\phi$; *% goal matrix element of* $\Delta\phi$
      $n := k + l - 1$; *% goal diagonal of* $\Delta\phi$
      $D := D_{ij} + d(\mathbf{t}_k, \mathbf{r}_l)$; *% new accumulated distance*
      if $D > D_n + s_{\text{beam}}$ *% distance exceeds beam criterion*
        continue; *% with next* $\Delta\phi$
      if $(l \in \mathcal{A}_n) \wedge (D > D_n)$ *% another path hypothesis has smaller distance*
        continue; *% with next* $\Delta\phi$
      $D_{kl} := D$; *% D is new smallest distance for matrix element* (k, l)
      if $D < D_n$ *% D is new smallest distance for diagonal* n
        $D_n := D$;
      if $(l \notin \mathcal{A}_n)$
        if $(D < D_n)$
          $\mathcal{A}_n := [l] + +\mathcal{A}_n$; *% prepend l to list of active states of diagonal* n
        else
          $\mathcal{A}_n := \mathcal{A}_n + + [l]$; *% append l to list of active states of diagonal* n

$D^*[d](\boldsymbol{t}, \boldsymbol{r}) := D_{N_t+N_r-1}$;

The second extension addresses a boost of efficiency when comparing multiple hypotheses at a character level: The beam search strategy described above is applied for an evaluation of Equation (4.11) *inside* the minimization of Equation (4.19). In *frog on hand*, similar pruning strategies are employed *across* multiple evaluations of Equation (4.11) in the minimization of Equation (4.19). In this respect, hypotheses for the computation of the particular $\mathcal{R}^{lk}$-Viterbi matrix $D^*[\hat{d}](\boldsymbol{t}, \mathcal{R}^{lk})$ are regularly compared with hypotheses for the Viterbi matrices of the other allograph models $D^*[\hat{d}](\boldsymbol{t}, \mathcal{R}^{l'k'})$. If $\mathcal{R}^{lk}$ is unlikely to be the optimal solution in that context, the computation of $D^*[\hat{d}](\boldsymbol{t}, \mathcal{R}^{lk})$ is aborted.

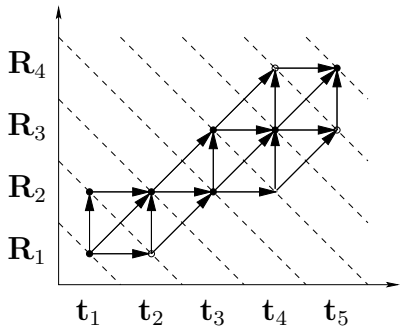


FIGURE 4.3: Diagonal synchronous beam search: Contrary to the frame synchronous beam search, the pruning is performed synchronous with respect to the matrix diagonals, in order to account for the symmetry of the transitions $\mathbb{P} = \{(1,0),(0,1),(1,1)\}$ (compare to Figure 2.8).

4.3.2 CSDTW training

This section describes the learning of the statistical allograph models $\Re = \left\{\mathcal{R}^{lk}\right\}_{l\in\{1,\ldots,L\},k\in\{1,\ldots,K_l\}}$. As motivated previously, CSDTW aims to combine two complementary strategies in order to cope with data variations. On a higher level, distinct writing styles are explicitly separated into sub-classes. On a lower level, each of these writing styles is statistically modeled. A solution for these two issues is incorporated in the CSDTW training. To anticipate the idea in a few words, the first issue is treated by hierarchical cluster analysis, the second by maximum-likelihood (ML) parameter estimation. A thorough description will be given in the following.

As CSDTW is a generative approach, the training can be performed for each class independently. Given a set of data examples, provided with the corresponding character class labels, CSDTW training gives solutions to the following problems:

1. Generate allograph clusters $\mathbb{C}^{lk}, k = 1,\ldots K_l$ of the training set of class l. Since neither the data is labeled with cluster memberships, nor we want to label clusters by hand, this part corresponds to an unsupervised learning task.

2. Estimate a CSDTW reference model $\mathcal{R}^{lk}$ for each cluster $\mathbb{C}^{lk}$.

4.3.2.1 Generation of the allograph clusters

Techniques for clustering, in particular the agglomerative hierarchical clustering algorithm have been reviewed in Section 2.4. Here, it is applied as a solution for step 1 in the CSDTW training.

In CSDTW a cluster is modeled by unimodal Gaussian probability densities. In this respect, it is favorable to assume compact clusters rather than elongated or shell-shaped ones. Further, since the number of clusters may be different for each class and not known a priori, a flexible treatment of the clustering granularity is desired. The agglomerative hierarchical clustering fulfills these requirements.

Some issues of the agglomerative hierarchical clustering have to be specifically adapted to the present sequence data. It will be described in the following and, for convenience, the order of the issues is kept the same as in Section 2.4.

Point and cluster dissimilarities CSDTW aims to reveal clusters based on a distance, which is consistent with the distance measure in classification, i.e., $\tilde{D}^*[\hat{d}]\,(\boldsymbol{t}, \mathcal{R})$. As argued above, the modified DTW Viterbi distance $\tilde{D}^*[\tilde{d}]\left(\boldsymbol{x}^{(k)}, \boldsymbol{x}^{(k')}\right)$ satisfies this claim and thus is our favorable choice for the *point dissimilarity* in the cluster space,

$$D\left(\boldsymbol{x}^{(k)}, \boldsymbol{x}^{(k')}\right) = \tilde{D}^*[\tilde{d}]\left(\boldsymbol{x}^{(k)}, \boldsymbol{x}^{(k')}\right). \tag{4.25}$$

The *cluster dissimilarity* function $D\left(\mathbb{C}^k, \mathbb{C}^{k'}\right)$ shall be based on the point dissimilarity of the cluster elements. As argued, the *average dissimilarity* gives a favor to compact clusters — which is the aim in the context of the Gaussian PDF models — and achieved best recognition results in our experiments.

Cluster representatives As will be claimed in Section 4.3.2.2, CSDTW uses examples from the domain of training sequences for initialization of the iterative parameter estimation. Hence, *point representatives* $\tilde{\boldsymbol{x}}^{(k)}$ rather than hyperplane or hyperspherical representatives are a reasonable choice. Further, the *median center* is compatible with the suggestion of Theodoridis and Koutroumbas [1999], as Equation (4.25) is not a metric.

Number of clusters An example of a dissimilarity dendrogram for the clustering of five lower case characters "b" is shown in Figure 2.9. The reader can get an idea that a sensible range of settings for $D_{\max}$ could roughly lie between 2 and 20.

Pruning clusters We experienced that it is beneficial to eliminate clusters, the cardinality of which is smaller than a threshold $O_{\min}$. This benefit presumably arises from the following two reasons. First, small clusters are likely to be produced by data outliers. In the *UNIPEN* database outliers are quite often present due to noisy and mislabeled data. Second, the statistical parameter estimation, as will be introduced in the following section, is not robust for small clusters and should be avoided for those.

4.3.2.2 Estimation of the statistical model parameters

The outcome of the last section is a set of clusters $\left\{\mathbb{C}^{lk}\right\}_{l\in\{1,\ldots,L\},k\in\{1,\ldots,K_l\}}$. In the sense of Section 4.3.1.2, $\mathbb{C}^{lk}$ is to be modeled by $\mathcal{R}^{lk} = \left[\mathbf{R}_1^{lk}, \ldots, \mathbf{R}_{N_{\mathcal{R}^{lk}}}^{lk}\right]$. Assuming a fixed set of $N_{\mathcal{R}^{lk}}$, the parameters to be estimated are: the discrete transition probabilities $\alpha_j^{lk}\,(\Delta\boldsymbol{\phi})$ for all $\Delta\boldsymbol{\phi} \in \mathbb{P}$, the mean $\boldsymbol{\mu}_j^{lk}$ and the covariances $\boldsymbol{\Sigma}_j^{lk}$.

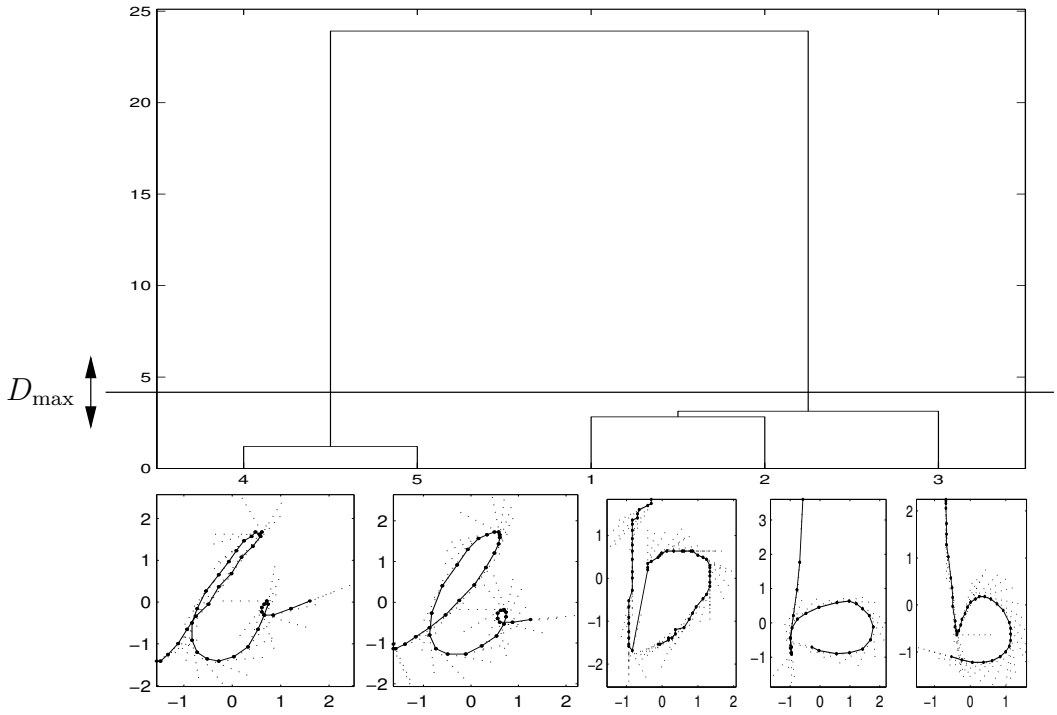


FIGURE 4.4: A dissimilarity dendrogram of a clustering of five characters "b". Each leaf in the tree represents the below positioned character pattern; a node denotes a merge step of two particular clusters in algorithm 3. The level of the node on the ordinate corresponds to the cluster dissimilarity $D\left(\mathbb{C}^k, \mathbb{C}^{k'}\right)$, when merging. The choice as is made in the figure will generate two clusters. A specific value for $D_{\max}$ determines the granularity of the final clustering. For an interpretation of the character features refer to Figure 3.2 (b).

A common solution for the estimation problem is to pursue the maximum likelihood (ML) approach. In the CSDTW context, ML seeks parameters $\mathcal{R}^{lk}$ that maximize the objective

$$\mathcal{L}\left(\mathcal{R}^{lk}\right) = \sum_{\boldsymbol{x} \in \mathbb{C}^{lk}} \ln P\left(\boldsymbol{x} | \mathcal{R}^{lk}\right). \tag{4.26}$$

This task already has been tackled in the HMM context (cf. Section 2.3.2) and we can benefit from approved optimization algorithms. For its simplicity and speed, the decision directed *Viterbi training* (also known as *segmental K-means algorithm*), which instead of $\mathcal{L}\left(\mathcal{R}^{lk}\right)$ maximizes the Viterbi path-optimized variant

$$\mathcal{L}^*\left(\mathcal{R}^{lk}\right) = \sum_{\boldsymbol{x} \in \mathbb{C}^{lk}} \ln P\left(\boldsymbol{x}, \boldsymbol{\Phi}^*_{\boldsymbol{x},\mathcal{R}^{lk}} | \mathcal{R}^{lk}\right), \tag{4.27}$$

is a beneficial choice. This approach gives estimates for the model parameters in an iteration of two basic steps.

Given an initial guess for $\hat{\mathcal{R}}^{lk}$, step *one* computes the Viterbi alignments $\boldsymbol{\Phi}^*_{\boldsymbol{x},\mathcal{R}^{lk}}$ for all $\boldsymbol{x} \in \mathbb{C}^{lk}$. In step *two*, an ML parameter re-estimation of $\hat{\boldsymbol{\mu}}^{lk}_j$ and $\hat{\boldsymbol{\Sigma}}^{lk}_j$ is performed, based on

the set $\mathbb{Z}_j^{lk}$ of sample points that have been aligned to j in step *one*:

$$\hat{\boldsymbol{\mu}}_j^{lk} = \frac{1}{\left|\mathbb{Z}_j^{lk}\right|} \sum_{(\mathbf{x},n,\boldsymbol{\mathcal{X}})\in\mathbb{Z}_j^{lk}} \mathbf{x} \tag{4.28}$$

$$\hat{\boldsymbol{\Sigma}}_j^{lk} = \frac{1}{\left|\mathbb{Z}_j^{lk}\right| - 1} \sum_{(\mathbf{x},n,\boldsymbol{\mathcal{X}})\in\mathbb{Z}_j^{lk}} \left(\mathbf{x} - \hat{\boldsymbol{\mu}}_j^{lk}\right)\left(\mathbf{x} - \hat{\boldsymbol{\mu}}_j^{lk}\right)^T \tag{4.29}$$

with

$$\mathbb{Z}_j^{lk} = \left\{(\mathbf{x}_i, n, \boldsymbol{\mathcal{X}}) \,|\boldsymbol{\phi}^*_{\boldsymbol{\mathcal{X}},\mathcal{R}^{lk}}(n) = (i,j)\,,\, \forall n = 1,\ldots,N_{\boldsymbol{\mathcal{X}},\mathcal{R}^{lk}},\, \forall \boldsymbol{\mathcal{X}} = [\mathbf{x}_1,\ldots,\mathbf{x}_{N_x}] \in \mathbb{C}^{lk}\right\}, \tag{4.30}$$

where $N_{\boldsymbol{\mathcal{X}},\mathcal{R}^{lk}}$ is the length of $\boldsymbol{\Phi}^*_{\boldsymbol{\mathcal{X}},\mathcal{R}^{lk}}$.

The transition probability estimates $(\hat{\alpha}')_j^{lk}(\Delta\boldsymbol{\phi})$ are calculated by counting the number of transitions taken with respect to the Viterbi alignments and a following normalization:

$$(\tilde{\alpha}')_j^{lk}(\Delta\boldsymbol{\phi}) = \left|\tilde{\mathbb{Z}}_j^{lk}(\Delta\boldsymbol{\phi})\right| \tag{4.31}$$

$$(\hat{\alpha}')_j^{lk}(\Delta\boldsymbol{\phi}) = \frac{(\tilde{\alpha}')_j^{lk}(\Delta\boldsymbol{\phi})}{\sum_{\Delta\boldsymbol{\phi}\in\mathbb{P}}(\tilde{\alpha}')_j^{lk}(\Delta\boldsymbol{\phi})} \tag{4.32}$$

with

$$\tilde{\mathbb{Z}}_j^{lk}(\Delta\boldsymbol{\phi}) = \left\{(\mathbf{x}_i, n, \boldsymbol{\mathcal{X}}) \,|\Delta\boldsymbol{\phi}^*_{\boldsymbol{\mathcal{X}},\mathcal{R}^{lk}}(n) = \Delta\boldsymbol{\phi} \wedge \boldsymbol{\phi}^*_{\boldsymbol{\mathcal{X}},\mathcal{R}^{lk}}(n) = (i,j)\,,\, \forall n = 1,\ldots,N_{\boldsymbol{\mathcal{X}},\mathcal{R}^{lk}},\, \forall \boldsymbol{\mathcal{X}} = [\mathbf{x}_1,\ldots,\mathbf{x}_{N_x}] \in \mathbb{C}^{lk}\right\}.$$

The re-estimated parameters $\hat{\boldsymbol{\mu}}_j^{lk}$, $\hat{\boldsymbol{\Sigma}}_j^{lk}$ and $(\hat{\alpha}')_j^{lk}(\Delta\boldsymbol{\phi})$ then define a new model $\hat{\mathcal{R}}^{lk}$ and the iteration is repeated.

While the Viterbi training is well defined, theoretically founded and successfully used in many situations, an open question in the context of the parameter estimation affects the problem of *dimensioning* and *initializing* $\mathcal{R}^{lk}$.

The problem of dimensioning is important, since the number of model states, i.e., $N_{\mathcal{R}^{lk}}$ is not included in the ML optimization and has to be specified manually beforehand. On the other hand, the problem of initialization is important particularly with regard to the fact that the iterative training gives a *local* optimum to the ML criterion. The quantities among $\mathcal{R}^{lk}$, that are particularly sensitive to a proper initialization, are the parameters specifying the probability density function [Rabiner and Juang, 1993, Chapter 6.12], i.e., $\boldsymbol{\mu}_j^{lk}$ and $\boldsymbol{\Sigma}_j^{lk}$ in our context.

CSDTW solves this problem by a unique, data-driven procedure. As the clustering was designed for generating compact clusters and point cluster representatives, the latter, in particular the median centers $\tilde{\boldsymbol{\mathcal{X}}}^{lk} = \left[\tilde{\mathbf{x}}_1^{lk},\ldots,\tilde{\mathbf{x}}_{N_{\tilde{\mathcal{X}}^{lk}}}^{lk}\right]$, are well suited for dimensioning and initialization. Thus, the CSDTW parameter estimation initializes:

1. Transition probabilities

$$(\alpha')_j^{lk}(\Delta\boldsymbol{\phi}) = 1/\left|\mathbb{P}\right|, \forall j = 1,\ldots,N_{\tilde{\mathcal{X}}^{lk}}, \forall \Delta\boldsymbol{\phi} \in \mathbb{P} \tag{4.33}$$

2. Mean

$$\boldsymbol{\mu}_j^{lk} = \tilde{\mathbf{x}}_j^{lk}, \forall j = 1,\ldots,N_{\tilde{\mathcal{X}}^{lk}} \tag{4.34}$$

3. Covariance

$$\boldsymbol{\Sigma}_j^{lk} = \boldsymbol{\Sigma}, \forall j = 1,\ldots,N_{\tilde{\mathcal{X}}^{lk}} \tag{4.35}$$

(cf. Section 4.3.1.4 for an explanation of $\boldsymbol{\Sigma}$)

Also in that regard, $N_{\mathcal{R}^{lk}}$ is implicitly fixed by this prescription to $N_{\tilde{\mathcal{X}}^{lk}}$. Thanks to the Sakoe-Chiba transitions $N_{\mathcal{R}^{lk}}$ can be arbitrary, contrary to HMM implementations without null-transitions. Algorithm 5 summarizes the parameter estimation with the Viterbi training for a cluster $\mathbb{C}^{lk}$.

Algorithm 5 CSDTW Viterbi training

Variables:

S: *number of iterations*

$\hat{\mathcal{R}}^{lk}$: *actual estimate of reference model for cluster* $\mathbb{C}^{lk}$

1: Initialize $\hat{\mathcal{R}}^{lk}$ according to Equations (4.33)–(4.35)

2: **for** $s = 1,\ldots,S$ *% iteration*

3: determine $\boldsymbol{\Phi}^*_{\mathcal{X},\hat{\mathcal{R}}^{lk}}$ for all $\mathcal{X} \in \mathbb{C}^{lk}$ *% compute Viterbi alignments*

4: re-estimate parameters $\hat{\boldsymbol{\mu}}_j^{lk}$, $\hat{\boldsymbol{\Sigma}}_j^{lk}$ and $(\hat{\alpha}')_j^{lk}(\Delta\boldsymbol{\phi})$ with help of Equations (4.28)–(4.32)

Note that with the proposed initialization the SDTW Viterbi distances $\tilde{D}^*[\hat{d}]\left(\mathcal{X}, \mathcal{R}^{lk}\right),\ \mathcal{X} \in \mathbb{C}^{lk}$ in the first Viterbi training iteration are equivalent to the DTW clustering dissimilarities $\tilde{D}^*[\tilde{d}]\left(\mathcal{X}, \tilde{\mathcal{X}}^{lk}\right),\ \mathcal{X} \in \mathbb{C}^{lk}$, as can be seen by a comparison of Equations (4.1) and (4.12). This connection additionally reveals the attractive holistic property, regarding a seamless integration of clustering and statistical parameter estimation in CSDTW.

Finally, it should be mentioned that other training algorithms like the Baum-Welch parameter re-estimation as well as discriminative approaches (maximum mutual information, MMI, minimum classification error, MCE) [Rabiner and Juang, 1993, Juang et al., 1997] can be employed instead of the Viterbi training, if desired.

4.4 Analysis of complexity

An SDTW distance evaluation $\tilde{D}^*[\hat{d}]\left(\boldsymbol{t}, \mathcal{R}\right)$ using beam search has the asymptotic complexity

$$C_{\text{Time}}\left(\tilde{D}^*[\hat{d}]\left(\boldsymbol{t}, \mathcal{R}\right)\right) = \mathcal{O}(\tilde{N} \cdot \left|\mathbb{P}\right| \cdot F^2), \tag{4.36}$$

with $\tilde{N}$ the average length of sequences $\mathcal{R}^{lk}$ and $\boldsymbol{t}$. $F\,(=3\,\text{in the present case})$ equals the feature space dimension and $\mathcal{O}(F^2)$ corresponds to the evaluation of Equation (4.12). Experimentally, for $\tilde{N}=42$ the evaluation time was measured as

$$\text{Time}\left(\tilde{D}^*[\hat{d}]\,(\boldsymbol{t},\mathcal{R})\right)\approx 0.0005\,\text{sec} \tag{4.37}$$

in a *C++* implementation on an AMD Athlon 1600MHz.

For a model size (= total number of generated allographs models) $A^{\text{tot}}=600$ the classification time is

$$\begin{aligned}\text{Time}\,(\text{Classification}) &\approx A^{\text{tot}}\cdot\text{Time}\left(\tilde{D}^*[\hat{d}]\,(\boldsymbol{t},\mathcal{R})\right)\\ &= 600\cdot 0.0005\,\text{sec}=0.3\,\text{sec},\end{aligned} \tag{4.38}$$

across-model beam search (as explained in Section 4.3.1.5) further reduces the expense to

$$\text{Time}\,(\text{Classification})\approx 0.13\,\text{sec}.$$

The training complexity $C_{\text{Time}}\,(\text{Train})$ of CSDTW is dominated by the cluster analysis, which is asymptotically

$$C_{\text{Time}}(\text{Cluster})=\mathcal{O}\left(L\cdot\left(\tilde{M}^2\cdot C_{\text{Time}}\left(\tilde{D}^*[\tilde{d}]\,(\boldsymbol{t},\boldsymbol{r})\right)+\tilde{M}^3\right)\right), \tag{4.39}$$

with L the number of classes and $\tilde{M}$ the average number of training patterns in *one* class. The term $\tilde{M}^2\cdot C_{\text{Time}}\left(\tilde{D}^*[\tilde{d}]\,(\boldsymbol{t},\boldsymbol{r})\right)$ corresponds to the computation of the pairwise pattern dissimilarities, $\tilde{M}^3$ for the cluster linkage [Theodoridis and Koutroumbas, 1999]. In the lower case character experiments ($\tilde{M}\approx 1500$) the training time was measured as

$$\text{Time(Train)}\approx 20\,\text{h}. \tag{4.40}$$

Beneficially, training can easily be parallelized into L independent processes.

The memory complexity C_{Memory} basically consists of the storage of all CSDTW reference models. For $A^{\text{tot}}=600$,

$$\begin{aligned}\text{Memory} &\approx A^{\text{tot}}\cdot\tilde{N}\cdot(F+F^2+|\mathbb{P}|\cdot F)\cdot 4\,\text{Byte}\\ &= 600\cdot 42\cdot(3+9+9)\cdot 4\,\text{Byte}=2067\,\text{KByte}\end{aligned} \tag{4.41}$$

without compression.

4.5 CSDTW word classification

In *frog on hand*, an expansion of the CSDTW character recognition to the recognition of words has been studied by Simon [2003], with inspiration from HWR and speech recognition literature [Plamondon and Srihari, 2000, Deshmukh et al., 1999, Dengel et al., 1997, Manke et al., 1996, Ney, 1984]. A detailed description of the implementation is given in the above mentioned work. The following sections shall sketch the idea.

4.5.1 The most probable super reference

The problem of word classification corresponds to the uncovering of the true character symbol sequence

$$\mathcal{l} = [l_1, \ldots, l_S]\,, \tag{4.42}$$

which has generated the writing under consideration, or in the context of classification, the writings sequence $\boldsymbol{t}$ of feature vectors, respectively. Note that, contrary to the previous convention, $\boldsymbol{t}$ appears here in the semantic of an entire word.

Principally, the strategy of the pursued word classification aims at the matching of $\boldsymbol{t}$ with concatenations (also called *super reference*s) $\mathfrak{R}^\tau$ of allograph references $\mathcal{R}^{lk}$, that is,

$$\mathfrak{R}^\tau = \mathcal{R}^{\tau(1)} \oplus \ldots \oplus \mathcal{R}^{\tau(S)}. \tag{4.43}$$

For simplicity, a fixed super reference length S is assumed in this formulation. The mapping

$$\tau : \mathbb{N} \rightarrow \{1, .., L\} \times \{1, .., K\} \tag{4.44}$$

with $K = \max_l \{K_l\}$ identifies the sequence of character allographs $\mathcal{R}^{lk}$ that are concatenated for the formation of $\mathfrak{R}^\tau$. Clearly, the knowledge of the true $\mathfrak{R}^\tau$ (or equivalently τ) implies $\mathcal{l}$.

For a solution to the classification problem, Simon adopts a statistical point of view and assumes that a-posteriori probabilities $P\left(\mathfrak{R}^\tau|\boldsymbol{t}\right)$ are accessible for the super reference candidates $\mathfrak{R}^\tau$. Under this assumption, and following the above (cf. Section 2.2.1.1) addressed considerations from decision theory, the optimal classifier decides, given $\boldsymbol{t}$, on the most probable super reference

$$\hat{\mathfrak{R}^\tau} = \underset{\mathfrak{R}^\tau}{\operatorname{argmax}} \left\{P\left(\mathfrak{R}^\tau|\boldsymbol{t}\right)\right\}. \tag{4.45}$$

Most often, $P\left(\mathfrak{R}^\tau|\boldsymbol{t}\right)$ can not directly be estimated, but — like in previous explanations — via its decomposition into the product of a-priori probability and likelihood using the Bayes rule:

$$\hat{\mathfrak{R}^\tau} = \underset{\mathfrak{R}^\tau}{\operatorname{argmax}} \left\{\frac{P\left(\mathfrak{R}^\tau\right) p\left(\boldsymbol{t}|\mathfrak{R}^\tau\right)}{p\left(\boldsymbol{t}\right)}\right\} = \underset{\mathfrak{R}^\tau}{\operatorname{argmax}} \quad \Big\{\underbrace{P\left(\mathfrak{R}^\tau\right)}_{\text{Linguistic model}} \quad \underbrace{p\left(\boldsymbol{t}|\mathfrak{R}^\tau\right)}_{\text{Appearance model}}\Big\}\,. \tag{4.46}$$

This decomposition provides a further, intuitive insight into the word recognition problem, with which it can be regarded as two decoupled challenges: *First*, like in the character recognition context, the term $p\left(\boldsymbol{t}|\mathfrak{R}^\tau\right)$ gives a leverage for the statistical modeling of the *appearance* of the handwriting, that is, $\boldsymbol{t}$.

Second, the explicit appearance of word prior probabilities $P\left(\mathfrak{R}^\tau\right)$ in Equation (4.46) represents a starting point for incorporating the constraints of the domain *linguistic*.

The importance of this linguistic modeling is particularly apparent as the computational complexity of Equation (4.46) is proportional to the number of $\mathfrak{R}^\tau$ candidates, which — without constraint — increases exponentially with the sequence length S. For instance, in the

case of $A^{\text{tot}} = 100$ allographs and a sequence length $S = 10$ a total number of $100^{10} = 10^{20}$ different super references can be formed, which is intractable for a practical evaluation.

However, in natural languages, super references with $P(\mathfrak{R}^\tau) = 0$ are very frequent, thus the costly evaluation of their appearance model $p(\boldsymbol{t}|\mathfrak{R}^\tau)$ can be skipped with this prior knowledge. As another benefit, the reduction of the classification problem to fewer classes reduces the risk of misclassifications.

Existing solutions for an implementation of the domain linguistic are predominantly founded on two techniques: *n-grams* [Ney, 1992], which aim at an estimation of the conditional character probabilities $P(l_i|l_{i-1}, \ldots, l_{i-n+1})$, and *lexica* (or *dictionaries*). The work of Simon [2003], and also the following depiction, is based on the latter.

Before the linguistic modeling with dictionaries will be addressed in the next sections, an important issue should be pointed out also in this context: The true statistics $P(\mathfrak{R}^\tau)$ and $p(\boldsymbol{t}|\mathfrak{R}^\tau)$ are mostly not known in practice, and have to be approximated by estimates with help of data samples. In addition to the already addressed difficulty of estimating the appearance likelihood $p(\boldsymbol{t}|\mathfrak{R}^\tau)$, this additionally refers to the estimation of the word priors $P(\mathfrak{R}^\tau)$. In situations where the sampling is not sufficient, a correct word could be assigned a prior of $\hat{P}(\mathfrak{R}^\tau) = 0$. Clearly, this word will never be considered for the decision of the classifier. In literature, this problem is associated with the term "out-of-dictionary".

4.5.2 Linguistic modeling with a dictionary trie

The rationale of the linguistic modeling with a dictionary is to give a list of super references with $P(\mathfrak{R}^\tau) \neq 0$ — the actual entries in the dictionary — and implicitly assume $P(\mathfrak{R}^\tau) = 0$ for the remaining ones. Obviously, this modeling technique reduces the computational complexity of Equation (4.46) to an order that is linear with the dictionary size, assuming a naive, so-called flat dictionary representation.

Although this is more efficient than an exhaustive search, it can still be tremendously high for capacious dictionaries (e.g., with a size of 20 000 entries). In this respect, further modeling techniques are necessary in order to achieve real time processing.

One technique which permits a more efficient dictionary modeling exists by means of the the so-called *trie*[2] [Manke et al., 1996, Dengel et al., 1997] (cf. Figure 4.5). This approach has also been chosen for the *frog on hand* implementation.

Principally, a trie organizes the dictionary entries with help of a tree. In this formulation, an edge is labeled with a character class. Each tree node is associated with a character label sequence, in particular the one that is formed by concatenating the edge character labels leading from the root to the node. For the dictionary word representation, the tree contains a set of distinguished nodes which correspond to the dictionary words. In the example of Figure 4.5, those special nodes are marked with a bold circle.

[2]The spelling *trie* originates from the term "re*trie*val" [Dengel et al., 1997].

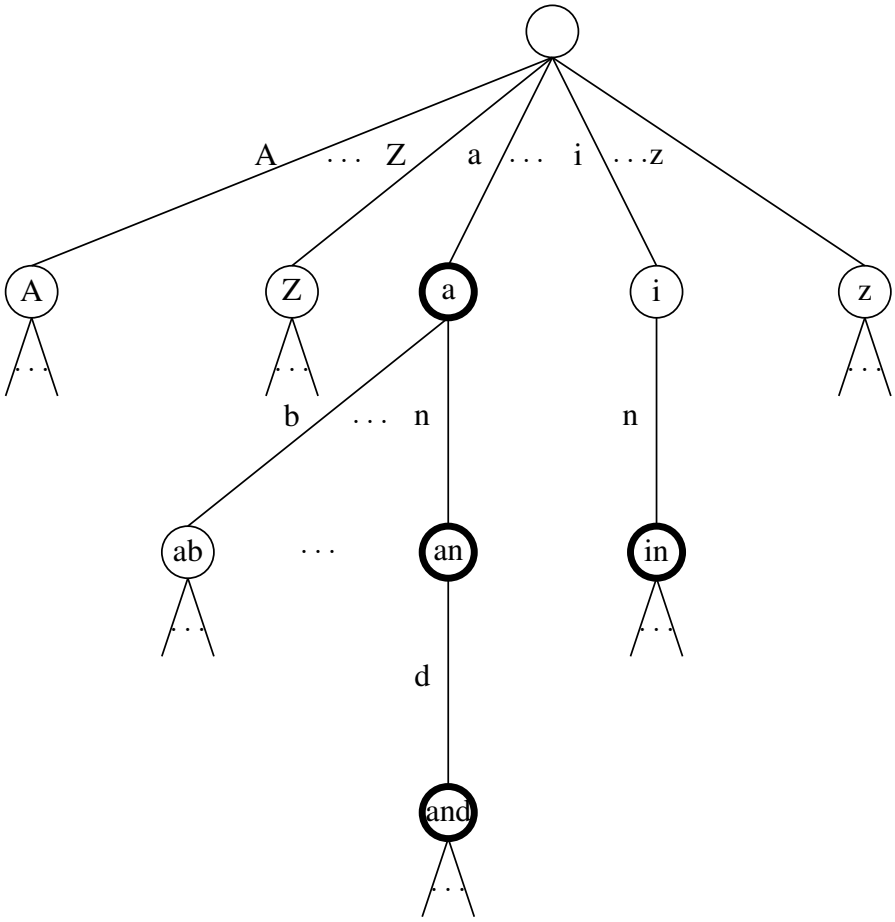


FIGURE 4.5: A *dictionary trie* organizes entries of a word dictionary in a tree structure. Each edge is associated with a character label, each node with a word prefix, corresponding to the edge labels originating from the root node. Dictionary entries are indicated by distinguished nodes, in this figure by a bold circle. (Figure reproduced from [Simon, 2003].)

4.5.3 Word classification with a dictionary trie, DP, and beam search

In the handwriting recognition community and contiguous research areas it is well understood how the above described techniques of dictionary tries, dynamic programming and beam search (cf. Section 2.3.3) can be combined to implement HMM based word recognition. In his work, Simon [2003] transfers these techniques to *frog on hand* and the CSDTW modeling.

In its simplified variant, the idea can be summarized by the following aspects:

1. Trie edges are associated with the corresponding CSDTW allograph models. Dynamic programming techniques, based on those explained in Section 4.3, are used to compute warping distances $D\left[\hat{d}\right]\left(t^s, |\mathcal{R}^{\tau(s)}\right)$ of those allograph models $\mathcal{R}^{\tau(s)}$ and sub-sequences t^s of the test pattern. Similar to the deliberations from Propo-

sition 4.1, the warping distance can be set into correspondence with the likelihood $p\left(t^s, (\mathbf{\Phi}^s)^* \,|\mathcal{R}^{\tau(s)}\right)$, where $(\mathbf{\Phi}^s)^*$ denotes the respective Viterbi alignment. In order to allow for different allograph hypotheses, allograph models for one edge character class are evaluated in parallel.

2. The topology of the individual CSDTW allograph models is extended in a way that transitions from the last state to the first state of subsequent CSDTW allograph models in the trie are permitted. The computed character likelihood $p\left(t^s, (\mathbf{\Phi}^s)^* \,|\mathcal{R}^{\tau(s)}\right)$ can be passed across the trie node in order to form the entire word likelihood $p\left(t, \mathbf{\Phi}^* | \mathfrak{R}^\tau\right)$. Figure 4.6 illustrates this idea for the example of a transition between the CSDTW models of "i" and "n".

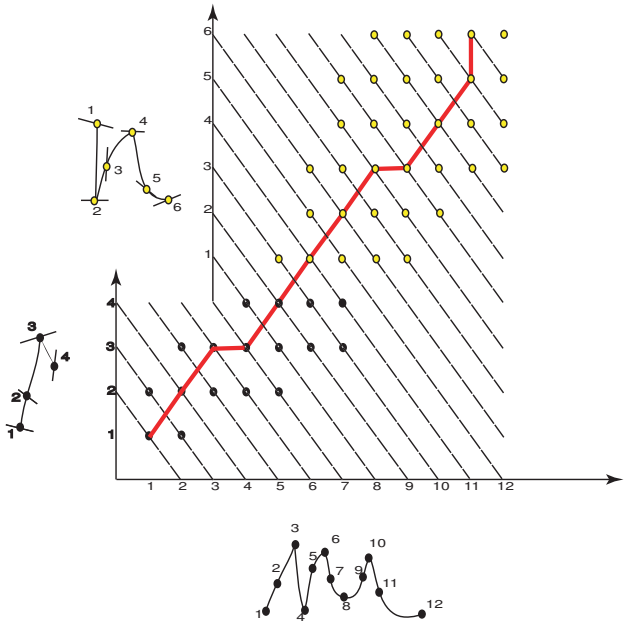


FIGURE 4.6: Word recognition with a dictionary trie and dynamic programming: The abscissa corresponds to the feature vectors of a simple test sequence, the word "in". The ordinate shows a sequence of the two CSDTW models that form the most probable super-reference $\mathfrak{R}^\tau$. Compared to the character recognition, additional transitions are included at the boundaries of the CSDTW models, in accordance to the nodes in the lexicon trie. Regard that the obtained Viterbi path implicitly gives a segmentation of t into its character parts t^1 and t^2. In this example, the two characters "i" and "n" are segmented between sample points 5 and 6. (Figure reproduced from [Simon, 2003].)

3. Despite the efficient tree structure, the evaluation of the whole trie is still too complex for a real time evaluation of reasonable dictionary sizes. Hence, beam search criteria similar to those explained in Section 2.3.3 are included in order to prune the evaluation of unlikely word hypotheses in an early stage of the search.

An interesting characteristic of the above sketched word recognition is that an optimal segmentation of t into $t^1, \ldots, t^S$ is obtained implicitly as a by-product from the recognition. In particular, the segmentation can be read from the Viterbi path transitions between two CSDTW character models (cf. Figure 4.6).

For further details about the recognition of words, its implementation and experimental results, it is referred to the above mentioned literature.

4.6 Experiments

In order to evaluate the performance of *frog on hand* with CSDTW classification, and to compare with other recognition systems, a number of experiments have been carried out.

4.6.1 Data

The experiments are based on sections 1a, 1b and 1c (digits, upper and lower case characters, respectively) of the *UNIPEN* [Guyon et al., 1994] "Train-R01/V07" database. For these sections the data set size is $\approx$ 16.000, 28.000 and 61.000 characters, respectively. The characters were randomly and disjointly divided into training and test sets of a ratio $2:1$. The division was completely random, thus one writer was allowed to be present in both of the sets.

Such a testing environment is also called *multi-writer* evaluation in literature. Contrary, an *omni-writer* evaluation takes care that one writer only appears in one of the sets but not in both, which is apparently a more difficult testing environment.

It should be noted that *UNIPEN* consists of very difficult data due to the variety of writers and noisy or mislabeled data. The present experiments use the database without cleaning in order to be as comparable as possible to other classification reports.

4.6.2 Results

For an effective evaluation, a few model parameters had to be set. A value for an initial, global covariance $\mathbf{\Sigma}$ (cf. Section 2.3.1) could be estimated on the basis of an ML estimation of previous models to $\mathbf{\Sigma} = \mathrm{diag}\,(0.08, 0.05, 0.15)$.

Beside the error rate, a significant property of a CSDTW classification model is its size, that is, the total number A^{tot} of generated allograph models. By varying $D_{\max}$ in the range $[2.0, \ldots, 7.0]$ (compare Figure 4.4) and $O_{\min}$ in $[6, \ldots, 23]$, differently sized CSDTW classification models were generated. Table 4.1 summarizes mean classification error rates $\tilde{E}$ and mean model sizes $\tilde{A}^{\mathrm{tot}}$ for selected configurations, each of which is the average from five different experiments with varying dataset partitionings.

From the table it can be taken that an error minimum is given in all sections for $D_{\max} \approx 3.5$–4.0 and $O_{\min} = 6$, leading to model sizes of $\tilde{A}^{\mathrm{tot}} = 150,\ 268$ and 608 for sections 1a/b/c, respectively. Notably, a further decrease of $D_{\max}$ (and thus an enlarging of the model size) degrades the accuracy. On the other hand, the model size can significantly be reduced (by a

UNIPEN section	Approach	Error rate $\tilde{E}$	$\tilde{A}^{\text{tot}}$	$D_{\max}$	$O_{\min}$	*UNIPEN* dataset comment
1a (digits)	**CSDTW**	3.9 % **2.9 %** 3.3 % 4.3 %	309 **150** 65 27	2.0 **3.5** 6.0 7.0	6 **6** 6 23	Train-R01/V07 67 % train / 33 % test set
	DAG-SVM-GDTW (cf. Section 7)	3.8 %				Train-R01/V07 40 % train / 40 % test set
	MLP [Parizeau et al., 2001]	3.0 %				DevTest-R02/V02
	HMM [Hu et al., 2000]	3.2 %				Train-R01/V06 4 % bad data removed
1b (upper case)	**CSDTW**	9.2 % **7.2 %** 7.8 % 9.5 %	522 **268** 161 67	2.0 **4.0** 6.0 7.0	6 **6** 6 23	Train-R01/V07 67 % train / 33 % test set
	DAG-SVM-GDTW (cf. Section 7)	7.6 %				Train-R01/V07 40 % train / 40 % test set
	HMM [Hu et al., 2000]	6.4 %				Train-R01/V06 4 % bad data removed
1c (lower case)	**CSDTW**	9.9 % **9.3 %** 10.3 % 11.1 %	1050 **608** 283 117	2.0 **3.5** 6.0 7.0	6 **6** 6 23	Train-R01/V07 67 % train / 33 % test set
	DAG-SVM-GDTW (cf. Section 7)	12.1 %				Train-R01/V07 20 % train / 20 %test set
	MLP [Parizeau et al., 2001]	14.4 %				DevTest-R02/V02
	HMM-NN hybrid [Gauthier et al., 2001]	13,2 %				Train-R01/V07
	HMM [Hu et al., 2000]	14,1 %				Train-R01/V06 4 % bad data removed

TABLE 4.1: Experiments on the *UNIPEN* sections 1a/b/c (indicated in the first column). The second column denotes the classification approach used. Here, also other HWR approaches have been collected from literature. The third column shows the mean error rate $\tilde{E}$ of five different dataset partitionings. For CSDTW it was computed for differently sized models, as indicated in the forth column by the average total number of allographs $\tilde{A}^{\text{tot}}$. The different model sizes were generated by varying $D_{\max}$ in the range $[2.0, \ldots, 7.0]$ and $O_{\min}$ in $[6, \ldots, 23]$ (see column five and six). The CSDTW result with the lowest error rate is typed bold face. As some experiments were computed on different *UNIPEN* distributions, details are given in the last column.

factor ≈ 5) with $D_{\max} = 7.0$ and $O_{\min} = 23$. For this parameter configuration, the recognition accuracy decreases by absolute ≈ 2 %, however remains in an acceptable range.

For a comparison of the recognition accuracy in a global context other *UNIPEN* results were collected from literature and included in Table 4.1. It must be stated that though all experiments were computed on *UNIPEN* data, various reports used different character sets. Benchmarks were computed on miscellaneous versions and sizes of a *UNIPEN* database or some authors removed low quality/mislabeled characters, as indicated in the table's last column. Further, differences with respect to the multi- and omni-writer testing environments aggravate a comparison, as argued above. In this respect caution must be given, when interpreting the rates. However, beyond this caution, a comparison of all results indicates that *frog on hand* with CSDTW classification achieves equivalent or superior rates with respect to all other approaches in the three categories. Especially for the important lower case character section, *frog on hand* with CSDTW performs significantly better.

4.6.3 Visual study of CSDTW models

Contrary to the speech recognition domain and due to the rather geometrically selected features, a feature representation of the second order statistic (the Gaussian parameters $\boldsymbol{\mu}_j^{lk}$ and $\boldsymbol{\Sigma}_j^{lk}$) of a handwritten character pattern can nicely and meaningfully be illustrated in the present HWR system. One example has already been depicted in Figure 4.1. Figure 4.7 shows — with the same representation — a selection of CSDTW references $\mathcal{R}^{lk}$.

A few conclusions can be derived from the figure:

1. The combined cluster analysis and parameter estimation of the CSDTW training is well suited to generate typical and pairwise dissimilar allograph reference models $\mathcal{R}^{lk}$. In the case of the character "x" note the different stroke order and direction for the models $\mathcal{R}^{(\text{"x"})1}$ and $\mathcal{R}^{(\text{"x"})2}$.

2. Larger variations, indicated by the length of the lines attached to the sample points, are often found in the beginning and the end of the characters, smaller variations in the center part.

3. For reference models $\mathcal{R}^{lk}$ with large $N_{\mathcal{R}^{lk}}$ the transition $(0, 1)$ has high probability, for reference models $\mathcal{R}^{lk}$ with small $N_{\mathcal{R}^{lk}}$ the transition $(1, 0)$. This can be explained by the fact that a test pattern t has to be aligned against all sample points in $\mathcal{R}^{lk}$. In the case of a "long" $\mathcal{R}^{lk}$ and an average sized t this is only possible if the transition $(0, 1)$ is taken frequently.

4.7 Summary

This chapter has described CSDTW as a generative learning framework, which combines cluster analysis and statistical sequence modeling. On the one hand, it employs a specifically

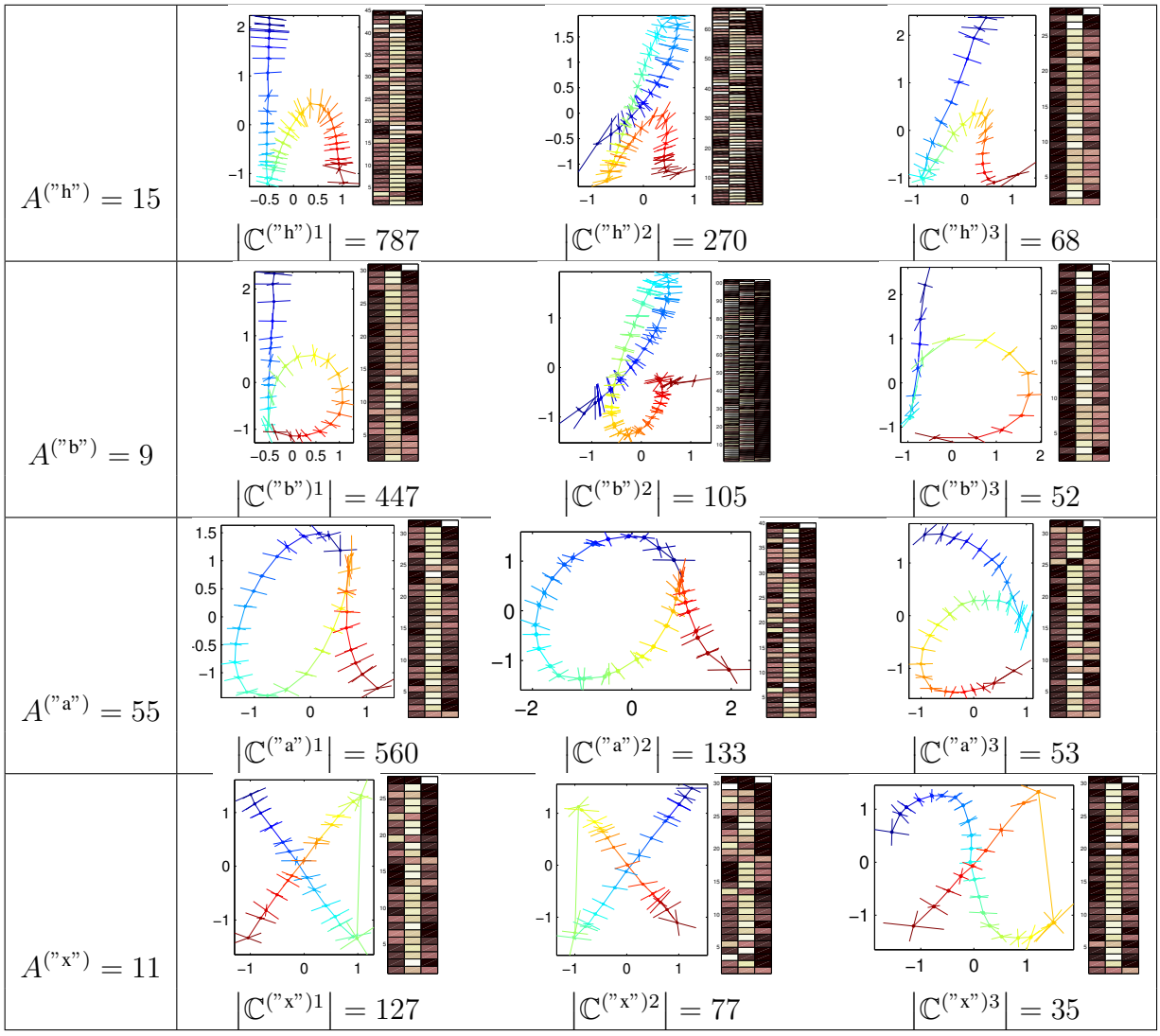


FIGURE 4.7: A selection of allograph reference models for the character classes “h”, “b”, “a” and “x”. For a description of the graphics it is referred to the caption of Figure 4.1. Basis for these reference models is a *UNIPEN* lowercase character experiment with the parameter configuration $D_{\max} = 3.5$ and $O_{\min} = 6$ (cf. Table 4.1). The number of allographs A^l for the character class l is indicated in the first column. The second, third and fourth column represents three selected reference models, the cardinality $|\mathbb{C}^{lk}|$ of the respective cluster is given below.

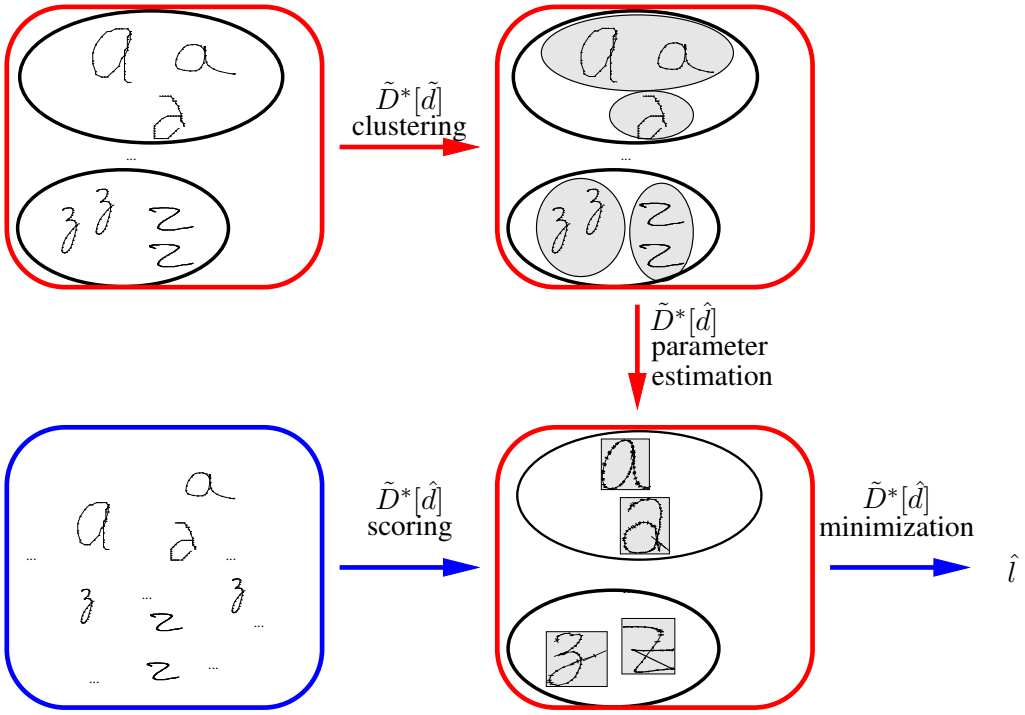


FIGURE 4.8: A summary of the CSDTW training and classification framework. Training comprises clustering and parameter estimation, classification comprises scoring and minimizing. The clustering uses $\tilde{D}^*[\tilde{d}]$ as the underlying distance function, parameter estimation and scoring use $\tilde{D}^*[\hat{d}]$. Notably, $\tilde{D}^*[\tilde{d}]$ and $\tilde{D}^*[\hat{d}]$ are closely related. In this respect, CSDTW is a holistic combination of clustering and statistical sequence modeling.

modified DTW Viterbi distance $\tilde{D}^*[\tilde{d}]\,(\boldsymbol{t}, \boldsymbol{r})$ in the cluster analysis. On the other hand, the SDTW Viterbi distance $\tilde{D}^*[\hat{d}]\,(\boldsymbol{t}, \mathcal{R})$ is used in the statistical parameter estimation and classification. Notably, $\tilde{D}^*[\tilde{d}]\,(\boldsymbol{t}, \boldsymbol{r})$ is a special incarnation of $\tilde{D}^*[\hat{d}]\,(\boldsymbol{t}, \mathcal{R})$. This aspect has two attractive effects. First, clustering and statistical modeling appear as a holistic combination, where cluster and classification space coincide. Clusters in the cluster space correspond to well formed clusters in the classification space. Second, the result of the clustering can be consulted as a natural solution to the problem of dimensioning and initializing the CSDTW reference models in the context of the iterative parameter estimation. In this respect, clustering and statistical parameter estimation are seamlessly integrated into each other. Figure 4.8 summarizes the training and classification issues graphically.

Another beneficial property of CSDTW is the following. With the clustering parameters $D_{\max}$ and $O_{\min}$ the classifier designer has direct influence on the granularity of the clustering. He or she can scale the classifier with respect to a wide range of cluster numbers and sizes, finding a compromise between the recognition accuracy and the computational time and memory requirements. A concrete example is the following: a large $D_{\max}$ implies that the role of the allograph models correspond to broader (maybe culturally conditioned) variations, while using a smaller $D_{\max}$ further reflects variations that may be due to different writer habits or the

word context in that a character is written. Figure 4.4 gives an illustration for this idea. The scalability has been shown to be very practicably in particular for the transfer of *frog on hand* with CSDTW on a Compaq iPAQ PDA (cf. Chapter 8). In this application, limited memory and computational resources play a significant role.

CHAPTER 5

Statistics of semi-directional data

In Chapter 3 the tangent slope angle has been defined and integrated as one valuable element into the feature sub-space representation of online handwriting. Following, Chapter 4 has introduced the CSDTW modeling, which is — among others — based on a statistical modeling of the feature sub-space. In this context, there is still one open question to be answered. Namely, the tangent slope angle differs from the other features with respect to one particular aspect: it originates from directional *instead of the common* linear *data. As will be shown later, directional data require specific techniques for a statistical modeling. An additional difficulty is the circumstance that the tangent slope angle appears in combination with linear data, $\tilde{x}$ and $\tilde{y}$. This chapter describes, how linear* and *directional variables can be combined in a* single*, multivariate PDF: the* multivariate semi-wrapped Gaussian *PDF. In order to reduce analytic complexity, considerations are confined to the constraint that variances of the circular variables are small. The use of the modeling in the CSDTW classification shows that the described solution gives significant improvements in recognition accuracy, computational speed and memory requirements, compared to commonly employed modeling approaches.*

5.1 Introduction

In statistical pattern recognition the modeling of an abstract *feature space* with parametric PDFs is very common. Often — also in Chapter 4 — the Gaussian (or normal) PDF

$$p(\mathbf{x}) = \mathcal{N}_{\boldsymbol{\mu}_\mathbf{x},\boldsymbol{\Sigma}_\mathbf{x}}(\mathbf{x}) = \left(|2\pi\boldsymbol{\Sigma}_\mathbf{x}| \exp\left((\mathbf{x}-\boldsymbol{\mu}_\mathbf{x})^T \boldsymbol{\Sigma}_\mathbf{x}^{-1}(\mathbf{x}-\boldsymbol{\mu}_\mathbf{x})\right)\right)^{-1/2} \tag{5.1}$$

or a mixture of it is used to describe the probability density of a random vector $\mathbf{x} \in \mathbb{R}^F$. For *linear* data, i.e., data that are distributed on the real line $\mathbb{R}$, sensible definitions of the Gaussian

parameters *mean* and *covariance* exist, namely

$$\boldsymbol{\mu}_{\mathbf{x}} = \mathcal{E}\left[\mathbf{x}\right] \tag{5.2}$$

$$\boldsymbol{\Sigma}_{\mathbf{x}} = \mathcal{E}\left[\left(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}}\right)\left(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}}\right)^{T}\right], \tag{5.3}$$

with $\mathcal{E}\left[\cdot\right]$ the expectation value of its argument.

However, not all data in real world applications are of this particular linear type. In some situations [Mardia, 1972, Shatkay and Kaelbling, 1998, Döker et al., 1999, Hyberts et al., 1992, Lovell et al., 1991] data originate from *directions*. Directions, in general, may be visualized as points on the surface of a hypersphere, in two dimensions on the circumference of a circle. In the latter situation we will talk about circular data. A directional variable is inherently cyclic and the common statistical modeling used for linear data — including Equations (5.1)–(5.3) — is not appropriate here, as will be shown later.

5.2 Literature review

Also in (online) HWR we are faced with the problem of directional data, since a valuable feature of this particular circular type exists, namely the *tangent slope angle* θ (cf. Section 3.5). The answer of many HWR systems [Guyon et al., 1991, Schenkel et al., 1995, Jäger et al., 2000] to the problem of modeling circular features with parametric PDFs is somewhat defensive. Basically, they avoid a direct modeling by a transformation of θ into the representation $(\cos\theta, \sin\theta)$. Contrary to θ, the elements $\cos\theta$ and $\sin\theta$ themselves are not circular. In this respect, previous systems take them for a linear feature and model $(\cos\theta, \sin\theta)$ instead of θ as part of a linear feature vector by Equations (5.1)–(5.3).

However, there are some drawbacks in this strategy. *First*, the dimension of the resulting feature space is unnecessarily increased (by one), since the strategy uses two dimensions to describe one degree of freedom. Among others, computational speed and memory requirements are negatively affected. *Second*, the $(\cos\theta, \sin\theta)$ representation includes high dependencies: the feature pair $(\cos\theta, \sin\theta)$ lies on the circumference of the unit circle. In the situation when parametric basis functions are used to model probability densities, those dependencies have to be addressed. This, however, is very difficult to achieve since basis functions generally assume a restrictive shape.

On the other hand, statisticians have developed methodologies that deal with directional data, in last decades especially influenced by the work of Mardia [1972]. Remarkably, it seems that this work has not found its way into the pattern and handwriting recognition community.

This unfortunate situation might be explained by the following reason. While the directional methodologies developed so far are well suited for distributions of solely directional variables (as they appear in physical, biological, geological, geographical or medical applications) they still lack a clear description how they can be advantageously applied for multivariate PDFs of both linear *and* directional variables. Contrary to the applications listed above, where mostly a two- or three-dimensional space of spatial or physical coordinates is to be modeled, in pattern

recognition we are faced with the problem of modeling an abstract feature space. Mostly, in these cases the situation of mixed linear and directional data exist.

This chapter proposes an approach to the problem of integrating directional *and* linear data into a multivariate PDF that aims to model an abstract feature space: the *multivariate semi-wrapped Gaussian* PDF [Bahlmann, 2005]. In order to reduce analytic complexity, considerations are confined to a special constraint: it is assumed that the circular feature has a small variance.

The argumentation of this chapter is as follows: Sections 5.3 and 5.4 first motivate and review basic concepts from the statistics of directional data. They further introduce a distribution for directional data (the *wrapped Gaussian distribution*) and propose an approximation of it. Following, Section 5.5 introduces the formulation of a *multivariate semi-wrapped Gaussian distribution* and transfers the deliberations made about approximation issues from Section 5.4 to this construct. Finally, Section 5.6 gives experimental results.

5.3 Statistics of directional data

Directional data may be visualized as points on the surface of a hypersphere, in two dimensions on the circumference of a circle. Figure 5.1 illustrates an example. One problem that arises with directional data in combination with the use of "conventional", i.e., linear statistics shall be illustrated in the following.

5.3.1 Linear statistics and directional data

Consider a linear variable x and a transformation $t_s(x) = \tilde{x} = x - \nu$. The transformation t_s represents a shift of the coordinate system origin. Valuable properties of the statistics with linear data, in particular of *mean* and *variance*, can be expressed by the equations

$$\mu_{\tilde{x}} = \mu_x - \nu \tag{5.4}$$
$$\sigma^2_{\tilde{x}} = \sigma^2_x. \tag{5.5}$$

Equation (5.4) implies that the relative position of the mean remains invariant under a shift of the coordinate system origin. Equation (5.5) refers to the invariance of the variance. In other words, the validity of Equations (5.4) and (5.5) guarantees a statistical behavior which is essentially independent from the chosen coordinate system.

Now consider a circular variable ϑ. For ϑ an addition "$a + b$" becomes "$(a + b) \bmod 2\pi$". Here and in the remainder, we assume a period of 2π and adopt the convention of angles represented in the interval $(-\pi, \pi]$. Note that under this assumption the $\bmod$ operator also maps to $(-\pi, \pi]$. Let the variables $\mu^{\mathrm{c}}_\vartheta$ and V^{c}_ϑ denote the circular counterparts of mean and variance. Reasonable definitions for $\mu^{\mathrm{c}}_\vartheta$ and V^{c}_ϑ should have a similar behavior as Equation (5.4) and (5.5) under a shift of the zero direction which is expressed by $\tilde{\vartheta} = (\vartheta - \nu) \bmod 2\pi$. In

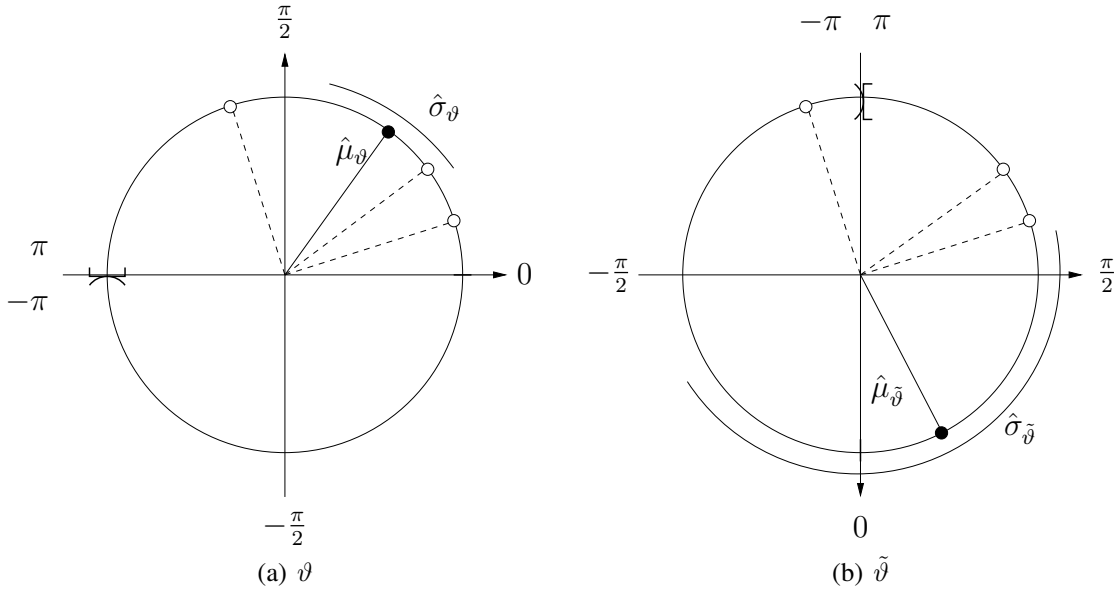


FIGURE 5.1: Data points of a circular variable are shown as white dots on the unit circle. This figure illustrates that linear definitions of mean and variance violate the invariance of location and variance for directional data under a shift of zero direction $\tilde{\vartheta} = (\vartheta - \nu) \bmod 2\pi$: (a) For the example observations $\Theta = \{0.1\pi, 0.2\pi, 0.6\pi\}$, unbiased ML estimates for mean and standard deviation can be calculated to $\hat{\mu}_\vartheta = 0.3\pi$ (corresponding to the black dot) and $\hat{\sigma}_\vartheta \approx 0.26\pi$ (corresponding to the length of the arc). (b) The observations are shifted to $\tilde{\Theta} = \{0.6\pi, 0.7\pi, -0.9\pi\}$, corresponding to $\nu = -0.5\pi$. In the figure this corresponds to a rotation of the coordinate axes about 0.5π clockwise. Estimates for mean and standard deviation of the data points in the new coordinates can be calculated to $\hat{\mu}_{\tilde{\vartheta}} \approx 0.13\pi$ and $\hat{\sigma}_{\tilde{\vartheta}} \approx 0.90\pi$. Obviously, $\hat{\mu}_{\tilde{\vartheta}} \neq (\hat{\mu}_\vartheta - \nu) \bmod 2\pi$ and $\hat{\sigma}^2_{\tilde{\vartheta}} \neq \hat{\sigma}^2_\vartheta$, thus neither the location nor the variance are invariant with respect to a shift of the origin.

this respect, equivalent invariances for a circular variable are

$$\mu^{\mathrm{c}}_{\tilde{\vartheta}} = (\mu^{\mathrm{c}}_\vartheta - \nu) \bmod 2\pi \tag{5.6}$$
$$V^{\mathrm{c}}_{\tilde{\vartheta}} = V^{\mathrm{c}}_\vartheta. \tag{5.7}$$

However, it can easily be verified that with the linear definitions of mean and variance, given in Equations (5.2) and (5.3), the desired invariance is violated, i.e.,

$$\mu_{\tilde{\vartheta}} \neq (\mu_\vartheta - \nu) \bmod 2\pi \tag{5.8}$$
$$\sigma^2_{\tilde{\vartheta}} \neq \sigma^2_\vartheta \tag{5.9}$$

in general. Figure 5.1 gives an example for this misbehavior, employing a simple set of circular observations $\Theta = \{0.1\pi, 0.2\pi, 0.6\pi\}$ and $\tilde{\Theta} = \{0.6\pi, 0.7\pi, -0.9\pi\}$, which corresponds to $\nu = -0.5\pi$. For these observations, unbiased maximum likelihood (ML) estimates for mean and variance can be computed to $\hat{\mu}_\vartheta = 0.3\pi$, $\hat{\sigma}_\vartheta \approx 0.26\pi$, $\hat{\mu}_{\tilde{\vartheta}} \approx 0.13\pi$ and $\hat{\sigma}_{\tilde{\vartheta}} \approx 0.90\pi$, which are obviously not in agreement with Equations (5.6) and (5.7).

As a concluding remark, we note that for circular data the linear definitions of mean and variance are highly dependent on the zero direction, which is an inappropriate behavior and demands for a suitable handling.

Physicists and statisticians have developed a methodology for dealing with statistics of directional data. Original publications lead back to the early 20th century [von Mises, 1918, Zernike, 1928, Wintner, 1933, Gumbel et al., 1953]. The nowadays most comprehensive description can be found in the book of Mardia [1972]. We briefly summarize the basics in the following.

5.3.2 Circular Mean direction and circular variance

Assume a circular random variable ϑ with a PDF $p(\vartheta)$. The PDF should satisfy $p(\vartheta) \geq 0$ and $\int_{-\pi}^{\pi} p(\vartheta)\, d\vartheta = 1$. Mardia [1972] represents ϑ as a complex number $e^{J\vartheta}$ (with $J^2 = -1$) and employs the notation of *circular mean direction* μ_ϑ^c and *circular variance* V_ϑ^c. They are defined by

$$\rho_\vartheta e^{J\mu_\vartheta^c} = \mathcal{E}\left[e^{J\vartheta}\right] \tag{5.10}$$

and

$$V_\vartheta^c = 1 - \rho_\vartheta. \tag{5.11}$$

The quantity ρ_ϑ is called the *resultant length*. Figuratively speaking, μ_ϑ^c is the expected phase and ρ_ϑ the expected length of $e^{J\vartheta}$. $V_\vartheta^c \in [0, 1]$ measures the amount of *dispersion* (cf. Figure 5.2).

It can be shown [Mardia, 1972] that contrary to the linear definitions of mean and variance, μ_ϑ^c and V_ϑ^c fulfill the demanded invariance of Equations (5.6) and (5.7) and can be utilized as suitable counterparts for the linear mean and variance.

5.4 Wrapped Gaussian distribution

Based on μ_ϑ^c and V_ϑ^c, Mardia describes two *circular normal distributions* that should serve as appropriate substitutes for the univariate linear normal distribution. One is the *wrapped Gaussian (or normal) distribution*, the other the *von Mises distribution*. Both have particular benefits and drawbacks compared to each other. Among others, the wrapped Gaussian has theoretical advantages, the von Mises distribution practical benefits including the parameter estimation [Mardia, 1972, section 3.4.10]. However, it can be shown that they can be made to approximate each other closely. In this respect, a concrete assessment for one of the alternatives is practically not too restrictive.

As previously explained, the aim in the context of this thesis is to set up a *multivariate* Gaussian distribution of both linear *and* circular variables. In this context it appears that due to its apparent closeness to the linear Gaussian the wrapped Gaussian (a definition will follow shortly) is the more natural choice for the present problem. Thus, in spite of the practical drawbacks in the parameter estimation, the wrapped Gaussian distribution, which will briefly be reviewed in the remainder of this section, has been chosen.

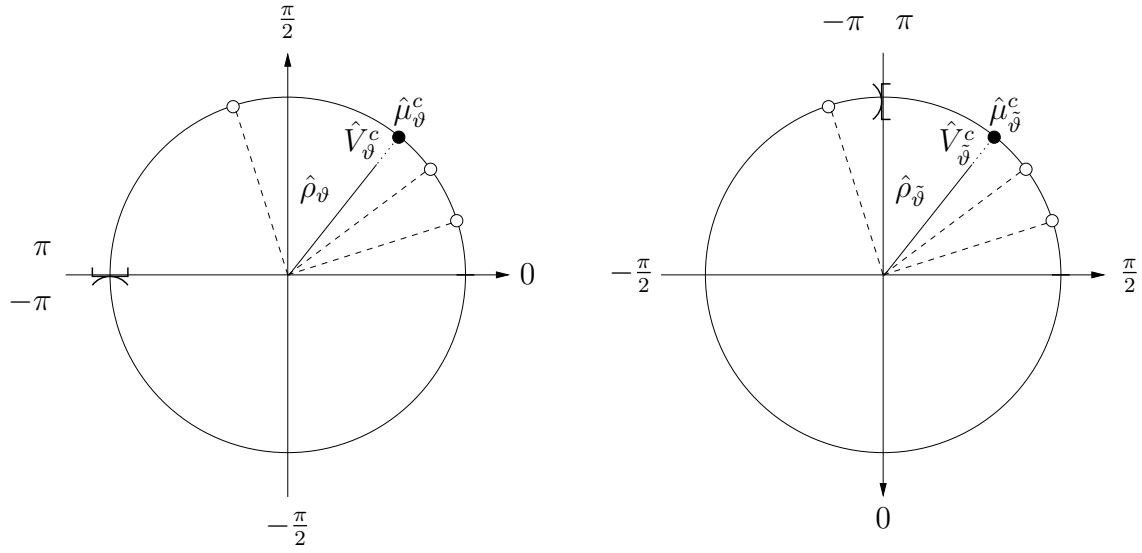


FIGURE 5.2: For the same observations Θ (part (a)) and $\tilde{\Theta}$ (part (b)) as in Figure 5.1, this figure shows estimates of the circular mean direction $\hat{\mu}^{\mathrm{c}}_\vartheta$, the resultant length $\hat{\rho}_\vartheta$ and the circular variance $\hat{V}^{\mathrm{c}}_\vartheta$. $\hat{\mu}^{\mathrm{c}}_\vartheta$ corresponds to the phase of the complex number associated to the black dot, $\hat{\rho}_\vartheta$ to the solid line and $\hat{V}^{\mathrm{c}}_\vartheta = 1 - \hat{\rho}_\vartheta$ to the dotted line towards $\hat{\mu}^{\mathrm{c}}_\vartheta$. The figures also express the validity of Equations (5.6) and (5.7).

5.4.1 General wrapped distribution

Any given PDF $p(x)$ of a linear variable x on the line can be "wrapped" around the circumference of a circle of unit radius. That is, the PDF $p_{\mathrm{w}}(\vartheta)$ of the wrapped variable

$$\vartheta = x_{\mathrm{w}} = x \bmod 2\pi \quad \in (-\pi, \pi] \tag{5.12}$$

is

$$p_{\mathrm{w}}(\vartheta) = \sum_{k=-\infty}^{\infty} p(\vartheta + 2\pi k). \tag{5.13}$$

5.4.2 Wrapped Gaussian distribution

In particular, for $p(x)$ being a univariate Gaussian distribution $\mathcal{N}_{\mu_x,\sigma_x}(x)$ the *wrapped univariate Gaussian distribution* is defined as

$$\mathcal{N}^{\mathrm{w}}_{\mu^c_\vartheta, V^c_\vartheta}(\vartheta) = \sum_{k=-\infty}^{\infty} \left(2\pi (\sigma_x)^2 \exp\left(\frac{(\vartheta - \mu_x + 2\pi k)^2}{(\sigma_x)^2} \right) \right)^{-1/2}. \tag{5.14}$$

It can be shown [Mardia, 1972] that for the circular mean direction μ_ϑ^c and the circular variance V_ϑ^c the equations

$$\mu_\vartheta^c = \mu_x \bmod 2\pi \tag{5.15}$$

$$(\sigma_x)^2 = -2\log(1 - V_\vartheta^c) \tag{5.16}$$

hold. Further, $\mathcal{N}^{\mathrm{w}}_{\mu_\vartheta^c, V_\vartheta^c}(\vartheta)$ is unimodal (i.e., has a single local maximum) and symmetric about μ_ϑ^c. With the relations of Equations (5.15) and (5.16) in mind we can use the notations of $\mathcal{N}^{\mathrm{w}}_{\mu_\vartheta^c, V_\vartheta^c}$ and $\mathcal{N}^{\mathrm{w}}_{\mu_x, \sigma_x}$ interchangeably in the remainder of this chapter.

Figure 5.3 (a) shows an example of the wrapped Gaussian distribution with parameters $\mu_x = \pi/2$ and $\sigma_x = 1.50$. The dashed lines show three contributing Gaussian terms, corresponding to $k = -1$, $k = 0$ and $k = 1$ of Equation (5.14).

5.4.3 An approximation to the wrapped Gaussian distribution

The wrapped Gaussian and von Mises distributions have been successfully applied to a variety of problems [Mardia, 1972]. Those problems can be assigned into one of the following categories:

1. The PDF is one-dimensional and the random variable corresponds to a direction in two dimensions.

2. The PDF is two-dimensional and the random variables correspond to a direction in three dimensions.

However, contrary to these problems, in many pattern recognition applications we are faced with the situation that a multivariate (> 2) PDF is to be modeled where only one (or a few) dimension(s) correspond to circular and the rest to linear variables. A suitable transfer of the mentioned directional distributions (the wrapped Gaussian or the von Mises distribution) to these "semi-directional" situations is not straightforward. In order to cope with this difficulty the remaining deliberations will be confined to an approximation of the wrapped Gaussian distribution: it is assumed that the wrapped Gaussian can be approximated by only one, but the most meaningful wrap of it. It will turn out in Section 5.5 that with this confinement a semi-directional PDF can be modeled directly.

In this respect, the following deliberations shall be restricted to situations in that mainly one wrap of Equation (5.14) contributes to the computation of values $\mathcal{N}^{\mathrm{w}}_{\mu_\vartheta^c, V_\vartheta^c}(\vartheta)$. As can be verified from Figure 5.3 (a), this is the case, if the overlap of neighboring Gaussian wraps is negligible. In this case, it is permissible to approximate $\mathcal{N}^{\mathrm{w}}_{\mu_\vartheta^c, V_\vartheta^c}(\vartheta)$ by the mostly contributing wrap for $\vartheta \in (-\pi, \pi]$. This approximation can be summarized by the formulation

$$\mathcal{N}^{\mathrm{aw}}_{\mu_\vartheta^c, V_\vartheta^c}(\vartheta) = \left(2\pi(\sigma_x)^2 \exp\left(\frac{((\vartheta - \mu_x) \bmod 2\pi)^2}{(\sigma_x)^2}\right)\right)^{-1/2}. \tag{5.17}$$

where the upper index "aw" shall indicate the term "approximated wrapped". Again, the notations $\mathcal{N}^{\text{aw}}_{\mu_\vartheta^c, V_\vartheta^c}$ and $\mathcal{N}^{\text{aw}}_{\mu_x,\sigma_x}$ shall be used interchangeably, with Equations (5.15) and (5.16) giving the transformations between the respective parameters.

Figures 5.3 (b)–(d) compare $\mathcal{N}^{\text{w}}_{\mu_x,\sigma_x}(\vartheta)$ and $\mathcal{N}^{\text{aw}}_{\mu_x,\sigma_x}(\vartheta)$ for the particular parameter settings $\mu_x = \pi/2$ and $\sigma_x \in \{1.50, 1.25, 1.00\}$. For $\sigma_x = 1.50$ and $\sigma_x = 1.25$ the reader can verify approximation errors, centered at the anti-mode of $\mathcal{N}^{\text{w}}_{\mu_x,\sigma_x}(\vartheta)$. However, for $\sigma_x = 1.00$ only small deviations of the two functions can be found.

Thus, it will be assumed in the remainder of this chapter that for $\sigma_x \lesssim 1$ errors were small, if one uses $\mathcal{N}^{\text{aw}}_{\mu_x,\sigma_x}(\vartheta)$ instead of $\mathcal{N}^{\text{w}}_{\mu_x,\sigma_x}(\vartheta)$ to model the PDF of a circular variable. A quantitative statement with respect to the approximation error can be specified in terms of the integrated error

$$E_{\text{int}}(\sigma_x) = \int_{-\pi}^{\pi} \left| \mathcal{N}^{\text{aw}}_{\mu_x,\sigma_x}(\vartheta) - \mathcal{N}^{\text{w}}_{\mu_x,\sigma_x}(\vartheta) \right| d\vartheta, \tag{5.18}$$

which corresponds to the area between the solid and the upper dotted lines in Figure 5.3 (a). If only directly adjacent wraps have a considerable overlap (which is the case for small variances), the mentioned area is approximately equivalent to the area below the lower dotted lines. Further, as the intersection of two adjacent wraps is at the position $(\mu_x - \pi) \bmod 2\pi = (\mu_x + \pi) \bmod 2\pi$, $E_{\text{int}}(\sigma_x)$ corresponds to the area of $\mathcal{N}_{\mu_x,\sigma_x}(x)$ that falls outside the interval $(\mu_x - \pi, \mu_x + \pi]$. Due to the symmetry of $\mathcal{N}$, it can be computed to

$$E_{\text{int}}(\sigma_x = 1) = 2 \int_{-\infty}^{-\pi} \mathcal{N}_{0,1}(x)\, dx = \operatorname{erfc}\left(\frac{\pi}{\sqrt{2}}\right) \approx 0.0017 \tag{5.19}$$

with

$$\operatorname{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^{\infty} \exp\left(-t^2\right) dt \tag{5.20}$$

the complementary error function.

5.4.4 Parameter estimates

In statistical pattern recognition the estimation of the parameters μ_ϑ^{c} and V_ϑ^c for $\mathcal{N}^{\text{aw}}_{\mu_\vartheta^c, V_\vartheta^c}(\vartheta)$ from a set $\Theta = \left\{\vartheta^{(1)}, \ldots, \vartheta^{(M)}\right\}$ of circular observations is of practical importance.

A maximum likelihood estimate $\hat{\mu}_\vartheta^{\text{c}}$ can rather straightforwardly be computed. Mardia [1972] derives the formula

$$\hat{\mu}_\vartheta^{\text{c}} = \arg\left(\frac{1}{M}\sum_{m=1}^{M} e^{J\vartheta^{(m)}}\right). \tag{5.21}$$

An estimate for V_ϑ^{c} can be obtained similarly by

$$\hat{V}_\vartheta^{\text{c}} = 1 - \left\| \frac{1}{M}\sum_{m=1}^{M} e^{J\vartheta^{(m)}} \right\|. \tag{5.22}$$

However, when the assumption $\sigma_x \lesssim 1$ holds, another approximative solution is valid. Then, as argued above, $\mathcal{N}^{\mathrm{w}}_{\mu_x,\sigma_x}(\vartheta)$ becomes close to $\mathcal{N}^{\mathrm{aw}}_{\mu_x,\sigma_x}(\vartheta)$. Under this assumption, one can think of $\mathcal{N}^{\mathrm{aw}}_{\mu_x,\sigma_x}$ being a single Gaussian centered at $\hat{\mu}^{\mathrm{c}}_{\vartheta}$ (corresponding to only one wrap) and only a small accuracy is sacrificed with a linear-like estimate for V^c_ϑ via σ_x by means of

$$(\hat{\sigma}_x)^2 \;\approx\; \frac{1}{M-1}\sum_{m=1}^{M}\left(\left(\vartheta^{(m)} - \hat{\mu}^{\mathrm{c}}_{\vartheta}\right) \bmod 2\pi\right)^2. \tag{5.23}$$

Note that Equation (5.23) employs the circular mean direction estimate $\hat{\mu}^{\mathrm{c}}_{\vartheta}$ instead of the linear mean estimate $\hat{\mu}_\vartheta$. As it has been shown in Section 5.3.1, the latter is inappropriate for directional data.

In the present case, the advantage of Equation (5.23) over Equation (5.22) is that it can be straightforwardly extended when covariances of directional and linear data are to be computed (cf. Section 5.5.5).

5.5 Multivariate semi-wrapped Gaussian distribution

In the following, the formulation of a combination of wrapped and non-wrapped Gaussian distributions for multivariate situations will be introduced. The resulting distribution will be formulated as *multivariate semi-wrapped Gaussian distribution*. To start with, the *multivariate wrapped distribution* and the *multivariate semi-wrapped distribution* shall be defined first.

5.5.1 Multivariate wrapped distribution

The concept of a univariate wrapped distribution can be extended to the multivariate context by an extension of the simple sum in Equation (5.13) to a number of F sums that cover all dimensions in the feature space:

$$p_{\mathrm{w}}(\mathbf{x}) = \sum_{k_1=-\infty}^{\infty} \cdots \sum_{k_F=-\infty}^{\infty} p\left(\mathbf{x} + 2\pi k_1 \mathbf{e}_1 + \cdots + 2\pi k_F \mathbf{e}_F\right). \tag{5.24}$$

In this equation, $\mathbf{e}_k = (0,\ldots,0,1,0,\ldots,0)^T$ is the k-th Euclidean basis vector (with an entry of 1 at the k-th element and 0 elsewhere). Figure 5.4 illustrates an example of a bivariate wrapped Gaussian PDF. In correspondence to Figure 5.3 (a), the reader can see nine Gaussian summands (corresponding to $k_1 = -1, 0, 1$ and $k_2 = -1, 0, 1$) as well as their sum (the small patch, restricted to the interval $(-\pi, \pi] \times (-\pi, \pi]$).

5.5.2 Multivariate semi-wrapped distribution

As it has been indicated previously, in some applications only a subset of variables in a feature vector originates from directional data, the remaining variables may be of linear type. For these situations, a suitable modeling should employ a distribution that is wrapped in the

directional and non-wrapped in the linear dimensions. A multivariate distribution with this property shall be named *multivariate semi-wrapped distribution*. For a simpler notation, it is assumed that the directional variable refers to only one dimension. Let f_w denote the dimension index of it. Then, the multivariate semi-wrapped distribution p_sw of an F-dimensional random vector $\mathbf{x}$ can be defined as

$$p_\mathrm{sw}\left(\mathbf{x}, f_\mathrm{w}\right) = \sum_{k=-\infty}^{\infty} p\left(\mathbf{x} + 2\pi k \mathbf{e}_{f_\mathrm{w}}\right). \tag{5.25}$$

For the sake of completeness, the wrapping index f_w is included as a function argument in p_sw.

5.5.3 Multivariate semi-wrapped Gaussian distribution

For p being the Gaussian PDF, i.e., $p\left(\mathbf{x}\right) = \mathcal{N}_{\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x}}\left(\mathbf{x}\right)$, Equation (5.25) becomes the *multivariate semi-wrapped Gaussian PDF*

$$\mathcal{N}^\mathrm{sw}_{\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x}}\left(\mathbf{x}, f_\mathrm{w}\right) = \sum_{k=-\infty}^{\infty} \left(\left|2\pi\boldsymbol{\Sigma}_\mathbf{x}\right| \exp\left(\left(\mathbf{x} - \boldsymbol{\mu}_\mathbf{x} + 2\pi k \mathbf{e}_{f_\mathrm{w}}\right)^T \boldsymbol{\Sigma}_\mathbf{x}^{-1} \left(\mathbf{x} - \boldsymbol{\mu}_\mathbf{x} + 2\pi k \mathbf{e}_{f_\mathrm{w}}\right)\right)\right)^{-1/2} \tag{5.26}$$

Figure 5.5 (b) shows a plot of a bivariate semi-wrapped Gaussian PDF $\mathcal{N}^\mathrm{sw}_{\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x}}\left(\mathbf{x}, 1\right)$. The abscissa corresponds to the circular variable ϑ, the ordinate to a linear variable x. The reader can verify the wrap on the abscissa.

5.5.4 An approximation to the multivariate semi-wrapped Gaussian distribution

A practical handling of $\mathcal{N}^\mathrm{sw}_{\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x}}$ is rather involved. In particular, the computation of the infinite sum and the estimation of its parameters is a complex task. In order to cope with these problems and in agreement with the approximation derived in Section 5.4.3, a transition from a multivariate semi-wrapped Gaussian distribution to an approximation of it shall be approached.

Again, the assumption for the approximation is a small variance in the circular variable in the sense that neighboring Gaussian terms of Equation (5.26) have only a small overlap. For the multivariate situation this is the case, when $\sqrt{\left(\boldsymbol{\Sigma}_\mathbf{x}\right)_{f_\mathrm{w}, f_\mathrm{w}}} \lesssim 1$.

Under these conditions, $\mathcal{N}^\mathrm{sw}_{\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x}}\left(\mathbf{x}\right)$ can be approximated by only one wrap — in correspondence to Equation (5.17):

$$\mathcal{N}^\mathrm{asw}_{\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x}}\left(\mathbf{x}, f_\mathrm{w}\right) = \left(\left|2\pi\boldsymbol{\Sigma}_\mathbf{x}\right| \exp\left(\left(\left(\mathbf{x} - \boldsymbol{\mu}_\mathbf{x}\right) \bmod {}_{f_\mathrm{w}} 2\pi\right)^T \boldsymbol{\Sigma}_\mathbf{x}^{-1} \left(\left(\mathbf{x} - \boldsymbol{\mu}_\mathbf{x}\right) \bmod {}_{f_\mathrm{w}} 2\pi\right)\right)\right)^{-1/2}. \tag{5.27}$$

The function $\bmod_{f_\mathrm{w}}$ performs the modulo operation solely on the dimension f_w.

5.5.5 Parameter estimates

Parameter estimation refers to the estimation of $\boldsymbol{\mu}_{\mathbf{x}}$ and $\boldsymbol{\Sigma}_{\mathbf{x}}$ from a set $\mathbb{X} = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}\right\}$ of observations $\mathbf{x}^{(m)}$. The non-wrapped and the wrapped elements of $\boldsymbol{\mu}_{\mathbf{x}}$ can be estimated independently. For the non-wrapped ones the linear mean estimates can be used, for the wrapped ones Equation (5.21). In this respect, an exact estimate of $\mu_{\mathbf{x}}$ is obtained by a case distinction

$$(\hat{\mu}_{\mathbf{x}})_f = \begin{cases} \frac{1}{M}\sum_{m=1}^{M} x_f^{(m)} & , \text{if } f \neq f_{\text{w}} \\ \arg\left(\frac{1}{M}\sum_{m=1}^{M} e^{Jx_f^{(m)}}\right)_f & , \text{else} \end{cases} . \tag{5.28}$$

For covariance estimates we again become conscious about the assumption of small variances in the directional variable and present an approximate solution. Combining the linear ML and the univariate estimate of Equation (5.23), we can write

$$\hat{\boldsymbol{\Sigma}}_{\mathbf{x}} \approx \frac{1}{M-1}\sum_{m=1}^{M} \mathbf{x}^{(m)\prime}\mathbf{x}^{(m)\prime T} \tag{5.29}$$

with

$$x_f^{(m)\prime} = \begin{cases} x_f^{(m)} - (\hat{\mu}_{\mathbf{x}})_f & , \text{if } f \neq f_{\text{w}} \\ \left(x_f^{(m)} - (\hat{\mu}_{\mathbf{x}})_f\right) \bmod 2\pi & , \text{else.} \end{cases} \tag{5.30}$$

In summary, Equations (5.28) and (5.29) define the counterparts for the linear ML estimates that are employed during training. Equation (5.27) is used in order to evaluate the probability density $p_{\text{asw}}(\mathbf{t}, f_{\text{w}})$ of an observed feature vector $\mathbf{t}$ during classification.

5.6 Experiments

This section describes experiments using the multivariate semi-wrapped Gaussian PDF, applied to the problem of online handwriting character recognition in the context of the learning framework CSDTW. The same data as described in Section 4.6 has been used.

In order to assess the impact of the described directional feature representation, three different feature extractions have been studied.

1. The first shall help judging about the discriminative power of the tangent slope angle at all. One might argue that the tangent slope angle θ is just a redundant representation of $\tilde{x}$ and $\tilde{y}$ and is thus useless, since it is directly computed from the other two features $\tilde{x}$ and $\tilde{y}$. We want to disprove these considerations by our experiments. To this end, we include a selection comprising only the normalized horizontal and vertical coordinates $\mathbf{t}_i = (\tilde{x}_i, \tilde{y}_i)^T$ in our experiments.

2. A second selection follows the commonly taken approach [Guyon et al., 1991, Schenkel et al., 1995, Jäger et al., 2000] and uses the indirect vector modeling, $\mathbf{t}_i = (\tilde{x}_i, \tilde{y}_i, \cos\theta_i, \sin\theta_i)^T$.

UNIPEN	Feature representation $\mathbf{t}_i$	Mean error rate $\tilde{E}$	Rel. change of $\tilde{E}$
1a (digits)	$(\tilde{x}_i, \tilde{y}_i)^T$	4.2 %	+ 45 %
	$(\tilde{x}_i, \tilde{y}_i, \cos\theta_i, \sin\theta_i)^T$	3.6 %	+ 24 %
	$(\tilde{x}_i, \tilde{y}_i, \theta_i)^T$	2.9 %	
1b (upper case)	$(\tilde{x}_i, \tilde{y}_i)^T$	9.8 %	+ 36 %
	$(\tilde{x}_i, \tilde{y}_i, \cos\theta_i, \sin\theta_i)^T$	7.5 %	+ 4 %
	$(\tilde{x}_i, \tilde{y}_i, \theta_i)^T$	7.2 %	
1c (lower case)	$(\tilde{x}_i, \tilde{y}_i)^T$	13.1 %	+ 40 %
	$(\tilde{x}_i, \tilde{y}_i, \cos\theta_i, \sin\theta_i)^T$	10.1 %	+ 9 %
	$(\tilde{x}_i, \tilde{y}_i, \theta_i)^T$	9.3 %	

TABLE 5.1: Experiments on the *UNIPEN* 1a/b/c sections (as indicated in the first column). The second column denotes the choice of the features. The third column shows the mean error rate $\tilde{E}$ of five different experiments with varying dataset partitionings, the fourth column the change of error rate relative to the here proposed feature set $(\tilde{x}_i, \tilde{y}_i, \theta_i)^T$. The results show that in all sections the proposed feature selection including a direct representation of the directional feature θ_i gives best recognition results.

3. The third approach sets $\mathbf{t}_i = (\tilde{x}_i, \tilde{y}_i, \theta_i)^T$ and applies the described methodology, summarized by Equations (5.27), (5.28) and (5.29).

It is worth noting, that all other recognition parameters were kept constant over the three feature extractions, where possible. In particular, the pre-processing does not vary across the experiments.

Unfortunately, the invariance of the classifier parameters could not be forced *straightforwardly*. The reason for this effect is founded on the allograph clustering. As explained in Section 4.3.2, the diversity of the clustering is controlled by the use of two *clustering thresholds*. However, one of these thresholds includes assumptions about the range of the dissimilarities in the feature space. In different features spaces a constant choice of this threshold would lead to different clustering diversities and thus to a different number of allograph reference models. This, however, would not be a basis for a reasonable comparison.

Thus, the goal was to train classifiers with the objective of getting the same (or a similar) number of allographs by a variation of this particular clustering threshold. Finally, the clustering parameter was chosen such that the overall number of allograph models is about 150, 270 and 600 for the digits, lower and upper case characters, respectively.

Table 5.1 summarizes mean classification error rates $\tilde{E}$ of the three scenarios, each of which is the average from five different dataset partitionings (of the ratio $2:1$, as explained previously).

The following inferences can be drawn from the results in the table:

1. Although the feature θ is computed from the other two features $\tilde{x}$ and $\tilde{y}$, it is shown

that the incorporation of θ significantly improves the recognition accuracy. The relative change of error rate varies from 36 % to 45 %.

2. A direct representation of θ instead of a detour over $(\cos\theta_i, \sin\theta_i)$ results in a higher accuracy in the context of the employed classification. The proposed solution achieved lower error rates in all three *UNIPEN* sections. The relative change of error rate varies from 4 % to 24 %.

Further, the computational complexity and the memory requirements of the $(\cos\theta_i, \sin\theta_i)$-feature extraction are systematically higher compared to a single feature θ_i, as evaluating Equation (5.27) and the storage of $\boldsymbol{\mu}_{\mathbf{x}}$ and $\boldsymbol{\Sigma}_{\mathbf{x}}$ is of order $\mathcal{O}\left(F^2\right)$.

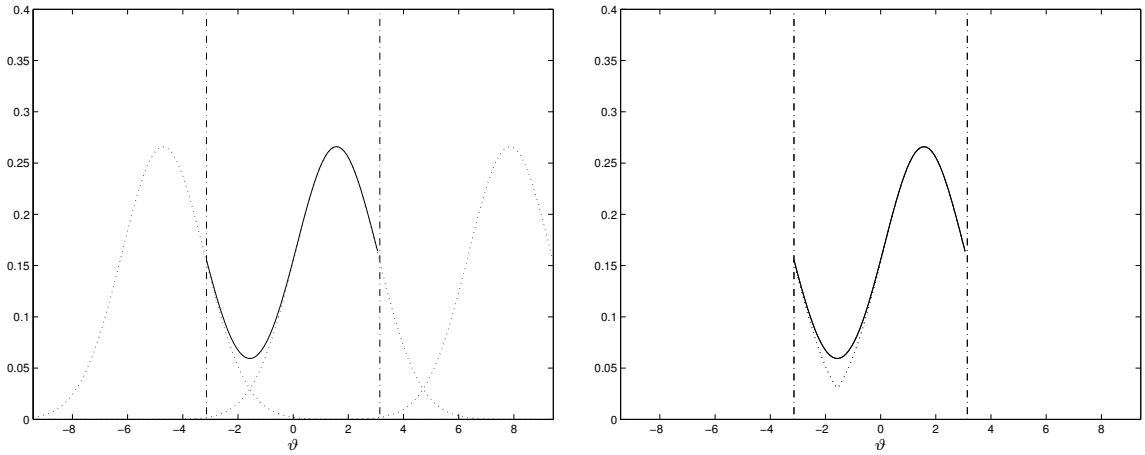
5.7 Summary

This chapter has described a solution for a unified statistical modeling of linear and circular data with a Gaussian-style PDF. In order to reduce the analytic complexity, the approach has been confined to the constraint of small variances in the circular variables.

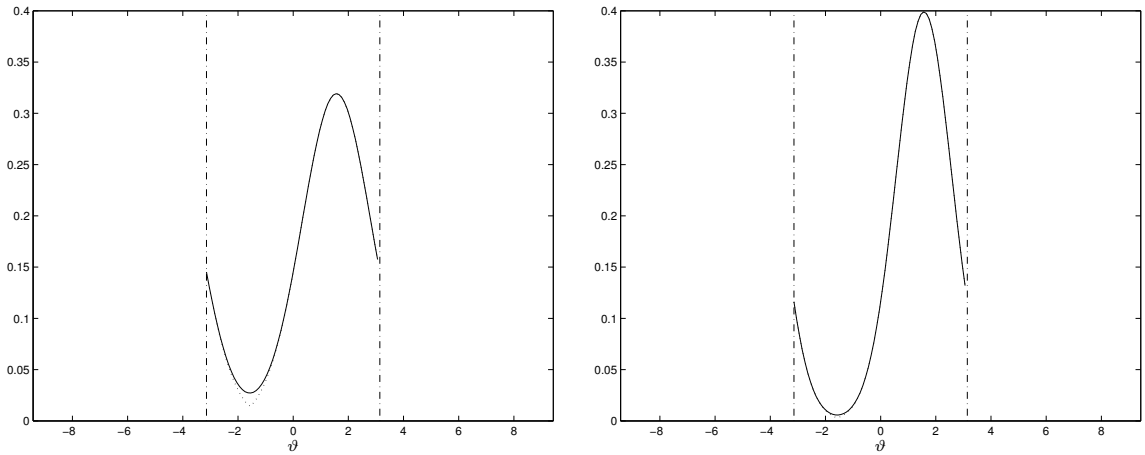
The chapter started with a brief review of directional data, its statistics, and the wrapped Gaussian distribution. Following, a scenario has been formulated in that the wrapped Gaussian distribution can be substituted by an approximation. Approximative solutions to the problem of parameter estimation have been given. Further, extensions of the wrapped Gaussian distribution to multivariate and semi-wrapped situations have been presented. As with the univariate case, a complete framework for an approximative practical handling has been given, including solutions for the tasks of parameter estimation and function evaluation.

It has been shown that the proposed framework significantly improves the recognition accuracy in our application of online HWR. Compared to alternative approaches to the task of incorporating directional data into a statistical feature space modeling errors were relatively reduced by 4 %–24 %. Further benefits of the proposed solution are savings in computation time and memory.

The suggested solution is transferable to many existing HWR systems. It can be plugged into any system that uses a feature representation of the tangent slope angle (or other circular features) in combination with a parametric PDF modeling based on the Gaussian function class. It can be expected that also in those systems the recognition accuracy can generically be increased and the time and memory complexity be decreased with the proposed modeling.



(a) $\sigma_\vartheta^{\mathrm{c}} = 1.50$ (b) $\sigma_\vartheta^{\mathrm{c}} = 1.50$



(c) $\sigma_\vartheta^{\mathrm{c}} = 1.25$ (d) $\sigma_\vartheta^{\mathrm{c}} = 1.00$

FIGURE 5.3: (a): An example of the wrapped Gaussian distribution $\mathcal{N}^{\mathrm{w}}_{\mu_x,\sigma_x}(\vartheta)$ with parameters $\mu_x = \pi/2$ and $\sigma_x = \sqrt{-2\log(1 - V_\vartheta^{\mathrm{c}})} = 1.50$ (solid line). The dashed lines show three contributing terms in Equation (5.14), corresponding to $k = -1$, $k = 0$ and $k = 1$. (b)–(d): Wrapped Gaussian $\mathcal{N}^{\mathrm{w}}_{\mu_x,\sigma_x}(\vartheta)$ (solid line) together with their approximation $\mathcal{N}^{\mathrm{aw}}_{\mu_x,\sigma_x}(\vartheta)$ (dotted line) with parameters $\mu_x = \pi/2$ and $\sigma_x \in \{1.50, 1.25, 1.00\}$. It can be seen that for $\sigma_x = 1.0$ approximation errors are small.

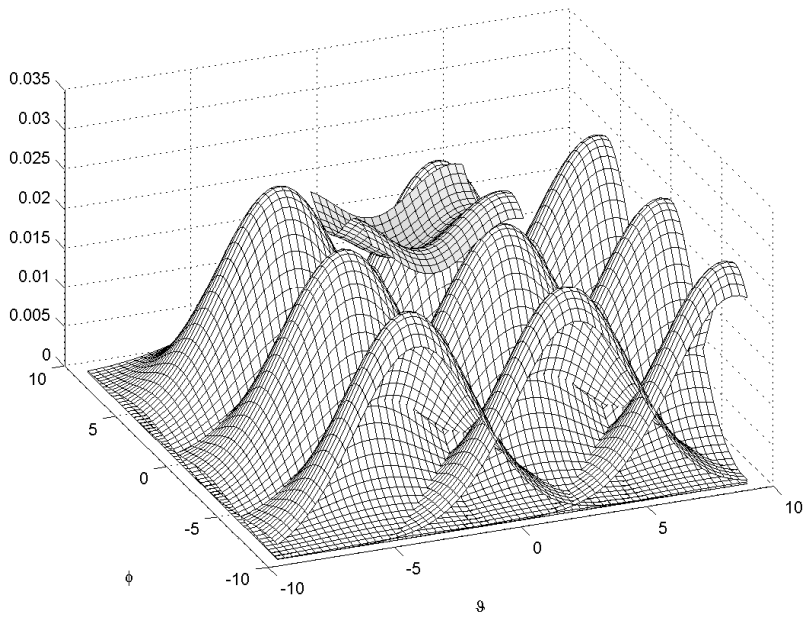
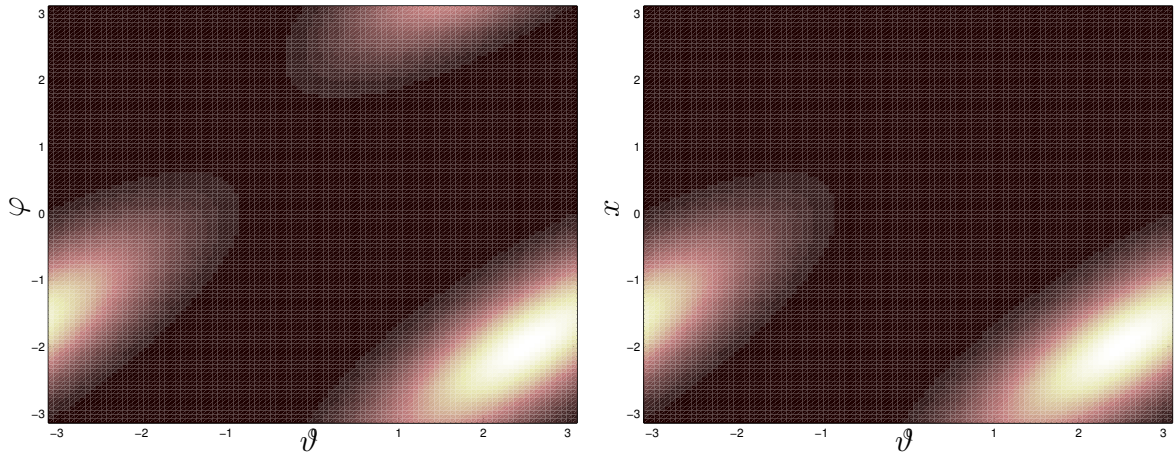


FIGURE 5.4: An example for a bivariate wrapped PDF $\mathcal{N}^{\mathrm{w}}_{\boldsymbol{\mu}_{\mathbf{x}},\boldsymbol{\Sigma}_{\mathbf{x}}}$ is shown by the small patch above the (comparably small) interval $(-\pi,\pi] \times (-\pi,\pi]$. In correspondence to Figure 5.3 (a), the figure additionally gives a plot of nine summands of Equation (5.24), corresponding to $k_1 = -1, 0, 1$ and $k_2 = -1, 0, 1$ in Equation (5.24). The parameter settings in this figure are $\boldsymbol{\mu}_{\mathbf{x}} = (2.5, -2.0)^T$ and $\boldsymbol{\Sigma}_{\mathbf{x}} = \begin{bmatrix} 6 & 0.7 \\ 0.7 & 6 \end{bmatrix}$.



(a) Wrapped Gaussian distribution

(b) Semi-wrapped Gaussian distribution

FIGURE 5.5: Pseudo-color plots of (a) a wrapped $\mathcal{N}^{\mathrm{w}}_{\boldsymbol{\mu}_{\mathbf{x}},\boldsymbol{\Sigma}_{\mathbf{x}}}(\mathbf{x})$ and (b) a semi-wrapped $\mathcal{N}^{\mathrm{sw}}_{\boldsymbol{\mu}_{\mathbf{x}},\boldsymbol{\Sigma}_{\mathbf{x}}}(\mathbf{x},1)$ multivariate Gaussian PDF with the parameters $\boldsymbol{\mu}_{\mathbf{x}} = (2.5,-2.0)^T$ and $\boldsymbol{\Sigma}_{\mathbf{x}} = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 0.8 \end{bmatrix}$. Bright colors denote high values. In both cases the abscissa corresponds to a circular variable. The ordinate corresponds to a circular variable in (a), however to a linear in (b). Note that for the chosen values $(\boldsymbol{\Sigma}_{\mathbf{x}})_{i,i} \leq 1$, $i = 1,2$ the assumption of a "small variance" for the circular quantity is valid and an approximation of $\mathcal{N}^{\mathrm{sw}}_{\boldsymbol{\mu}_{\mathbf{x}},\boldsymbol{\Sigma}_{\mathbf{x}}}(\mathbf{x},1)$ by $\mathcal{N}^{\mathrm{asw}}_{\boldsymbol{\mu}_{\mathbf{x}},\boldsymbol{\Sigma}_{\mathbf{x}}}(\mathbf{x},1)$ is justified.

CHAPTER 6

A (dis-) similarity measure for CSDTW and hidden Markov models

This chapter describes a similarity (and equivalently a dissimilarity) measure for CSDTW models. As the CSDTW is very closely related to hidden Markov modeling the concept can also be transferred to particular HMM implementations. Similar to the CSDTW scoring, the CSDTW model dissimilarity computation is based on dynamic programming. The Viterbi algorithm identifies an optimal alignment through a lattice of local (dis-) similarities. The main difference to the CSDTW scoring is that the local (dis-) similarities are dependent on two *PDFs as their arguments. In the present approach, the local measures are defined with help of the Bayes probability of error. Applications of the described (dis-) similarity measure are manifold. It can be applied as a stop criterion in the iterative CSDTW training, as a speed-up in classification, a distance measure in the context of CSDTW model clustering or as an optimization criterion for a discriminative CSDTW training. Further, as the present similarity measure is based on the Bayes probability of error, it can be utilized as a tool to interpret misclassifications. The similarity measure is experimentally evaluated in the context of online HWR and compared with the error rates in the lowercase character classification problem. This evaluation affirms the usefulness of the proposed measure, as it shows that class pairs with a high similarity measure are often confused during classification.*

6.1 Introduction

In Chapter 4 the generative classification approach CSDTW has been introduced and applied to online HWR. It has been shown to be very powerful for the classification of handwritten characters. Like with many HMM based methods the success is in large part due to the ca-

pability of a nonlinear alignment of two sequences. The complex structure of this concept, however, makes it difficult to give a unique measure of (dis-) similarity between two (or more) CSDTW models. The present chapter shows a solution for this aim.

Applications that can use a CSDTW model (dis-) similarity are manifold. They include the following:

1. The iterative training process, as e.g. it has been introduced for CSDTW in Algorithm 5, can be monitored and controlled.

2. It can be used as a discriminative training criterion for the CSDTW parameter estimation. See, for instance, the work of Kwong et al. [1998] for an example in the context of HMMs.

3. It can serve as a distance measure for a CSDTW model clustering.

4. It could speed up the minimization of Equation (4.19) during CSDTW classification. In this respect, the computation of $D^*[\hat{d}]\left(\boldsymbol{t}, \mathcal{R}^{lk}\right)$ can be stopped if $\mathcal{R}^{lk}$ is similar to a particular $\mathcal{R}^{l'k'}$ and $D^*[\hat{d}]\left(\boldsymbol{t}, \mathcal{R}^{l'k'}\right)$ has been found to be relatively large.

5. It can help to get a thorough insight into misclassifications.

The latter will constitute the application of interest for this chapter.

6.2 Literature review

Some concepts of a (dis-) similarity measure for statistical sequence models have been studied in literature, mostly in the context of HMMs.

Levinson et al. [1983] have been the first to propose an HMM distance measure. Given two discrete HMMs λ_1 and λ_2, they defined a dissimilarity measure Δ as

$$\Delta\left(\lambda_1, \lambda_2\right) = \left(\frac{1}{O N_S} \sum_{n=1}^{N_S} \sum_{o=1}^{O} \left(b_{no}^{(1)} - b_{p(n)o}^{(2)} \right) \right)^{1/2} . \tag{6.1}$$

Here, $b_{no}^{(l)}$ is the discrete observation probability of the HMM λ_l for a symbol, which is indexed by $o \in \{1, \ldots, O\}$, while being in the state $q_n \in \{q_1, \ldots, q_{N_S}\}$. The authors introduce $p\left(n\right)$ as "a state permutation that minimizes" Equation (6.1). Apparently, one imperfection of this approach is that only the observation probabilities $b_{no}^{(l)}$ and not the additional characteristics of the HMM, in particular the HMM topology and the state transition probabilities, are addressed.

Juang and Rabiner [1985] introduce and evaluate the dissimilarity

$$\Delta'\left(\lambda_1, \lambda_2\right) = \lim_{N \to \infty} \frac{1}{N} \left(\ln P\left(\boldsymbol{t}^{(2)} | \lambda_1\right) - \ln P\left(\boldsymbol{t}^{(2)} | \lambda_2\right) \right), \tag{6.2}$$

where $\boldsymbol{t}^{(2)} = \left[\mathbf{t}_1^{(2)}, \ldots, \mathbf{t}_N^{(2)}\right]$, $\mathbf{t}_i^{(2)} \in \mathbb{R}^F$ is an observation sequence generated by the HMM λ_2. Equation (6.2) can be interpreted in terms of cross-entropy, divergence or discrimination information. The authors also give a summary of this approach in their textbook [Rabiner and Juang, 1993, Section 6.11].

Indeed, the latter approach has found application in a number of examples. Falkhausen et al. [1995] identify "similar" HMMs for their problem of speech recognition. The authors also study a decision directed variant of Equation (6.2) in a further set of experiments.

Singer and Warmuth [1999] integrate a variant of Equation (6.2) into their iterative HMM parameter estimation. The aim is here to get a faster convergence for the iteration. The authors report positive results with respect to this aim in speech recognition problems.

Kwong et al. [1998] have defined a discriminative optimization criterion based on Equation (6.2) in the context of HMM parameter estimation. The authors call it *maximum model distance* (MMD) and employ it in the area of speech recognition.

Another distance measure is defined on *co-emission probabilities* [Lyngsø et al., 1999], in particular

$$\Delta''(\lambda_1, \lambda_2) = \sum_{t \in \mathbb{X}} P(\boldsymbol{t}|\lambda_1)\, P(\boldsymbol{t}|\lambda_2)\,. \tag{6.3}$$

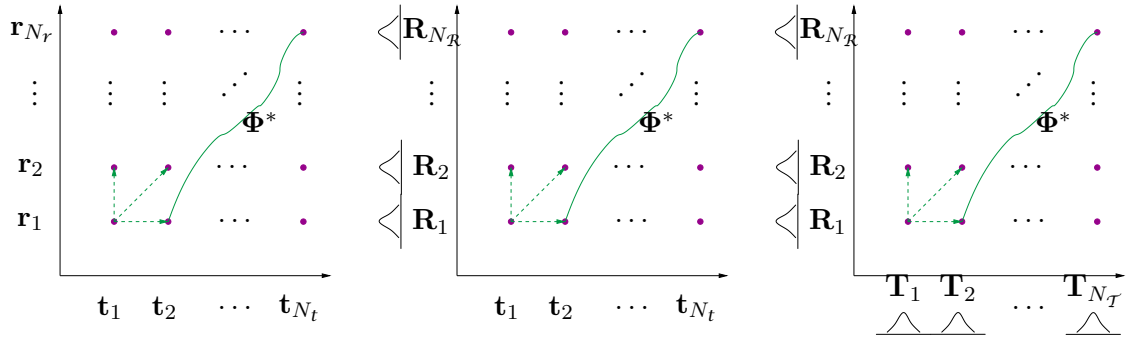
The authors show experiments based on Equation (6.3) in the context of protein processing.

6.3 The CSDTW model dissimilarity

The primary objective of this section is to describe how — with a slight modification — the Viterbi framework used for the CSDTW classification can be applied to define a CSDTW model dissimilarity [Bahlmann and Burkhardt, 2001]. Later on, it will further be shown how the dissimilarity measure can straightforwardly be transformed into a similarity measure, and how it can be interpreted as a particular probability.

The principle idea shall be illustrated in a simple sketch, as is shown in Figure 6.1. Parts (a) and (b) review the basic scheme of computing the DTW (Equation (2.32)) and the SDTW (Equation (4.11)) distance, respectively. In both cases the Viterbi algorithm (or the Viterbi beam search algorithm) is used with the aim of uncovering the Viterbi alignment through a matrix of local distances. As discussed, those distances are computed from the respective elements of the two questioned sequences. The difference between the DTW and SDTW dissimilarity is the representation of the reference pattern and, in consequence, the definition of the local distance: simple templates $\mathbf{r}_j$ and $d(\mathbf{t}_i, \mathbf{r}_j)$, defined by Equation (2.34), are addressed in the first and a tuple $\mathbf{R}_j = \left(\alpha_j^{\mathcal{R}}, \beta_j^{\mathcal{R}}\right)$ of statistical quantities and $\hat{d}(\mathbf{t}_i, \Delta\boldsymbol{\phi}, \mathbf{R}_j)$, defined by Equation (4.12), in the second case.

The here described CSDTW model dissimilarity measure (part (c)) goes one step further. Recall that in the aimed situation two CSDTW models — say $\mathcal{T} = \left(\mathbf{T}_1, \ldots, \mathbf{T}_{N_\mathcal{T}}\right)$ with $\mathbf{T}_i = \left(\alpha_i^{\mathcal{T}}, \beta_i^{\mathcal{T}}\right)$ and $\mathcal{R} = (\mathbf{R}_1, \ldots, \mathbf{R}_{N_\mathcal{R}})$ with $\mathbf{R}_j = \left(\alpha_j^{\mathcal{R}}, \beta_j^{\mathcal{R}}\right)$ — are the center of interest. To put it back into memory (cf. Section 4.3.1.2), $\alpha_i^{\mathcal{T}}$ and $\alpha_j^{\mathcal{R}}$ statistically model the align-



(a) DTW distance (b) SDTW distance (c) CSDTW model dissimilarity

FIGURE 6.1: Comparing the ideas of the DTW, the SDTW and the CSDTW model dissimilarity: (a) The DTW local distances are computed from two pattern templates $\mathbf{t}_i$ and $\mathbf{r}_j$. (b) The SDTW local distances are computed from pattern templates $\mathbf{t}_i$ and a statistical tuple $\mathbf{R}_j = \left(\alpha_j^\mathcal{R}, \beta_j^\mathcal{R}\right)$, here indicated by a Gaussian shape. (c) In the CSDTW model dissimilarity measure local distances are computed from two statistic tuples $\mathbf{T}_i = \left(\alpha_i^\mathcal{T}, \beta_i^\mathcal{T}\right)$ and $\mathbf{R}_j = \left(\alpha_j^\mathcal{R}, \beta_j^\mathcal{R}\right)$. Whereas local distances are defined differently the principle of the Viterbi search is employed throughout all concepts.

ment transitions and $\beta_i^\mathcal{T}$ and $\beta_j^\mathcal{R}$ describe a PDF in the feature sub-space $\mathbb{R}^F$. With this in mind, a similar Viterbi framework like the one defined by Equations (4.10) and (4.11), can be formulated:

$$\tilde{D}_{\mathbf{\Phi}}[\bar{d}]\left(\mathcal{T}, \mathcal{R}\right) = \frac{1}{N}\sum_{n=1}^{N} \bar{d}\left(\mathbf{T}_{\phi_\mathcal{T}(n)}, \mathbf{R}_{\phi_\mathcal{R}(n)}\right) \tag{6.4}$$

$$\tilde{D}^*[\bar{d}]\left(\mathcal{T}, \mathcal{R}\right) = \tilde{D}_{\mathbf{\Phi}^*}[\bar{d}]\left(\mathcal{T}, \mathcal{R}\right) = \min_{\mathbf{\Phi}}\left\{\tilde{D}_{\mathbf{\Phi}}[\bar{d}]\left(\mathcal{T}, \mathcal{R}\right)\right\}, \tag{6.5}$$

along with the same symmetric transition set

$$\mathbb{P} = \left\{(1,0), (0,1), (1,1)\right\}. \tag{6.6}$$

Note that the definition of Equation (6.5) employs the path length normalized variant of the warping distance, as introduced in Equation (2.35).

The remaining challenge is to adopt the local distances $\bar{d}\left(\mathbf{T}_i, \mathbf{R}_j\right)$ of the here two tuples $\mathbf{T}_i = \left(\alpha_i^\mathcal{T}, \beta_i^\mathcal{T}\right)$ and $\mathbf{R}_j = \left(\alpha_j^\mathcal{R}, \beta_j^\mathcal{R}\right)$ to this new situation. For the sake of simplicity, a probabilistic modeling of the transitions shall be dropped. With this simplification, we are faced with the problem of determining the dissimilarity of two PDFs, i.e.,

$$\bar{d}\left(\mathbf{T}_i, \mathbf{R}_j\right) = \bar{d}\left(\beta_i^\mathcal{T}, \beta_i^\mathcal{R}\right). \tag{6.7}$$

A variety of choices for $\bar{d}$ are possible. Literature has studied e.g., the χ^2 measure, the Kullback-Leibler [Kullback and Leibler, 1951] or the Jensen-Shannon divergence [Lin, 1991].

In the present context, the CSDTW model dissimilarities shall be interpreted in comparison to empirically observed classification errors. As reviewed in Section 2.2.1.2, a measure for the probability of a classification error in a two-class problem and a vector space is the Bayes probability of error $P_e(1,2)$.

Conceptually, the Bayes probability of error is a similarity rather than a dissimilarity measure, as similar PDFs go in hand with a high error probability. Hence, in order to define a dissimilarity measure, its complement $1-2P_e(1,2)$ is a suitable substitute. As can be taken from Figure 2.2, $1-2P_e(1,2)$ corresponds to the area outside the overlap of the two PDFs. With the understanding that in the Bayesian modeling a class l is characterized by the tuple of prior probability $P(l)$ and PDF $p(\mathbf{x}|l)$, i.e.,

$$l \hat{=} \left(P(l), p(\mathbf{x}|l)\right), \tag{6.8}$$

one reasonable choice for $\bar{d}$ is — in loose agreement with Equation (2.8) and in the case of uniform prior probabilities —

$$\begin{aligned} \bar{d}(\mathbf{T}_i, \mathbf{R}_j) &= 1 - 2P_e\left(\left(\frac{1}{2}, \beta_i^{\mathcal{T}}(\mathbf{x})\right), \left(\frac{1}{2}, \beta_j^{\mathcal{R}}(\mathbf{x})\right)\right) \\ &= 1 - 2\int_x \min\left\{\frac{1}{2}\beta_i^{\mathcal{T}}(\mathbf{x}), \frac{1}{2}\beta_j^{\mathcal{R}}(\mathbf{x})\right\} d\mathbf{x}. \end{aligned} \tag{6.9}$$

If prior probabilities of the classification problem, e.g. by $P\left(\mathcal{T}\right)$ and $P(\mathcal{R})$, are available, the distance can be refined to

$$\begin{aligned} &\bar{d}\left(P\left(\mathcal{T}\right), \mathbf{T}_i, P(\mathcal{R}), \mathbf{R}_j\right) \\ &= 1 - 2P_e\left(\left(P\left(\mathcal{T}\right), \beta_i^{\mathcal{T}}(\mathbf{x})\right), \left(P(\mathcal{R}), \beta_j^{\mathcal{R}}(\mathbf{x})\right)\right) \\ &= 1 - 2\int_x \min\left\{P\left(\mathcal{T}\right)\beta_i^{\mathcal{T}}(\mathbf{x}), P(\mathcal{R})\beta_j^{\mathcal{R}}(\mathbf{x})\right\} d\mathbf{x} \end{aligned} \tag{6.10}$$

Indeed, the latter shall be used as local distance score for the two PDFs, which in combination with Equations (6.4), (6.5), (6.6) and (6.10) define the framework for the CSDTW dissimilarity measure.

Like in the CSDTW classification, the directional characteristic of the tangent slope angle feature θ has to be respected, when integrating in Equation (6.10). Also for this task, the use of the approximated semi-wrapped Gaussian of equation (5.27) gives a straightforward solution to this task.

One additional aspect is worth mentioning: although the probabilistic modeling of the transitions has been dropped by the acceptance of Equation (6.7), the fundamental warping capabilities of CSDTW have not. The set $\mathbb{P}$ of possible transitions (in other words: the model topology) remains the same as in the CSDTW classification.

In Section 4.3.1.4, it has been argued that CSDTW is equivalent to particular HMMs, in particular those with null-transitions. In this respect, the described dissimilarity measure is straightforwardly transferable to this widespread tool for probabilistic sequence modeling.

6.4 The CSDTW model similarity

In some situations, one is interested in a *similarity* measure of two CSDTW models instead of a *dissimilarity* measure. In these cases, $\tilde{D}^*[\bar{d}]\left(\mathcal{T},\mathcal{R}\right)$ can be transformed to a complementary expression, which shall be named

$$\tilde{P}_e\left(\mathcal{T},\mathcal{R}\right)=\frac{1}{2}\left(1-\tilde{D}^*[\bar{d}]\left(\mathcal{T},\mathcal{R}\right)\right). \tag{6.11}$$

Some simple transformations lead to

$$\begin{aligned}
&\frac{1}{2}\left(1-\tilde{D}^*[\bar{d}]\left(\mathcal{T},\mathcal{R}\right)\right)\\
&=\frac{1}{2}\left(1-\frac{1}{N^*}\sum_{n=1}^{N^*}\bar{d}\left(\mathbf{T}_{\phi^*_{\mathcal{T}}(n)},\mathbf{R}_{\phi^*_{\mathcal{R}}(n)}\right)\right)\\
&=\frac{1}{2}\left(1-\frac{1}{N^*}\sum_{n=1}^{N^*}\left(1-2P_e\left(\left(P\left(\mathcal{T}\right),\beta^{\mathcal{T}}_{\phi^*_{\mathcal{T}}(n)}\right),\left(P\left(\mathcal{R}\right),\beta^{\mathcal{R}}_{\phi^*_{\mathcal{R}}(n)}\right)\right)\right)\right)\\
&=\frac{1}{N^*}\sum_{n=1}^{N^*}P_e\left(\left(P\left(\mathcal{T}\right),\beta^{\mathcal{T}}_{\phi^*_{\mathcal{T}}(n)}\right),\left(P\left(\mathcal{R}\right),\beta^{\mathcal{R}}_{\phi^*_{\mathcal{R}}(n)}\right)\right),
\end{aligned} \tag{6.12}$$

Thus, it is shown, that $\tilde{P}_e\left(\mathcal{T},\mathcal{R}\right)$ equals the *average* Bayes error $P_e\left(\left(P\left(\mathcal{T}\right),\beta^{\mathcal{T}}_{\phi^*_{\mathcal{T}}(n)}\right),\left(P\left(\mathcal{R}\right),\beta^{\mathcal{R}}_{\phi^*_{\mathcal{R}}(n)}\right)\right)$ of the respective feature sub-spaces $\mathbb{R}^F$ along the Viterbi path. Obviously, the value of $\tilde{P}_e\left(\mathcal{T},\mathcal{R}\right)$ is in the range $[0,0.5]$.

However, it is important to note that $\tilde{P}_e\left(\mathcal{T},\mathcal{R}\right)$ does *not* equal to the Bayes probability of error — say $P_e\left(\mathcal{T},\mathcal{R}\right)$ — in a two-class classification problem of two sequences $\mathcal{T}$ and $\mathcal{R}$. Compared to the above described measure, a derivation of $P_e\left(\mathcal{T},\mathcal{R}\right)$ is theoretically and computationally rather involved. Though, it remains as an interesting field for future research. To sketch a starting point for this interesting research, one would — in correspondence to Equation (2.8) — define $P_e\left(\mathcal{T},\mathcal{R}\right)$ as

$$P_e\left(\mathcal{T},\mathcal{R}\right)=\int_{\mathcal{X}}\min\left\{P\left(\mathcal{T}\right)p\left(\boldsymbol{x}|\mathcal{T}\right),P\left(\mathcal{R}\right)p\left(\boldsymbol{x}|\mathcal{R}\right)\right\}d\boldsymbol{x}, \tag{6.13}$$

and factorize $p\left(\boldsymbol{x}|\mathcal{T}\right)$ and $p\left(\boldsymbol{x}|\mathcal{R}\right)$ similarly to the procedure within Proposition 4.1.

6.5 Analysis of complexity

The computation of the CSDTW model dissimilarity can utilize the same Viterbi beam search framework as the CSDTW classification, as the only significant modification concerns the definition of the local distance. Thus, in correspondence to Equation (4.36), the computational complexity of the CSDTW dissimilarity computation of one model pair is

$$C_{\text{Time}}\left(\tilde{D}^{*}[\bar{d}]\left(\mathcal{T},\mathcal{R}\right)\right)=\mathcal{O}(\tilde{N}\cdot|\mathbb{P}|\cdot G^{F}), \tag{6.14}$$

with $\tilde{N}$ the average length of the CSDTW models, $|\mathbb{P}|$ the number of transitions, F the dimension of the feature sub-space and G the number of sampling points per dimension for the integration of Equation (6.10). In Equation (6.14), the term $\mathcal{O}(\tilde{N}\cdot|\mathbb{P}|)$ corresponds to the beam search framework and $\mathcal{O}(G^{F})$ to the local distance computation, that is, the (naive) numerical integration of the feature space $\mathbb{R}^{F}$.

The average run-time for the computation of one CSDTW dissimilarity on an AMD Athlon 1600MHz in the $F=3$ dimensional feature space, has been evaluated to

$$\text{Time}\left(\tilde{D}^{*}[\bar{d}]\left(\mathcal{T},\mathcal{R}\right)\right)\approx 2\,\text{sec}. \tag{6.15}$$

Hereby, a numerical integration of $25\times 25\times 9$ (corresponding to $\tilde{x},\tilde{y},\theta$) grid points has been pursued. This resolution has been determined through an exemplary, visual inspection of the Gaussians. It should be stated that the applications for the CSDTW dissimilarity measure mainly arise during training and thus do not require real-time computation.

6.6 Experiments

For a practical evaluation, the similarity measure of Equation (6.11) has been computed for CSDTW model pairs of an entire CSDTW classifier. The classifier has been trained from the lowercase characters (section 1c) of the *UNIPEN* "Train-R01/V07" database (cf. Sections 3.3 and 4.6). The clustering parameter configuration had produced a classifier with $A^{\text{tot}}=237$ allograph models. Figure 6.2 shows a representative example of the Viterbi alignment of two CSDTW models, here for a "u" and an "a". One can verify that the Viterbi path also corresponds to our intuitive judgment about a probable PDF alignment.

In order to get an idea of its meaningfulness, the experimentally computed CSDTW model similarity values $\tilde{P}_{e}\left(\mathcal{R}^{l'k'},\mathcal{R}^{lk}\right)$ have been compared to empirical classification errors. In this respect, it can be expected that classes that have *similar* CSDTW models, produce rather high errors. In the following, classification errors are quantified through the classification confusion matrix $\mathbf{C}=[C_{l'l}]_{L\times L}$, the elements $C_{l'l}$ of which indicate the number of samples of class l recognized as class l' during the classification experiment.

In the experiments, $\tilde{P}_{e}\left(\mathcal{R}^{l'k'},\mathcal{R}^{lk}\right)$ has been computed for any of the $(A^{\text{tot}})^{2}=237^{2}=56169$ model pairs $\left(\mathcal{R}^{l'k'},\mathcal{R}^{lk}\right)$. Further, $C_{l'l}$ has been counted for all of the $L^{2}=26^{2}=676$

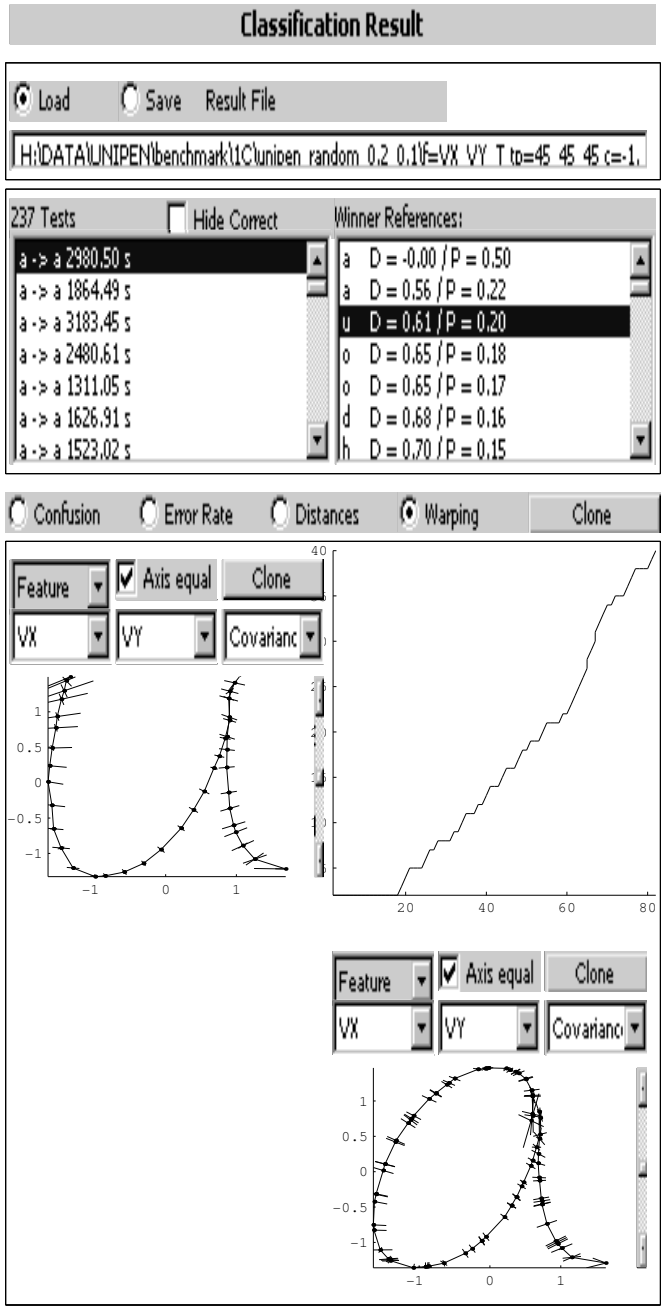


FIGURE 6.2: CSDTW model dissimilarity computation: in the lower part of this figure two CSDTW allograph models $\mathcal{T}$ (of class "a") and $\mathcal{R}$ (of class "u") are illustrated. For an interpretation of this illustration refer to the caption of Figure 4.1(a) on page 55. The CSDTW model dissimilarity is $\tilde{D}^*[\bar{d}]\left(\mathcal{T},\mathcal{R}\right) = 0.61$ and the similarity $\tilde{P}_e\left(\mathcal{T},\mathcal{R}\right) = 0.20$. The line right to the "u" and above the "a" shows the corresponding Viterbi alignment, which indeed coincides with our intuitive judgment. In the upper part the list to the right shows the "top 7" similarity scores relative to the given "a". The first entry in the list ($\tilde{D}^*[\bar{d}] = 0$, $\tilde{P}_e = 0.5$) corresponds to the distance of the "a" to itself.

class pairs. In this context, both tasks were performed on the same "training" dataset. In order to have a common basis for a comparison of the classification confusions and the CSDTW model similarity, a few heuristically motivated post-processing steps have been pursued:

1. For a straightforward comparison of error rate and model similarity, a single value $C_{l'l}$ shall correspond to a single value $\tilde{P}_e(l', l)$. However, the CSDTW models are available sub-class wise, while the classification confusions only class wise. To overcome this incompatibility, any set of $K_{l'} \times K_l$ allograph similarity values $\left\{\tilde{P}_e\left(\mathcal{R}^{l'k'}, \mathcal{R}^{lk}\right)\right\}_{k'=1,\ldots,K_{l'},k=1,\ldots,K_l}$ has been averaged to one scalar value $\tilde{P}_e(l', l)$, taking into account the normalized allograph prior probabilities $\tilde{\pi}_{lk} = \frac{P\left(\mathcal{R}^{lk}\right)}{\tilde{\pi}_l}$ (with $\tilde{\pi}_l = \sum_k P\left(\mathcal{R}^{lk}\right)$):

$$\tilde{P}_e(l', l) = \sum_{k'=1}^{K_{l'}} \tilde{\pi}_{l'k'} \sum_{k=1}^{K_l} \tilde{\pi}_{lk} \tilde{P}_e\left(\mathcal{R}^{l'k'}, \mathcal{R}^{lk}\right). \tag{6.16}$$

   The normalization ensures that still $\tilde{P}_e(l', l) \in [0, 0.5]$ is valid.

2. The number of classification confusions shall be normalized to the empirical error rate. As the CSDTW similarity is based on the Bayes error of a *two-class* classification problem, the normalization of the classification confusions shall also only take into account the respective two classes. Hence, the *two-class error rate* $C'_{l'l}$ for a classification with respect to only the two classes l' and l is defined as

$$C'_{l'l} = C_{l'l} / \left(C_{l'l} + C_{ll}\right). \tag{6.17}$$

3. Since $\tilde{P}_e(l', l)$ is the probability of falsely classifying class l' into l *or* l into l', the *symmetric two-class error rate* $\tilde{C}_{l'l}$ is defined as

$$\tilde{C}_{l'l} = \tilde{\pi}'_{l'l} C'_{l'l} + \tilde{\pi}'_{ll'} C'_{ll'}. \tag{6.18}$$

   $\tilde{\pi}'_{l'l} = \frac{\tilde{\pi}_l}{\tilde{\pi}_l + \tilde{\pi}_{l'}}$ is the prior probability of class l in the two-class context of l and l'.

Finally, a comparison of $\tilde{C}_{l'l}$ and $\tilde{P}_e(l', l)$ is carried out in Figure 6.3. At this point, we should again remark that $\tilde{P}_e\left(\mathcal{T}, \mathcal{R}\right)$ does not correspond to the probability of a classification error with respect to the sequence classification problem, but is the average of all sub-space Bayes probabilities of error along the Viterbi path. In this respect, a quantitative comparison of $\tilde{P}_e(l', l)$ and $\tilde{\mathbf{C}}_{l'l}$ is not appropriate. Nevertheless, the results shall be interpreted from a qualitative point of view. With this in mind, the following conclusions can be drawn from Figure 6.3:

- For 15 out of 26 classes the most similar and most frequently confused characters coincide (e.g., the "u" is the most *similar* and the most frequently confused character class to the "a").

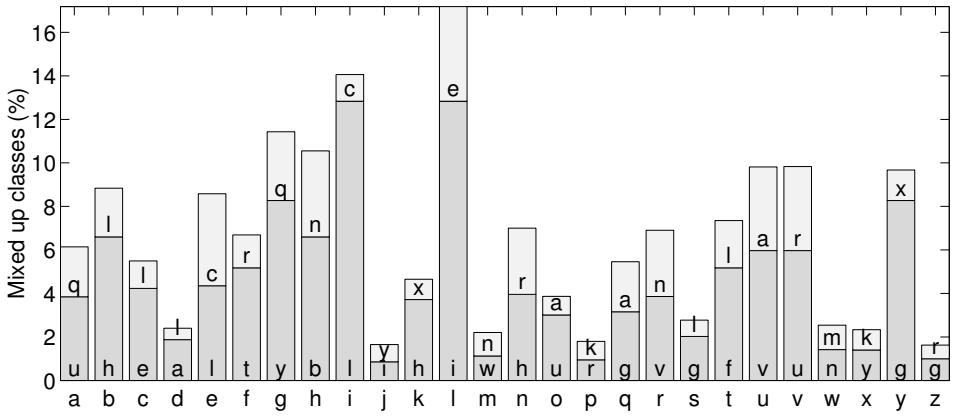
- For 25 out of 26 classes the sets of the two most similar and two most frequently mixed up classes share at least one character.
- Further, high values for $\tilde{P}_e\left(l', l\right)$ also coincide with our intuitive judgment of similarity between l' and l, as can also be seen from Figure 6.2 and other examples (not illustrated here).

In conclusion, the observations realized above indicate that $\tilde{D}^*[\bar{d}]\left(\mathcal{T}, \mathcal{R}\right)$ ($\tilde{P}_e\left(\mathcal{T}, \mathcal{R}\right)$) can advantageously be used as a CSDTW model dissimilarity (similarity) measure.

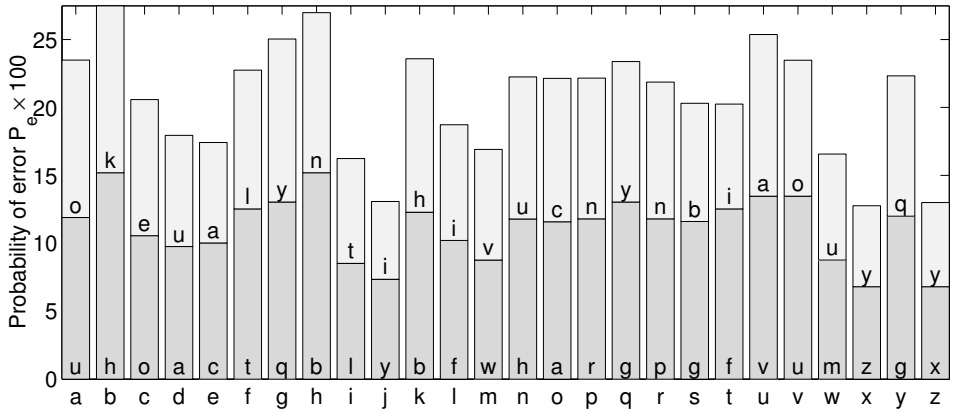
6.7 Summary

In a number of applications, there is a demand for a measure that determines the (dis-) similarity of two generative sequence models, such as CSDTW models or particular HMMs. This chapter has presented a solution for these demands. The presented measure is based on the Bayes probability of error with respect to the underlying sub-space PDFs. One advisable property of the presented solution is that the computation algorithm employs the same Viterbi framework as used for classification, in this case in order to combine multiple evaluations of the Bayes error. In the present context, the (dis-) similarity measure is used to analyze misclassifications in the context of online HWR. First, this is achieved by the interpretation the Viterbi path and the respective PDF correspondences. Second, experiments have shown that CSDTW model pairs, that are similar with respect to the described measure, also correspond to frequently confused classes.

The suggested method is very versatile, that is, the generative sequence model can be based on any PDF, for instance, simple Gaussians, mixed Gaussians or discrete probabilities. Further, the Bayes probability of error as the dissimilarity between two PDFs in the feature subspace can be substituted by other suitable distances, adaptable to the targeted application. Appropriate distances include, for example, χ^2, the Kullback-Leibler or Jensen-Shannon divergence.



(a) Top 2 confused classes $\tilde{C}$



(b) Top 2 Similarities $\tilde{P}_e$

FIGURE 6.3: (a) The bar diagram shows the two highest symmetric two-class error rates $\tilde{C}_{l'l}$ for every character $l \in \{\text{"a"}, \ldots, \text{"z"}\}$, as labeled on the abscissa. (b) In a similar fashion, the bars represent the similarities $\tilde{P}_e\,(l', l)$ of the two most similar classes for every character $l \in \{\text{"a"}, \ldots, \text{"z"}\}$. For many cases, one can observe a qualitative correlation of most frequently confused and most similar class pairs.

CHAPTER 7

SVM-GDTW — A discriminative sequence classification framework

This chapter describes a solution for the classification of sequence data which follows the discriminative classification paradigm. The approach combines dynamic time warping and support vector machines by the formulation of a particular SVM kernel — the Gaussian dynamic time warping (GDTW) kernel. As the classification approach is a pure discriminative one, it does not assume a model for the generative class conditional densities, like common HMM based techniques do. Instead, it addresses the direct creation of class boundaries. In this respect, the benefits of the discriminative paradigm apply to the described method. From another point of view, the approach can be regarded as an example for the incorporation of prior problem knowledge and invariances into SVM classification. The DTW distance imposes a dissimilarity measure that is specifically suited to typical distortion models of online handwriting data, namely nonlinear temporal distortions. Unlike other SVM kernels (e.g., the Gaussian or polynomial kernel) the GDTW kernel is not a valid kernel in a strict mathematical sense, as it is not positive definite. However, it is shown to behave very well in an online handwriting recognition application. Practical arguments for this behavior are given.

7.1 Introduction

For the solution of online HWR tasks, researchers mostly apply classification methods which are based on the generative approach, like it has been reviewed in Section 2.2.2: HMMs are employed to model class conditional PDFs which are based on — and thus restricted to — a certain function class. A discriminant function is obtained in a second step using Bayes' rule (Equation (2.4)). Indeed, HMMs and related approaches have proven to deal very well with the

complex online handwriting data structure. This is because HMM based methods specifically address characteristics of sequential data and typical variations, which are nonlinear temporal distortions. Also the CSDTW approach, as it has been introduced in Chapter 4, is founded on this generative paradigm.

In fact, as it has been pointed out in Section 2.2, the generative procedure is near to the optimal classifier if the underlying models are accurate. However, it may perform poorly if this assumption is not fulfilled. With respect to this issue, we have to be conscious about the number of prerequisites in the HMM approach:

1. The Markov property (Equation (2.38)) assumes the statistical independence of subsequent state transitions.
2. Equation (2.42) assumes the statistical independence of subsequent PDFs.
3. A specification of the HMM topology is usually not part of the HMM training and mostly specified by a manual adjustment. A data-driven approach like the one outlined in Section 4.3.2.2 is used seldomly.
4. The chosen PDF function class assumes a particular shape for the true PDF, for instance, Gaussian or Gaussian mixture models.
5. The training set $\mathbb{X}$ has to be representative and sufficiently large for a reliable parameter estimation of the PDFs, which are often rather complex functions.

It is unlikely that in real world problems all of these prerequisites meet the true situations. In this respect, discriminative classifiers that do not aim to estimate class conditional densities but directly address the discrimination by creating class boundaries may be less dependent on modeling assumptions. It was already noted in Section 2.2 that SVMs belong to this category of classifiers. In this respect, they will constitute the center of interest in the remainder of this chapter.

In many disciplines, SVM classifiers have been applied to the classification task very shortly after the publication of fast training algorithms [Boser et al., 1992]. However, for quite a long period they were not observed in online HWR and domains like speech recognition and bioinformatics. A reason for this missing observance can be seen in the data structure that is typical in these application fields: common SVM techniques were originally developed for vector data of a fixed dimension and mutually independent dimensions, whereas the mentioned areas have sequence data that typically vary in length and are typically nonlinearly temporally distorted. An ad-hoc solution to overcome this incompatibility — like a linear scaling of sequences to a fixed number of samples — does not seem to be promising to outperform standard HMM techniques, since it cannot deal with the nonlinear, temporal variations in the data. In this respect, a successful approach should both embed the discriminative power of SVMs as well as the flexibility of elastic matching techniques.

An elegant approach for this goal may address the kernel concept of SVMs by the constitution of a sequence compatible kernel. In fact, this procedure has been employed by the method that will be described in this chapter.

7.2 Literature review

The present approach [Bahlmann et al., 2002] has been the first one combining SVMs and an elastic matching (or nonlinear alignment) based modeling in the context of online HWR. However, other research areas like speech recognition or bioinformatics have similar representations of data, and also there much progress has been achieved in the very recent years. It remains the hope that these areas will further stimulate each other in the future. A concise literature review shall be given in the following.

Watkins [2000] overviews a number of methods, with help of which structured objects like sequences, trees or sentences can be integrated into a general kernel formulation. First, he shows universal methods that explicitly map structured objects to vectors. Second, he argues that under certain conditions a *joint probability distribution* is a proper SVM kernel. He puts this issue in concrete terms with the joint probability of two sequences according to a *pair HMM*.

Jaakkola et al. [1999] establish a sequence SVM kernel in their application of protein homology detection and refer to it as *Fisher kernel*. The Fisher kernel method combines a generative sequence model such as an HMM with an SVM classifier, using a two-step strategy both in training and classification. During training, HMM parameters $\boldsymbol{\theta}_\lambda$ are learned using usual training techniques in a first step. The second step installs an SVM on top of the HMM. In the SVM, not the sequence pattern $\boldsymbol{x}$ itself builds the SVM domain, but a transformation of it. This transformation is defined by the so-called *Fisher score vector* $U_{\boldsymbol{x}} = \nabla_{\boldsymbol{\theta}_\lambda} \ln p\left(\boldsymbol{x}|\lambda\right)$. The operator $\nabla_{\boldsymbol{\theta}_\lambda}$ denotes the partial derivatives with respect to the HMM parameters $\boldsymbol{\theta}_\lambda$. The log-likelihood $p\left(\boldsymbol{x}|\lambda\right)$ is simply evaluated as the output of the HMM. Contrary to $\boldsymbol{x}$, the Fisher score vector $U_{\boldsymbol{x}}$ is an element in a vector space of fixed dimension and can be combined with common SVM kernels such as those defined by Equations (2.20)–(2.22).

Also in the speech recognition community the idea of sequence compatible SVM kernels has become important in the very recent years. Smith and Gales [2002] transfer the Fisher kernel method to their isolated letter utterances recognizer. They additionally studied a refinement of the Fisher score based on a "likelihood ratio".

A different strategy, also in the field of speech recognition, is pursued by Ganapathiraju et al. [2002] and co-workers [Hamaker et al., 2002]. Their approach employs a hybrid philosophy, too, and uses an explicit two-stage approach with HMMs and SVMs. Each HMM state is associated with both a generative PDF *and* an SVM. Training as well as classification are divided into two stages. In the first stage, a Viterbi search aligns HMM states with the test sequence. In the second stage, the SVMs comes into the game. During training they are learned according to the Viterbi alignments, during classification they perform a post-processing and re-score the Viterbi alignment. The SVMs are specifically modified in the sense that their output mapping provides a-posteriori probabilities instead of the usual score of Equation (2.19). IBM studies a similar technique [Fine et al., 2002].

Chakrabartty and Cauwenberghs [2002] introduced the *forward decoding kernel machines (FDKM)* and apply them to the recognition of phoneme sequences in speech. An FDKM is a framework similar to HMMs, however uses an a-posteriori probability score from SVMs

for forward decoding MAP sequence estimation. For this purpose the authors developed a particular type of SVM called *GiniSVM*. GiniSVM is particularly designed to handle multi-class problems and a-posteriori probability output mappings.

The methods described so far have been mixtures of SVMs and HMMs. A deviation from this philosophy has been publicized by Shimodaira et al. [2001a,b], again in the domain of speech recognition. The authors call their approach *dynamic time alignment kernel SVM (DTAK-SVM)*. The DTAK is defined as an optimally nonlinearly aligned sum of vector kernel evaluations $K(\mathbf{x}_i, \mathbf{y}_j)$, each of it computed from the sequences' elements: $K_{\text{DTAK}}(\boldsymbol{x}, \boldsymbol{y}) = \max_{\Phi} \sum_n K\left(\mathbf{x}_{\phi_x(n)}, \mathbf{y}_{\phi_x(n)}\right)$. The maximization is algorithmically solved with dynamic programming.

7.3 The Gaussian DTW kernel

The literature review has shown that presently a number of researches from a variety of application domains address SVM techniques for sequence classification. Their approaches use highly imaginative methods in order to establish a sequence compatible SVM framework. They have achieved excellent results in their respective application. However, most part is still founded on an HMM and thus on the estimation of its generative parameters. Thus, one principle, major drawback is inherited from the generative classification paradigm: the approaches are sensitive to the generative modeling assumptions and might perform poorly if the underlying models are not accurate. Another difficulty with these approaches can be the presence of a double training process, one for the HMM and one for the SVM. In this respect, the practical handling might be awkward as an extensive parameter tuning and a larger set of training material is required.

The approach presented in this section will be shown to be less complex and presumes less model knowledge. Its aim can rather be categorized as a direct extension of the SVM from vector space data to sequence data. It is founded on the assumption of a specific dissimilarity relation of the sequence patterns.

As indicated in the introduction, when dealing with sequential online handwriting data the basic SVM framework given by Equations (2.19)–(2.24) cannot simply be employed. Different sequences $\boldsymbol{x}^{(i)}$, $i = 1, \ldots, M$ cannot be embedded in the same vector space in general, as the necessary dimensions differ. However, an important property of the fundamental SVM Equations (2.19), (2.23) and (2.24) is that the vectors $\mathbf{x}^{(i)}$ and $\mathbf{t}$ appear only in the context of kernel evaluations. Thus our objective, when adopting SVMs to sequential handwriting data, can be to state a kernel definition that is suitable to the particular properties of sequence data.

Consider the Gaussian kernel of Equation (2.20). It is founded on the Euclidean metric in the underlying vector space. The Euclidean metric treats each dimension in the Euclidean vector space independently. Contrary, when dealing with (typically temporally distorted) sequence data, there are usually high correlations among neighboring sequence elements. A distance measure that captures these correlations has already been introduced. It is the DTW distance $D^*[d](\boldsymbol{x}, \boldsymbol{y})$ of Equation (2.32). For two equally sized sequences and a linear align-

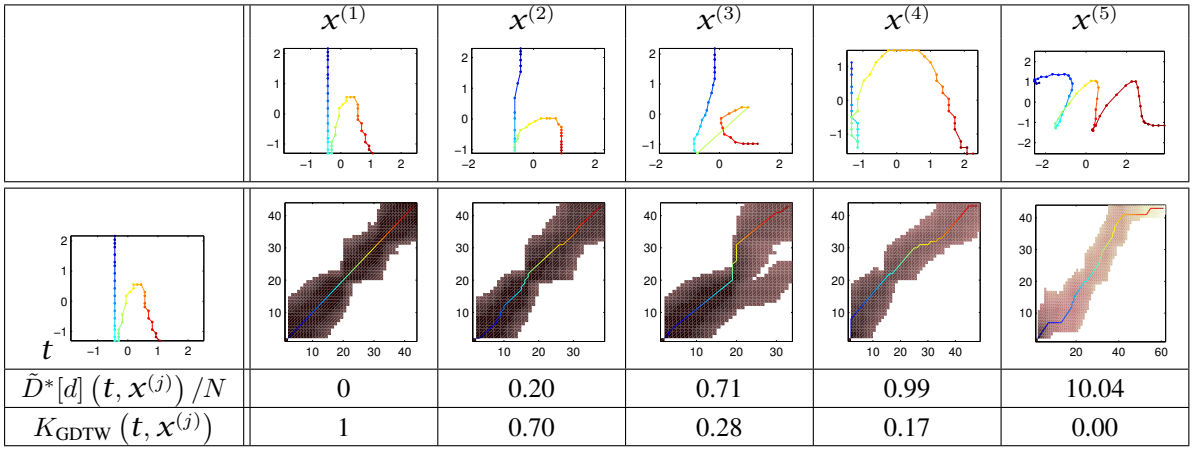


FIGURE 7.1: The Gaussian dynamic time warping (GDTW) kernel: Character patterns t (class "h") and $x^{(j)}$, $j = 1, \ldots, 5$ (character classes "h", "h", "k", "n", "m") are illustrated by the features $\tilde{x}$ and $\tilde{y}$ (see Section 3.5 for the definition). The values of the DTW distance $\tilde{D}^*[d]/N$ and the GDTW kernel evaluation K_{GDTW} for $\gamma = 1.8$ are provided in the third and fourth row, respectively. The values show an obvious but important fact: similar patterns give small values for $\tilde{D}^*[d]/N$ and large for K_{GDTW}. In the second row the Viterbi path $\mathbf{\Phi}^*$ (together with the beam search space) is illustrated: The sketched line traverses all aligned point pairs $\mathbf{\Phi}^* = \left[\left(\phi_t^*(1), \phi_{x^{(j)}}^*(1)\right), \ldots, \left(\phi_t^*(N), \phi_{x^{(j)}}^*(N)\right)\right]$ in the $N_t \times N_{x^{(j)}}$ Viterbi matrix.

ment $\phi(n) = (n, n)$ even the equation $D^*[d]\left(x, y\right) = \|\mathbf{x} - \mathbf{y}\|^2$ holds (with $\mathbf{x}$ and $\mathbf{y}$ a simple concatenation of x's and y's elements, respectively).

In this respect, an intuitive modification of Equation (2.20) is to substitute the squared Euclidean distance $\|\mathbf{x} - \mathbf{y}\|^2$ with $D^*[d]\left(x, y\right)$ (or $\tilde{D}^*[d]\left(x, y\right)$ for some practical reasons). Note that the squaring is already included in the summands of the alignment distance (cf. Equation 2.31). Due to experimental arguments we again use the normalized DTW distance $\tilde{D}^*[d]\left(x, y\right)$.

With this intuition the *Gaussian DTW (GDTW) kernel* for sequential data shall be defined by

$$K_{\text{GDTW}}\left(x, y\right) = \exp\left(-\gamma \tilde{D}^*[d]\left(x, y\right)\right). \tag{7.1}$$

Figure 7.1 illustrates the GDTW kernel in the view of some examples. The DTW distance $\tilde{D}^*[d]\left(t, x^{(j)}\right)$ and the GDTW kernel evaluation $K_{\text{GDTW}}\left(t, x^{(j)}\right)$ of a character pattern of class "h" and five different $x^{(j)}$ are shown. The reader can verify that the GDTW kernel evaluation $K_{\text{GDTW}}\left(t, x^{(j)}\right)$ can be identified with a similarity score that varies from 0 to 1.

7.4 SVM-GDTW character classification and training

With the establishing of K_{GDTW} the formulation of classification and training in an SVM context is straightforward. As in the SVM framework patterns only occur as arguments of kernel evaluations, the SVM framework for sequence data is completely defined by Equations (2.19),

(2.23), (2.24) and (7.1). In the following a two-class SVM classifier with a GDTW kernel will be abbreviated with the acronym SVM-GDTW.

For the multi-class classification it is advantageous to choose the DAG-SVM method (cf. Section 2.2.3.2). This approach has the benefit of a lower complexity in training and classification over the residuary multi-class methods [Platt et al., 2000, Hsu and Lin, 2001]. This is particularly relevant, as a GDTW kernel evaluation is rather demanding in computation time. Additionally, for the DAG-SVM approach a theoretical statement about the bound of the generalization error exists [Platt et al., 2000]. In the remainder, a DAG-SVM multi-class classifier with a GDTW kernel will be abbreviated with the acronym DAG-SVM-GDTW.

Figure 7.2 gives a graphical interpretation of a DAG-SVM-GDTW classification of a character of class "h".

7.5 Validity of the GDTW kernel

It was mentioned earlier in Section 2.2.3.1, that there are some constraints on an "allowable" kernel function that can be plugged into Equations (2.19), (2.23), (2.24). Formally, only *positive definite* functions can be used as SVM kernels [Schölkopf and Smola, 2002].

The introduction of the GDTW kernel in Section 7.3 originated from a rather intuitive argumentation than from a mathematical derivation. An investigation of its mathematical properties remains to be studied. In fact, the following deliberations will argue that the GDTW kernel does not fulfill the strict mathematical constraints in general.

The practical consequences in using positive indefinite kernels in SVMs have not been studied in detail so far. A few things, however, may be said about this issue. Theorems about the existence of a global solution of Equations (2.23) and (2.24) and the convergence of the iterative training algorithms are based on the prerequisite that the kernel used is positive definite. In this respect, it should be expected that a positive indefinite kernel negatively effects these issues. Though, from a practical point of view the extent of these effects depends on the underlying data and positive indefinite kernels can produce remarkable results like in the present case and others [DeCoste and Schölkopf, 2002, Haasdonk and Keysers, 2002, Shimodaira et al., 2001b, Haasdonk and Bahlmann, 2004, Haasdonk, 2005].

Both a theoretical and a practical study follow here.

7.5.1 Theoretical study

It has been mentioned above that a valid SVM kernel has to be positive definite. It shall be started with the definition of a positive definite function.

Definition 7.1. *A symmetric function $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ with $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ is positive definite if $K\left(\boldsymbol{x}, \boldsymbol{y}\right)$ corresponds to an inner product in a Hilbert space $\mathcal{H}$, i.e., $K\left(\boldsymbol{x}, \boldsymbol{y}\right) = \left\langle \varphi\left(\boldsymbol{x}\right), \varphi\left(\boldsymbol{y}\right)\right\rangle$ and $\varphi : \mathcal{X} \mapsto \mathcal{H}$. This property is equivalent to the statement, that for all M and all choices of $\mathbb{X} = \left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(M)}\right\} \subset \mathcal{X}$ the* Gram matrix *(or kernel matrix) $\mathbf{K} = \left[K_{ij}\right]_{M \times M}$, which is defined on the mutual kernel evaluations $K_{ij} = K\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right), i, j =$*

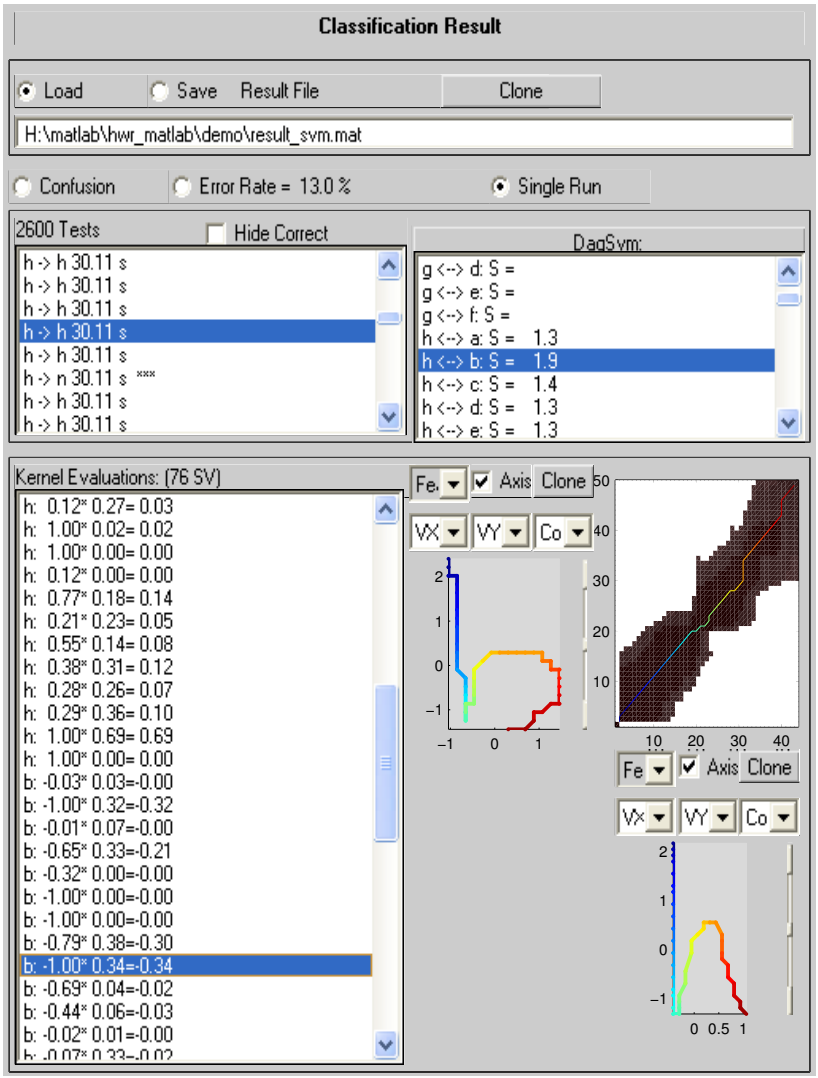


FIGURE 7.2: A snapshot from the multi-class DAG-SVM-GDTW classification GUI: it shows the classification of a test pattern t (of class "h"; selected in the upper left list). $K \cdot (K-1)/2$ two-class SVMs were trained with the DAG-SVM algorithm [Platt et al., 2000]. For illustration purposes the two-class SVM for the class pair (h $\leftrightarrow$ b) is selected in the upper right list. The score $f(t)$ of this SVM is 1.9, hence t is correctly classified as the positive class "h". In the lower part of the figure the terms $S_i w_i K\left(t, x^{(i)}\right)$ for $i = 1, \ldots, M_S$ are listed on the left, each of which is the contribution of a support vector to the classification criterion of Equation (2.19). E.g., for the selected support vector (of class "b") $S_i = -1$, $w_i = 1.00$ and $K\left(t, x^{(i)}\right) = 0.34$. To the right a graphical presentation of the selected support vector $x^{(i)}$, the test pattern t and the Viterbi matrix and path is illustrated.

$1, \ldots, M$ *is positive semi-definite. That is, all eigenvalues with respect to* $\mathbf{K}$ *are non-negative, i.e.,* $\lambda_i \geq 0$, $i = 1, \ldots, M$.

In the following the proof will be given that $K_{\text{GDTW}}\left(\boldsymbol{x}, \boldsymbol{y}\right)$ is *not* positive definite, in general. An easy procedure would be to give a simple counterexample. However, it might be more instructive to justify the missing positive definiteness by fundamental characteristics of $K_{\text{GDTW}}\left(\boldsymbol{x}, \boldsymbol{y}\right)$. In this respect, the goal is to disprove K_{GDTW}'s positive definiteness by a detour over the DTW distance $\tilde{D}^*[d]$, on which the GDTW kernel is based. First, we start with a review of two definitions and important theorems, taken from the book of Schölkopf and Smola [2002].

Definition 7.2. *A symmetric matrix* $\mathbf{K} = [K_{ij}]_{M \times M}$ *(*$M \geq 2$*) taking values in* $\mathbb{R}$ *and satisfying*

$$\sum_{i,j=1}^{M} c_i c_j K_{ij} > 0 \quad \textit{for all } c_i \in \mathbb{R}, \textit{ with} \sum_{i=1}^{M} c_i = 0 \tag{7.2}$$

is called conditionally positive definite*.*

Definition 7.3. *A function* $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ *which for all* $M \geq 2$, $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(M)} \in \mathcal{X}$ *gives rise to a conditionally positive definite Gram matrix is called a* conditionally positive definite kernel*.*

Theorem 7.4. *If a symmetric function* $-\delta^2\left(\boldsymbol{x}, \boldsymbol{y}\right)$, $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ *is conditionally positive definite and* $-\delta^2\left(\boldsymbol{x}, \boldsymbol{x}\right) = 0$ *for all* $\boldsymbol{x} \in \mathcal{X}$*, then there exists a Hilbert space* $\mathcal{H}$ *and a mapping* $\varphi : \mathcal{X} \mapsto \mathcal{H}$ *such that*

$$\left\|\varphi\left(\boldsymbol{x}\right) - \varphi\left(\boldsymbol{y}\right)\right\|^2 = \delta^2\left(\boldsymbol{x}, \boldsymbol{y}\right). \tag{7.3}$$

This implies that $\delta\left(\boldsymbol{x}, \boldsymbol{y}\right)$ *is a semi-metric.*

Proof. See [Schölkopf and Smola, 2002, Proposition 2.24]

□

Theorem 7.5. *A symmetric function* $-\delta^2\left(\boldsymbol{x}, \boldsymbol{y}\right)$ *is conditionally positive definite if and only if* $K\left(\boldsymbol{x}, \boldsymbol{y}\right) = \exp\left(-\gamma\delta^2\left(\boldsymbol{x}, \boldsymbol{y}\right)\right)$ *is positive definite for all* $\gamma > 0$*.*

Proof. See [Schölkopf and Smola, 2002, Proposition 2.28].

□

From Theorems 7.4 and 7.5 the following corollary can be derived.

Corollary 7.6. *If* $\delta\left(\boldsymbol{x}, \boldsymbol{y}\right)$, $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ *with* $\delta\left(\boldsymbol{x}, \boldsymbol{x}\right) = 0$, $\forall \boldsymbol{x} \in \mathcal{X}$ *is not a semi-metric, then* $K\left(\boldsymbol{x}, \boldsymbol{y}\right) = \exp\left(-\gamma\delta^2\left(\boldsymbol{x}, \boldsymbol{y}\right)\right)$ *is not a positive definite kernel.*

Proof. As $\delta\left(\boldsymbol{x}, \boldsymbol{y}\right)$ is not a semi-metric by assumption, $-\delta^2\left(\boldsymbol{x}, \boldsymbol{y}\right)$ is not conditionally positive definite, according to Theorem 7.4. From Theorem 7.5 it follows that $\exp\left(-\gamma\delta^2\left(\boldsymbol{x}, \boldsymbol{y}\right)\right)$ is not positive definite for at least one γ.

□

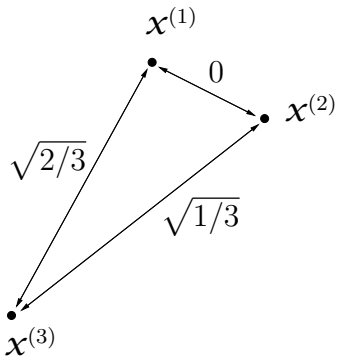


FIGURE 7.3: A situation where the triangle inequality of $\sqrt{\tilde{D}^*[d]}$ is violated.

Before we come to the point where we disprove $K_{\text{GDTW}}\left(x, y\right)$ to be positive definite, a statement about the metric properties of the square root of the DTW distance, $\sqrt{\tilde{D}^*[d]}$, shall be derived.

Proposition 7.7. *The square root of the DTW distance, $\sqrt{\tilde{D}^*[d]}$, is not a semi-metric as it violates the triangle inequality, i.e., patterns $x^{(1)}$, $x^{(2)}$ and $x^{(3)} \in \mathcal{X}$ exist for which*

$$\sqrt{\tilde{D}^*[d]\left(x^{(1)}, x^{(2)}\right)} + \sqrt{\tilde{D}^*[d]\left(x^{(2)}, x^{(3)}\right)} < \sqrt{\tilde{D}^*[d]\left(x^{(1)}, x^{(3)}\right)}. \tag{7.4}$$

Proof. The aim of this proof is to construct an example that violates the triangle inequality. A situation that is very sensitive to a violation is depicted in Figure 7.3. The figure shows hypothetical distance relations of the three patterns $x^{(1)}$, $x^{(2)}$ and $x^{(3)}$. In these, two different patterns $x^{(1)}$ and $x^{(2)}$ have zero distance, i.e.,

$$\sqrt{\tilde{D}^*[d]\left(x^{(1)}, x^{(2)}\right)} = \sqrt{\tilde{D}^*[d]\left(x^{(2)}, x^{(1)}\right)} = 0 \text{ and } x^{(1)} \neq x^{(2)} \tag{7.5}$$

and $x^{(3)}$ has different non-zero distances to $x^{(1)}$ and $x^{(2)}$, i.e.,

$$0 \neq \sqrt{\tilde{D}^*[d]\left(x^{(1)}, x^{(3)}\right)} \neq \sqrt{\tilde{D}^*[d]\left(x^{(2)}, x^{(3)}\right)} \neq 0. \tag{7.6}$$

Under these circumstances is obvious that either

$$\sqrt{\tilde{D}^*[d]\left(x^{(1)}, x^{(2)}\right)} + \sqrt{\tilde{D}^*[d]\left(x^{(2)}, x^{(3)}\right)} < \sqrt{\tilde{D}^*[d]\left(x^{(1)}, x^{(3)}\right)} \tag{7.7}$$

or

$$\sqrt{\tilde{D}^*[d]\left(x^{(2)}, x^{(1)}\right)} + \sqrt{\tilde{D}^*[d]\left(x^{(1)}, x^{(3)}\right)} < \sqrt{\tilde{D}^*[d]\left(x^{(2)}, x^{(3)}\right)}. \tag{7.8}$$

One of those equations gives rise to a violation of the triangle inequality, even without a specific parameter choice for $x^{(1)}$, $x^{(2)}$ and $x^{(3)}$.

A specific example for a situation where Equations (7.5) and (7.6) hold is the choice $x^{(1)} = [1, 1, 2]$, $x^{(2)} = [1, 2, 2]$, $x^{(3)} = [2, 2]$.

□

We conclude with the following corollary.

Corollary 7.8. *$K_{GDTW}\left(\boldsymbol{x}, \boldsymbol{y}\right) = \exp\left(-\gamma \tilde{D}^*[d]\left(\boldsymbol{x}, \boldsymbol{y}\right)\right)$ is a positive indefinite kernel.*

Proof. From Proposition 7.7 it follows that $\sqrt{\tilde{D}^*[d]\left(\boldsymbol{x}, \boldsymbol{y}\right)}$ is not a semi-metric. If we identify $\sqrt{\tilde{D}^*[d]} = \delta$ in Corollary 7.6, it follows that $K_{\text{GDTW}}\left(\boldsymbol{x}, \boldsymbol{y}\right)$ is a positive indefinite kernel. □

For the example illustrated in the proof of Proposition 7.7 and for $\gamma = 1$ the Gram matrix is $\mathbf{K}_{\text{GDTW}} = \begin{bmatrix} \exp(-0) & \exp(-2/3) & \exp(-1/3) \\ \exp(-2/3) & \exp(-0) & \exp(-0) \\ \exp(-1/3) & \exp(-0) & \exp(-0) \end{bmatrix}$ and the eigenvalue vector $\boldsymbol{\lambda}$ can numerically be evaluated to $\boldsymbol{\lambda} \approx (-0.03, 0.53, 2.51)^T$, including a negative λ_1.

7.5.2 Numerical study

The last section has argued that K_{GDTW} is positive indefinite and thus not a valid SVM kernel. As will be shown in the experiments (Section 7.8), it will nevertheless produce remarkable classification results in the context of online handwriting data. This section will study this issue from a practical point of view.

A reason for the good performance is presumely that for the particular application *typical* data behaves well with the function K_{GDTW}. The Gram matrix $\mathbf{K}_{\text{GDTW}}$ which is computed from the observed data might be good-natured with respect to the issue of positive semi-definiteness, although the kernel K_{GDTW} in general is not. A numerical study of $\mathbf{K}_{\text{GDTW}}$, computed from *UNIPEN* handwriting data, can enlighten deliberations about this question.

Therefore, experiments were conducted where Gram matrices were generated by randomly chosen characters $\boldsymbol{x}^{(i)}$, $i = 1, \ldots, M$ from the *UNIPEN* HWR database. The results of these simulations were as follows:

1. For sample sets of sizes $M \leq 40$ *all* λ_i were experimentally measured to be non-negative in *all* simulations, that is, the Gram matrices are positive semidefinite.

2. Larger sample sets $M > 40$ violate the positive definiteness only weakly: The missing positive definiteness was due to only a few negative eigenvalues with small absolute values compared to the other eigenvalues.

   Figure 7.4 shows a (sorted) eigenvalue vector $\boldsymbol{\lambda}$ as a bar plot. It has been computed from a training set including examples of an “b” and ”h” with $M = 382$.

To conclude this section, it has been shown that K_{GDTW} is not a valid SVM kernel in a strict mathematical sense. However, for typical data in the present application the violation of the positive definiteness by K_{GDTW} is limited to some extend. This aspect shall be the justification for the use of the SVM-GDTW classifier for HWR.

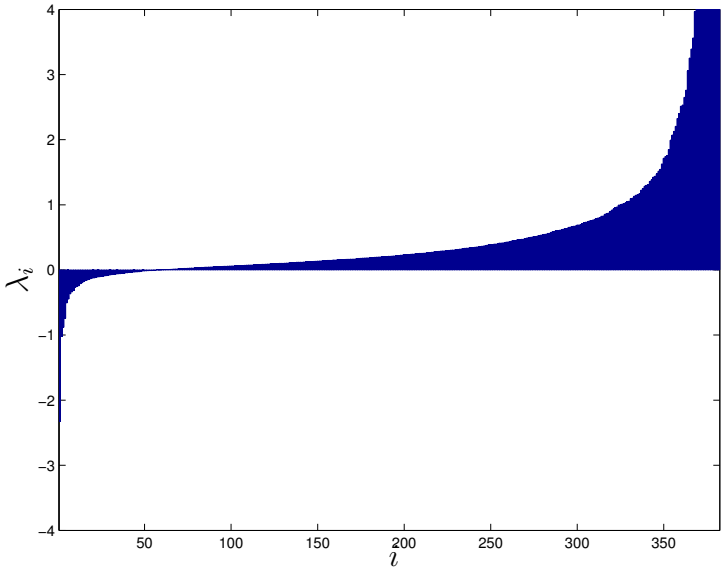


FIGURE 7.4: A (sorted) eigenvalue vector $\boldsymbol{\lambda}$, numerically computed for a randomly chosen 382×382 Gram matrix $\mathbf{K}_{\text{GDTW}}$ of kernel evaluations from examples of the classes "b" and "h". The minimal eigenvalue is $\lambda_1 \approx -2.3$, the maximal $\lambda_M \approx 87$. Note that only a part of the whole scale on the ordinate is shown. 57 eigenvalues are negative, corresponding to a fraction of $\approx 15\,\%$. The absolute values of these are relatively small with respect to the positive eigenvalues. In this example $\frac{\sum_{i,\lambda_i<0}|\lambda_i|}{\sum_{i,\lambda_i>0}|\lambda_i|} \approx 0.03$.

7.6 Analysis of complexity

The following deliberations shall analyze the complexity of the character SVM-GDTW modeling.

A kernel evaluation (Equation (7.1)) for a typical character pair asymptotically takes

$$\mathcal{C}_{\text{Time}}\,(\text{Kernel}) = \mathcal{O}(\tilde{N} \cdot F \cdot |\mathbb{P}|) \tag{7.9}$$

operations in a beam search environment. Experimentally the absolute computation time was measured as

$$\text{Time}\,(\text{Kernel}) \approx 0.0005\,\text{sec} \tag{7.10}$$

with $\tilde{N} = 42$ the average length of the sequences and $F = 3$ the dimension of $\mathbf{x}_i$ in a *C++* implementation on an AMD Athlon 1600MHz.

The asymptotic training time of the two-class SMO and L-class DAG-SVM training algorithm is

$$\mathcal{C}_{\text{Time}}\,(\text{Train,2-class}) = \mathcal{O}\left((M^{\text{tot}})^{\gamma}\right) \tag{7.11}$$

and

$$\mathcal{C}_{\text{Time}}\,(\text{Train,}L\text{-class}) = \mathcal{O}\left(2^{\gamma-1}L^{2-\gamma}\,(M^{\text{tot}})^{\gamma}\right), \tag{7.12}$$

respectively, with $\gamma \approx 2$ and M^{tot} the total number of training examples [Platt et al., 2000]. In a typical multi-class lower case character experiment (the 20 %/20 % training/test set partitioning, cf. Section 7.8), $M^{\text{tot}} = 12.000$ and the absolute training time was measured as

$$\text{Time (Train,2-class)} \approx 0.25\,\text{h} \tag{7.13}$$

and

$$\text{Time (Train,26-class)} \approx 81\,\text{h}. \tag{7.14}$$

Again, training can easily be parallelized into $L \cdot (L-1)/2$ independent processes.

The average number of support vectors in the multi-class lower case character experiments mentioned was $\tilde{M}_S \approx 170$. The classification time for one two-class SVM hence is

$$\begin{aligned} \text{Time (Classify,2-class)} &\approx \tilde{M}_S \cdot \text{Time (Kernel)} \\ &= 170 \cdot 0.0005\,\text{sec} = 0.085\,\text{sec}. \end{aligned} \tag{7.15}$$

Since the DAG-SVM evaluates $L-1$ two-class SVMs, a classification of a lower case character takes

$$\begin{aligned} \text{Time (Classify,26-class)} &\approx (L-1) \cdot \text{Time (Classify,2-class)} \\ &= 25 \cdot 0.085\,\text{sec} = 2.125\,\text{sec}. \end{aligned} \tag{7.16}$$

The memory complexity mainly consists of the storage of all support vectors, thus

$$\begin{aligned} \text{Memory(26-class)} &= L \cdot (L-1)/2 \cdot \tilde{M}_S \cdot \tilde{N} \cdot F \cdot 4\,\text{Byte} \\ &= 26 \cdot 25/2 \cdot 170 \cdot 42 \cdot 3 \cdot 4\,\text{Byte} \approx 28\,\text{Mbyte}. \end{aligned} \tag{7.17}$$

It should be noted that the above derived memory complexity corresponds to the worst case. The derivation assumes that each support vector in the $L \cdot (L-1)/2$ two-class SVMs is independently stored. In practical applications it can be expected that support vectors are shared among a number of different two-class SVMs. Then, a smart storage organization can reduce the memory requirements.

Of course, both time and memory complexity are immense and not practical for an operation of the DAG-SVM-GDTW classifier on handheld devices. For a serious use in real world applications a number of further optimization procedures have to be employed. This issue will be depicted in Section 9.2.

7.7 SVM-GDTW word classification — an outlook

An important question in the context of SVM based HWR concerns the generalization of the character to a word classification.

Section 4.5 has described how this problem is effectively solved for generative modeling approaches, for instance, HMM and CSDTW. There, the probabilistic formulation of the character appearance model enables an elegant integration of the character recognition within a

probabilistically reasoned language model. Also, the idea of decomposing the generative word model into a sequence of generative character models implicitly solves the segmentation problem. It has been sketched how the word segmentation is obtained as a globally optimal by-product from the word classification process. In its pure form, that is, without beam search, no commitment to an explicit segmentation decision is required before the entire classification is completed.

For SVMs, suitable word classification techniques are currently not so highly developed. Main difficulties arise from the following: First, the output of SVMs does generally not represent a probability value, and hence, an embedding within a probabilistic language model is not straightforward. Although additions to binary SVMs have been proposed, which assign probabilities posterior to the SVM training [Platt, 2000], a transfer of this concept to the multi-class case has not been thoroughly studied so far. Second, it is not clear nowadays how the segmentation problem can be solved with SVMs.

As an outlook, two interesting directions for SVM word recognition shall be pointed out:[1]

A pure discriminative approach learns discriminative decision boundaries between all K word classes. Depending on the multi-class strategy used (cf. Section 2.2.3.2), this approach would comprise K or $K \cdot (K-1)/2$ two-class SVMs, respectively, each of which based on a set of "support words", for a K-word dictionary. In the SVM intuition, those "support words" would correspond to instances of the difficult examples for the respective two-class problem.

As a direct implementation of this strategy is intractable due to its high computational complexity (for a reasonable K) and the requirement of a huge training set (it would require at least hundreds of training samples for each of the K word classes), the challenge is here to decompose the "support-words" and corresponding kernel evaluations into smaller units, for instance, into "support-characters". In fact, this issue is also related to a solution of the above mentioned segmentation problem.

Generative-discriminative hybrids follow a two-step philosophy, generalizing the ideas already mentioned in Section 7.1 [Ganapathiraju et al., 2002, Hamaker et al., 2002, Fine et al., 2002, Chakrabartty and Cauwenberghs, 2002]. Their principle can be summarized as follows: First, a generative model provides a top-N word candidate list, augmented with the respective segmentation hypothesis. Second, a character SVM is applied to re-score each of the word candidates for a final decision, based on the segmentations provided.

While the latter line of research is not as elegant as the pure discriminative approach, a practical realization is easier to achieve with the presently available technology. Contrary, no current work pursuing the first philosophy is known to the author, indicating that this strategy is truly a challenging task. However, a successful realization would result in an elegant and presumely powerful discriminative handwriting classification.

[1] Ideas partially originate from a discussion with Hiroshi Shimodaira in Niagara-on-the-Lake, August 2002.

	Character pairs	Training set size M^{tot}	# SV M_S	Error rate $E_{\text{SVM-GDTW}}$	Error rate E_{CSDTW}
dissimilar	"a" ↔ "b"	3540	298	**0.5 %**	0.8 %
	"d" ↔ "m"	2595	334	**0.1 %**	0.4 %
similar	"c" ↔ "e"	5088	351	**3.7 %**	7.2 %
	"u" ↔ "v"	2214	397	9.2 %	**6.8 %**
	"y" ↔ "g"	2088	358	11.2 %	**7.7 %**
	"b" ↔ "h"	2524	275	**2.3 %**	3.2 %

TABLE 7.1: Two-class experiments on *UNIPEN* data: error rate $E_{\text{SVM-GDTW}}$ of the SVM Gaussian DTW kernel approach for examples of *dissimilar* ("a" ↔ "b", "d" ↔ "m") and *similar* lower case character pairs ("c" ↔ "e", "u" ↔ "v", "y" ↔ "g", "b" ↔ "h"). The number of training samples (M), support vectors (M_S) and the error rate E_{CSDTW} for CSDTW are listed as well. The training set size was $67\,\%$, the test set size $33\,\%$ of the *UNIPEN* Train-R01/V07 database. The lowest error rate is typed bold face.

In conclusion, the advance of an efficient and accurate SVM word classification will be a challenging and exciting research area for the upcoming years.

7.8 Experiments

Like in the previous chapters, systematic experiments on the character recognition have been applied to *UNIPEN* data. The SVM-GDTW classifiers were trained with the *sequential minimal optimization (SMO)* algorithm [Platt, 1999] using a modification of a third party *Matlab* SVM toolbox [Cawley, 2000]. For the following experiments the SVM and kernel parameters were set to $C = 1$ and $\gamma = 1.8$, respectively, which has been a result of a careful, grid search based parameter tuning.

7.8.1 Two-class experiments

In a first study the concern was to detect whether an SVM-GDTW is able to classify clearly separable data. Hence, SVM-GDTW classifiers were applied to character class pairs which were shown to be dissimilar with respect to the CSDTW dissimilarity measure of Chapter 6 and achieved very low classification confusions with the CSDTW classification (cf. Chapter 4).

Like it has been described in Section 4.6, the characters were randomly and disjointly divided into training and test sets of a ratio $2:1$. Table 7.1 summarizes classification error rates and compares them to the results of CSDTW. The table shows the satisfying result, that for the dissimilar character pairs ("a" ↔ "b", "d" ↔ "m") classification errors are rare in both classification methods, actually are due to mislabelings.

In a second category of two-class experiments the discrimination of character class pairs,

UNIPEN section	Approach	Error rate $\tilde{E}_{\text{DAG-SVM-GDTW}}$ / $\tilde{E}_{\text{CSDTW}}$	*UNIPEN* Train-R01/V07 training / test set partitioning
1a (digits)	DAG-SVM-GDTW	**4.0 %** 3.8 %	20 %/20 % 40 %/40 %
	CSDTW	4.5 % **3.2 %**	20 %/20 % 40 %/40 %
1b (upper case)	DAG-SVM-GDTW	**7.6 %** **7.6 %**	20 %/20 % 40 %/40 %
	CSDTW	10.0 % 8.0 %	20 %/20 % 40 %/40 %
1c (lower case)	DAG-SVM-GDTW	**11.7 %** 12.1 %	10 %/10 % 20 %/20 %
	CSDTW	13.0 % **11.4 %**	10 %/10 % 20 %/20 %

TABLE 7.2: Multi-class experiments on *UNIPEN* sections 1a/b/c (indicated in the first column). The second column denotes the classification approach used, notably DAG-SVM-GDTW and CSDTW (cf. Chapter 4). The third column shows the mean error rate $\tilde{E}_{\text{DAG-SVM-GDTW}}$ and $\tilde{E}_{\text{CSDTW}}$ of five different dataset partitionings which were made with respect to the sample set sizes indicated in the last column. The lowest mean error rate is typed bold face.

which were shown to be *similar* and were frequently misclassified by CSDTW, was examined. Table 7.1 illustrates, that both approaches achieve comparable classification results. For some of the selected character pairs ("c" ↔ "e", "b" ↔ "h") SVM-GDTW gives lower error rates than SDTW, for others ("u" ↔ "v", "y" ↔ "g") vice versa.

Further, it can be taken from the table that the set of support vectors comprises roughly 10 % of the training set, in average $\tilde{M}_S \approx 300$–350.

7.8.2 Multi-class experiments

For a multi-class study experiments were carried out on two different *UNIPEN* dataset sizes in order to give an idea of the recognizer's dependence on this quantity.

Table 7.2 summarizes classification error rates of a DAG-SVM-GDTW classifier for *UNIPEN* sections 1a/b/c (digits/upper/lower case characters, respectively).

From the values it can be seen that DAG-GDTW-SVM achieves lower error rates than CSDTW for the relative small training set. For the larger training sets DAG-GDTW-SVM and CSDTW achieve comparable error rates.

The higher performance of DAG-GDTW-SVM in comparison with CSDTW on the small training set supports the statement that the generative CSDTW approach is highly dependent on accurate models, which cannot be satisfactorily gained from the relative small training set.

For a comparison of the DAG-GDTW-SVM performance in a broader context, the reader is referred to Table 4.1. From there it can be taken that DAG-SVM-GDTW classification outperforms other approaches (especially in the important lower case characters section), though a smaller training set has been used. However, also at this point the issues about different datasets, dataset partitionings, etc., count, as it has been argued in Section 4.6, and one should be careful when interpreting the results on different datasets.

7.8.3 Visual study of support vectors

Like in Section 4.6.3 we can again profit from the geometrically interpretable feature extraction. This time, it is instructive to have a closer look at a training result and graphically display selected support vectors. Further, a comparison with non-support vectors can give a visual feedback of an SVM-GDTW classifier. This issue is pursued in Figure 7.5. The basis of the illustration is a two-class SVM-GDTW that discriminates between the classes "h" and "b". The figure shows six selected training patterns for each class, represented by the $\tilde{x}$-$\tilde{y}$ feature representation. Three of the examples are support vectors (the respective top row), three are non-support vectors (the respective bottom row). The weights w_i are given beneath their illustration.

The patterns illustrated in Figure 7.5 represent only a very small fraction of all training patterns. In this respect it is difficult to conclude general interpretations from these pictures. However, in correspondence with a careful inspection of further patterns (not included in the figure), a number of statements can be made. Although, it should be mentioned that exceptions exist.

1. Most support vectors are

    a) patterns that are close to the respective competitive class in the feature representation (i.e., "h" patterns that are similar to an "b", "h" patterns that are similar to an "b"). Examples for these cases are the patterns (b), (c), and (h).

    b) patterns the shape of which occurs relatively sparely in the training set, e.g. patterns (a), (g) and (i).

2. Most non-support vector are

    a) characterized in that they can be unambiguously identified with their respective class ((d)–(f) and (j)–(l)) and

    b) rather often present in a similar feature representation in the training set (also (d)–(f) and (j)–(l)).

In fact, these observations correspond to the general idea of the discriminative classification paradigm and SVMs.

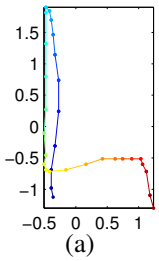
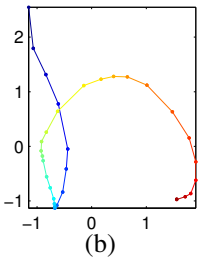
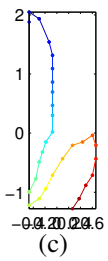
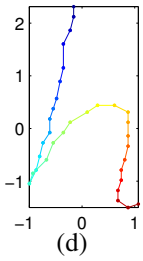
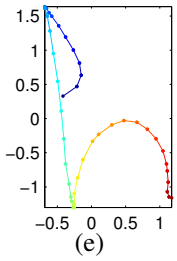
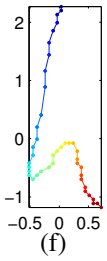
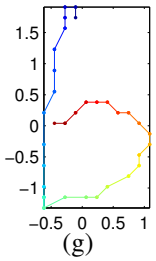
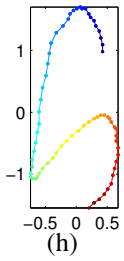
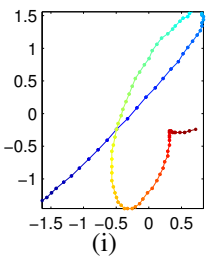
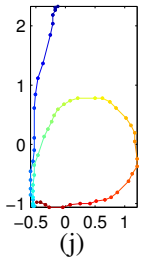
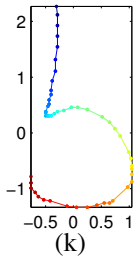
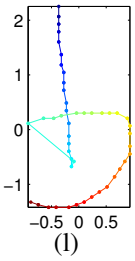
"h"	SV	 (a)	 (b)	 (c)
	Weight w_i	0.97	1.00	1.00
	non-SV	 (d)	 (e)	 (f)
	Weight w_i	0.00	0.00	0.00
"b"	SV	 (g)	 (h)	 (i)
	Weight w_i	0.69	1.00	0.32
	non-SV	 (j)	 (k)	 (l)
	Weight w_i	0.00	0.00	0.00

FIGURE 7.5: Support vectors (SV) and non-support vectors (non-SV) of a two-class SVM "h" $\leftrightarrow$ "b". The patterns are represented through the feature subspace according to the two features $\tilde{x}$ and $\tilde{y}$. Note that the tangent slope angle θ is also used in the SVM but not illustrated here.

7.9 Summary

The present chapter has presented a discriminative approach for the recognition of sequence data, especially of online handwritten characters. This technique combines dynamic time warping (DTW) and support vector machines (SVM) by integrating DTW into a Gaussian SVM kernel. The benefit of this approach over conventional HMM based techniques is the absence of restrictive assumptions about class conditional PDFs. The only essential assumption made is the selection of the kernel.

It was shown that the GDTW kernel, however, does not meet the condition of positive definiteness and thus is — in a strict mathematical sense — not a valid kernel in the context of SVM classifiers. However, it appeared that in the context of real-world online handwriting data the GDTW kernel does not have the negative effects as the violation of the positive definiteness might suggest. For a study of this issue, Gram matrices have been computed in simulations based on *UNIPEN* data. It has been shown that these violate the positive definiteness only weakly.

The classification technique has been applied to characters of the *UNIPEN* handwriting recognition database. Experiments have shown superior recognition rate in comparison to an HMM-based classifier for relative small training sets and comparable rates for larger training sets.

Though, a problem of this approach is the complexity. A classification of a character consumes ≈ 2 sec in a standard PC environment and ≈ 28 Mbyte of storage. For a use in real world applications this issue has to be further addressed in future research.

CHAPTER 8

An application: *frog on hand* on a PDA

This chapter describes an implementation of the developed HWR in a real world problem: The character recognition is realized in a PDA environment on a Linux Compaq iPAQ. As the computational demands on small devices are critical and, further, scalability of the system is desired, the CSDTW classification has been chosen in favor of the SVM-GDTW classification. Character recognition runs with high accuracy and almost in real time on the described platform.

8.1 Introduction

In order to demonstrate *frog on hand*'s performance in a typical application, the character recognition has been implemented for the use in a PDA environment. As noted in Chapter 1, the hardware of mobile devices gives particular constraints on the computational and memory complexity of the recognition implementation. Further, mobile devices are used by a large variety of users. Due to these reasons, appropriate recognition algorithms should be scalable to the hardware and robust with respect to large writing variations. With CSDTW, a classification approach exists that meets these requirements.

The remainder of this chapter describes the environment chosen for the implementation, details about the recognition algorithm and its computational and memory requirements.

8.2 Software and hardware environment

The underlying software interfaces are *Trolltech*'s *Qtopia*[1] PDA environment and its GPL licensed fork, the *Open Palmtop Integrated Environment (OPIE)*.[2] Both environments provide a complete PDA graphical user interface and a number of typical personal information management applications, for instance, date book, address book, and to-do list. They run in the embedded Linux operating system (kernel 2.4.19) from the *Familiar* distribution.[3]

The PDA implementation of the *frog on hand* character recognition integrates the recognition system, described in the previous chapters, into a plug-in that can be loaded both into *Qtopia* and *OPIE*. As known from previous character recognition implementations, the plugin's interface consists of a hidable input region. When visible, it is divided into three writing zones, each zone corresponding to one of the recognition domains "digits", "lowercase characters" and "uppercase characters". The user input is recognized and the classified character label ("0"–"9", "A"–"Z", "a"–"z") is transferred to the active application. The picture in Figure 8.1 gives an idea of the environment. A binary package of the implementation is available from the author's WWW site.[4]

The environments *Qtopia* and *OPIE* are designed for small devices. Currently supported hardware includes the PDAs *Compaq iPAQ* and *Sharp Zaurus*, the "webpad" *Siemens SimPAD* and the smart phone *Motorola A760*.

The present HWR plug-in was developed and tested on a *Compaq iPAQ 3660*. This device is equipped with a 203 MHz StrongARM processor, 16 MByte flash memory (corresponding to a PC's hard drive) and 64 MByte RAM. The StrongARM processor lacks a built-in floating point arithmetic, thus a fixed point data representation is used in the *frog on hand* PDA implementation.

8.3 Modules of the recognizer

The PDA recognizer employs pre-processing and feature selection, as explained in Chapter 3. For classification, the CSDTW classifier has been chosen in favor of the SVM-GDTW classifier, as its complexity is lower in general and, since, it is straightforwardly scalable to the device's hardware.

Each of the recognition domains (i.e., "digits", "lowercase characters" and "uppercase characters") are associated with a set $\Re = \left\{\mathcal{R}^{lk}\right\}_{l\in\{1,\ldots,L\},k\in\{1,\ldots,K_l\}}$ of CSDTW reference models, trained on the *UNIPEN* "Train-R01/V07" sections 1a/b/c. In order to obtain small reference model sets, the CSDTW training parameter configuration $D_{\max} = 7.0$ and $O_{\min} = 23$ has been selected (cf. Table 4.1).

[1] `http://www.trolltech.com/products/qtopia/`
[2] `http://opie.handhelds.org/`
[3] `http://familiar.handhelds.org/`
[4] `http://lmb.informatik.uni-freiburg.de/people/bahlmann/frog.en.html`

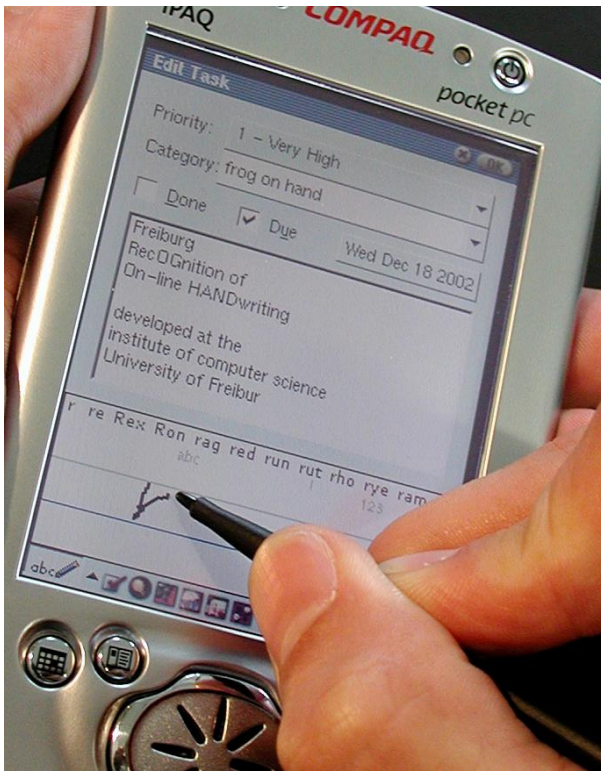


FIGURE 8.1: *frog on hand* with CSDTW classification on a Linux Compaq iPAQ.

8.4 The recognizer's footprint

The resulting classifier sizes vary in the range of 250–850 KByte, computation time in the range of 0.1–0.3 sec, respectively, as summarized by Table 8.1. The classifier fits easily on the iPAQ's flash memory. However, the recognition time exceeds the time that is commonly seen as comfortable for the user ($\approx$ 0.1 sec), by a factor of 1–3. Nevertheless, it can be expected that a further optimization of the code, in particular with respect to the underlying processor characteristics, could close this gap.

UNIPEN section	# allographs A^{tot}	Model size (uncompressed)	Recognition time
digits (1a)	27	250 KByte	0.1 sec
lowercase (1b)	67	650 KByte	0.2 sec
uppercase (1c)	117	850 KByte	0.3 sec

TABLE 8.1: Memory and computational complexity for the *UNIPEN* 1a/b/c sections on the iPAQ.

8.5 Summary

This chapter has described an implementation of the *frog on hand* character recognition in a PDA environment. Software and hardware environment are *Trolltech's Qtopia* and a *Compaq iPAQ*, respectively. The HWR is based on the CSDTW classification. In this respect, the implementation could benefit from the scalability of this approach. The implementation was shown to deal with large writing style variations and runs in near real time.

CHAPTER 9

Conclusion and perspectives

9.1 Conclusion

This thesis has discussed methods for accurate online HWR. All techniques have been integrated into the prototype *frog on hand*, which is the writer independent online HWR system at the University of Freiburg. The work is not only confined to online HWR, but also transferable to general pattern recognition techniques. In particular, it has addressed two important problems in pattern recognition:

1. a robust classification of sequence data, and
2. a unified statistical feature space modeling of a mixture of directional and linear variables.

In more detail, the following original aspects have been studied in this thesis:

Classification of sequence data with CSDTW and SVM-GDTW: With CSDTW and SVM-GDTW, two novel machine learning techniques, that are generically suited for sequence data, have been presented.

CSDTW has been introduced as a generative learning framework, which holistically integrates two powerful pattern recognition techniques in the context of sequence modeling: cluster analysis and HMM-like statistical modeling. The term "holistic" refers to one of CSDTW's strength: clustering and statistical modeling are embedded in a unique feature space and are seamlessly integrated into each other. Another quality of CSDTW is its scalability. By employing a hierarchical cluster analysis, CSDTW provides the possibility to scale the classifier, finding a compromise between the recognition accuracy and the computational requirements. The practical impact of this property has been

demonstrated with the implementation of *frog on hand* with CSDTW classification on a Linux Compaq iPAQ PDA.

Along with CSDTW, a (dis-) similarity measure for a pair of probabilistic sequence models has been introduced. It has been established with particular application to CSDTW, however the concept is transferable to HMMs with left-to-right transitions. The (dis-) similarity measure is based on the Bayes probability of error, thus it has been used to analyze classification errors. Indeed, it could be empirically verified that class pairs with a high similarity measure have frequent classification confusions.

A second innovative sequence classification has been introduced by means of the SVM-GDTW approach. Contrary to CSDTW, SVM-GDTW employs a discriminative classification philosophy, and thus performs no restrictive assumptions about generative sources for the data. Naturally, this philosophy is advantageous if no prior information about the generative source (e.g., the function class of the PDFs) or only few training data are available. SVM-GDTW combines the concepts of SVM and DTW by the formulation of the so-called Gaussian DTW (GDTW) kernel. The thesis has studied the GDTW kernel both from theoretical and numerical points of view. Theoretical studies have revealed that the GDTW kernel is generally not a valid SVM kernel in a strict mathematical sense, as it does not meet the condition of positive definiteness. Though, empirical studies of the kernel with application to typical online handwriting data have shown its practical usability. In this context, the positive definiteness has been shown to be violated only weakly and the theoretical drawbacks are mostly restricted to pathological cases. With this in mind, standard SVM methods for training and classification could be employed in the context of the GDTW kernel and, indeed, achieved excellent recognition results.

Both CSDTW and SVM-GDTW have been applied to character recognition using the standard *UNIPEN* online HWR database and benchmark. In comparison to state-of-the-art character recognizers of other research projects, both classification techniques achieved lower or similar error rates. In a direct comparison of CSDTW and SVM-GDTW, experiments have shown superior recognition rates of SVM-GDTW for relatively small training sets, while CSDTW has performed more accurate for larger training sets. Complexity wise, CSDTW resulted in a leaner classifier, while SVM-GDTW has required rather high computation time and memory.

Having a practical application — like the implementation of *frog on hand* on a PDA — in mind, it can be concluded that the CSDTW classification is a more suitable approach due to its lower complexity, its scalability, its straightforward extensibility to a word recognition, and higher recognition rates when a large set of training material is available. Nevertheless, from the background that SVM based sequence classification is a rather recent achievement and thus has a far less extensive research history and community, significant improvements can be expected from further research in this field. In addition, hybrid sequence classification techniques, based on either techniques, pro-

vide a promising direction for sequence classification, as generative and discriminative approaches employ complementary philosophies.

Due to their generality, both classification approaches can advantageously be applied to other pattern recognition problems where data are a sequence of feature vectors. Applications include speech recognition, medical applications, genome processing, robotics, etc.

Combined feature space modeling of linear and directional data: Part of this thesis has established a solution for a unified statistical Gaussian-like modeling of linear (i.e., values on the real line) and directional (i.e., values on the unit circle) features. To this end, the so-called *approximated multivariate semi-wrapped Gaussian distribution* has been introduced, which is an extension of the *wrapped Gaussian distribution*, often used in physics applications.

It has been shown that the proposed modeling significantly improves the recognition accuracy in the studied application of online HWR compared to previous standard approaches. Further benefits of this solution are savings in computation time and memory.

The approach is general in the respect that it is applicable in any pattern recognition problem (including vector space and sequence data) where both linear and directional variables occur in the feature space.

9.2 Perspectives

Some further challenges of the above summarized techniques arise from complexity issues, others from the fact that their applicability depends on the underlying data. Worthwhile approaches for improvements shall be discussed in the following. First, they shall be addressed from an individual point of view, followed by more general remarks.

CSDTW: Depending on the application context (e.g., for a real time word recognition) or the target device (e.g., for a cell phone), the CSDTW classification might still be too complex. In order to speed up classification and decrease the classifier size, a method for a sub-sampling of the allograph models would be an attractive area for further studies. The usefulness of this proceeding can be seen in the following fact: Figure 4.7 reveals that in many cases the variations in the $\tilde{x}$-$\tilde{y}$-plane are rather small towards the writing direction, compared to the perpendicular direction. This suggests that an accurate classification can also be achieved with larger variances towards the writing direction, and thus, a smaller number of CSDTW states. Similar to the current model topology selection, the sub-sampling should preferably be solved automatically and data-driven.

The presented approach has assumed a feature space with compact, Gaussian-like clusters. However, clusters in many real-world problems often have a more complex form, for instance, they can be elongated or shell-shaped. Clustering methods exist

[Zhang and Rudnicky, 2002] that cope with this issue by mapping the input space with a kernel. A future challenge is to integrate such a kernel formulation jointly into the clustering *and* the statistical, generative sequence modeling.

Recently, Vuori et al. [2001] proposed methods for an unsupervised, writer *dependent* adaptation by the elimination of character prototypes from the reference set. This idea, originally developed for simple character templates, can straightforwardly be transferred to the CSDTW allograph models.

CSDTW dissimilarity: The CSDTW dissimilarity based on the Bayes error is a meaningful measure for the dissimilarity between two generative sequence models. However, as mentioned above, it cannot be interpreted as the probability of a misclassification, as the Bayes error can for vector space data. In this respect, interesting future work lies in the development of a "sequence Bayes error" that can serve as an analogon for the vector space Bayes error.

Further, the incorporation of the dissimilarity measure into the model training and classification, as described in Section 6.1, is an interesting and practically relevant research topic.

SVM-GDTW: Currently, SVM-GDTW has two substantial limitations.

First, its computational and memory complexities are rather high. In order to alleviate this problem, an interpretation of its complexity (Equation (7.16)) suggests the following starting points: The number of kernel evaluations effects the complexity linearly. The number of kernel evaluations itself is dependent on the number of support vectors. Thus, one goal should be to decrease this number. Techniques have been reported [Schölkopf et al., 1999] that reduce the number of support vectors by a factor of up to ten, posterior to the SVM training, without large losses in classification accuracy. Further, in recent years, a kernel-based classification method similar to SVMs has been proposed, that is especially designed to produce sparse classifiers: the relevance vector machine (RVM) [Tipping, 2001]. This classifier has further the advantage that its output has a natural meaning of probability. Another source for speed-up is to omit or interrupt applicable kernel evaluations by pruning techniques for the DTW. Last but not least, a sub-sampling of the sequence patterns decreases the recognition complexity linearly.

Second, the SVM classification only solves the character recognition. It does not provide an implicit segmentation of the test sequence, when the latter corresponds to a word, nor a probability value, like generative algorithms usually do. From this background, an important, but demanding challenge is the installation of an SVM solution for the recognition of words and unconstrained handwriting.

In addition, it is worthwhile to study techniques that could further improve the accuracy of SVM-GDTW. Attractive starting points include the following. So far, the DTW uses the simple squared Euclidean distance for the kernel evaluation, i.e., each sub-space dimension is weighted uniformly. It can be expected that a non-uniform weighting

leads to an increased accuracy. A simple approach could use weights from generative models, for instance, those gained in the context of CSDTW in Section 4.6.2. Then, a further study of GDTW's theoretical characteristics, especially in the context of its positive definiteness, is worthwhile. Such research could suggest an adjustment of the kernel function, resulting in a deeper theoretical foundation and possibly an improved classification accuracy.

Statistical modeling of semi-directional data: The feasibility of the proposed directional data modeling depends on the underlying data, as a small variance in the directional dimension is required. In this respect, the described solution is a first, but not the final step to a unified handling of a semi-wrapped feature space in pattern recognition. For further research, in particular a non-approximative modeling of semi-directional situations based on the wrapped or the von Mises distribution is a challenge worthwhile to study.

From a more general view, a basis for interesting future research is the integration of CSDTW and SVM-GDTW into a hybrid classifier. This is especially promising, as both classifiers employ different philosophies, and thus their decisions can be expected to be uncorrelated to some degree. Indeed, studies from the two-class SVM-GDTW experiments (Table 7.1) indicate significant differences in the individual empirical performance of selected pairwise classifiers.

For a successful integration of the two concepts various approaches can be pursued: First, literature suggests a number of general classifier combination approaches [Jain et al., 2000]. Those combine the outputs of individual classifiers based on a parallel, cascading or hierarchical combination scheme.

Second, a trained SVM could be used to identify the critical boundary regions in the feature space. Following, the CSDTW models can be refined especially in these regions.

Third, CSDTW classification could be employed to identify a small number of class hypotheses instead of a definite decision. Then, SVM-GDTW would only have to choose out of a few classes, which would speed up SVM-GDTW classification drastically.

This thesis shall be be concluded with the remark that the area of HWR still offers a number of exciting challenges. In particular, the advent of the *Tablet PC* provides a ubiquitous platform, which demands for the recognition of "digital ink" in its most general form. In addition to the recognition of script, researchers in industry and academia are targeting for the recognition of further complex elements, such as personal notes, layout structures, mathematical formulas, gestures, etc., ideally to be embedded within an interactive and adaptive environment.

Bibliography

L. R. Bahl, F. Jelinek, and R. L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Trans. Pattern Anal. and Mach. Intell.*, 5:179–190, Mar. 1983.

L. R. Bahl, P. Brown, P. D. Souza, and R. L. Mercer. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 49–52, Tokyo, JP, 1986.

C. Bahlmann. Directional features in online handwriting recognition. *Accepted for publication by Pattern Recognition*, 2005.

C. Bahlmann and H. Burkhardt. Measuring HMM similarity with the Bayes probability of error and its application to online handwriting recognition. In *Proc. 6th Int. Conf. Doc. Anal. and Recognition (ICDAR)*, pages 406–411, Seattle, WA, 2001.

C. Bahlmann and H. Burkhardt. The writer independent online handwriting recognition system *frog on hand* and cluster generative statistical dynamic time warping. *IEEE Trans. Pattern Anal. and Mach. Intell.*, 26(3):299–310, Mar. 2004.

C. Bahlmann, B. Haasdonk, and H. Burkhardt. On-line handwriting recognition with support vector machines—a kernel approach. In *Proc. 8th Int. Workshop Front. Handwriting Recognition (IWFHR)*, pages 49–54, Niagara-on-the-Lake, Canada, 2002.

R. Bellmann. *Dynamic Programming*. Princeton University Press, 1957.

S. Bercu and G. Lorette. On-line handwritten word recognition: an approach based on hidden Markov models. In *Proc. 3rd Int. Workshop Front. Handwriting Recognition (IWFHR)*, pages 385–390, Buffalo, USA, 1993.

C. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

D. Bockhorn. Bestimmung und Untersuchung von Signifikanzgewichtungen für die Erkennung von handgeschriebenen Buchstaben. Master's thesis, Albert-Ludwigs-Universität Freiburg, Institut für Informatik, 2000. URL `http://lmb.informatik.uni-freiburg.de/`.

B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proc. Fifth Annual ACM Workshop on Computational Learning Theory*, pages 144–152, 1992.

E. R. Brocklehurst and P. D. Kenward. Preprocessing for cursive script recognition. *NPL Report DITC 132/88*, Nov. 1988.

C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

C. Campbell. An introduction to kernel methods. In R. J. Howlett and L. C. Jain, editors, *Radial Basis Function Networks: Design and Applications*, chapter 7. Springer Verlag, Berlin, 2000.

M. P. Carmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976.

G. C. Cawley. MATLAB support vector machine toolbox (v0.50β). University of East Anglia, School of Information Systems, Norwich, Norfolk, U.K. NR4 7TJ, 2000. URL `http://theoval.sys.uea.ac.uk/~gcc/svm/toolbox/`.

S. Chakrabartty and G. Cauwenberghs. Forward decoding kernel machines: A hybrid HMM/SVM approach to sequence recognition. In *Int. Workshop on Pattern Recognition with Support Vector Machines*, number 2388 in LNCS, pages 278–292. Springer Verlag, 2002.

S. D. Connell and A. K. Jain. Learning prototypes for on-line handwriting digits. In *Proc. 12th Int. Conf. Pattern Recognition (ICPR)*, pages 182–184, Brisbane, Australia, 1994.

N. Cristianini and J. Shawe-Taylor. *Support Vector Machines*. Cambridge University Press, 2000.

C. deBoor. *A Practical Guide to Splines*. Springer Verlag, 1978.

D. DeCoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 46(1/3):161, 2002.

A. Dengel, R. Hoch, F. Hönes, T. Jäger, M. Malburg, and A. Weigel. Techniques for improving OCR results. In H. Bunke and P. S. P. Wang, editors, *Handbook of Character Recognition and Document Image Analysis*, pages 227–258. World Scientific, 1997.

N. Deshmukh, A. Ganapathiraju, and J. Picone. Hierarchical search for large-vocabulary conversational speech recognition. *IEEE Signal Processing Magazine*, 16(5):84–107, Sept. 1999.

P. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice Hall, 1982.

R. Döker, T. Maurer, W. Kremer, K.-P. Neidig, and H. R. Kalbitzer. Determination of mean and standard deviation of dihedral angles. *Biochem. Biophys. Res. Com.*, 257(2):348–350, Apr. 1999.

R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley InterScience, second edition, 2001.

A. Elms. *The representation and recognition of text using hidden Markov models*. PhD thesis, University of Surrey, Vision, Speech and Signal Processing Group, Department of Electronic and Electrical Engineering, Apr. 1996.

M. Falkhausen, H. Reininger, and D. Wolf. Calculation of distance measures between hidden Markov models. In *Proc. Eurospeech*, pages 1487–1490, 1995.

S. Fine, G. Saon, and R. A. Gopinath. Digit recognition in noisy environments via a sequential GMM/SVM system. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2002.

A. Ganapathiraju, J. Hamaker, and J. Picone. Advances in hybrid SVM/HMM speech recognition. In *Int. Conf. on Speech and Language Processing (ICSLP)*, 2002.

N. Gauthier, T. Artières, B. Dorizzi, and P. Gallinari. Strategies for combining on-line and off-line information in an on-line handwriting recognition system. In *Proc. 6th Int. Conf. Doc. Anal. and Recognition (ICDAR)*, pages 412–416, Seattle, WA, 2001.

D. Goldberg and C. Richardson. Touch-typing with a stylus. In *Proc. Int. Conf. Hum. Factors Comput. Syst.*, 1993.

W. Guerfali and R. Plamondon. Normalizing and restoring on-line handwriting. *Pattern Recognition*, 16(5), 1993.

E. G. Gumbel, J. A. Greenwood, and D. Durand. The circular normal distribution: Theory and tables. *J. Amer. Statist. Assoc.*, 48:131–152, Mar. 1953.

S. R. Gunn. MATLAB support vector machine toolbox, Mar. 1998. URL `http://www.isis.ecs.soton.ac.uk/isystems/kernel/`.

I. Guyon, P. Albrecht, Y. LeCun, J. S. Denker, and W. Hubbard. Design of a neural network character recognizer for a touch terminal. *Pattern Recognition*, 24(2):105–119, 1991.

I. Guyon, L. R. B. Schomaker, R. Plamondon, M. Liberman, and S. Janet. UNIPEN project of on-line data exchange and recognizer benchmarks. In *Proc. 12th Int. Conf. Pattern Recognition (ICPR)*, pages 29–33, Jerusalem, Israel, 1994. URL `http://www.unipen.org/`.

I. Guyon, M. Schenkel, and J. Denker. Overview and synthesis of on-line cursive handwriting recognition techniques. In H. Bunke and P. S. P. Wang, editors, *Handbook of Character Recognition and Document Image Analysis*, chapter 7, pages 183–225. World Scientific, 1997.

B. Haasdonk. Feature space interpretation of SVMs with indefinite kernels. *IEEE Trans. Pattern Anal. and Mach. Intell.*, 2005. to appear.

B. Haasdonk and C. Bahlmann. Learning with distance substitution kernels. In *26th Pattern Recognition Symposium of the German Association for Pattern Recognition (DAGM)*, Tübingen, Germany, 2004. Springer Verlag.

B. Haasdonk and D. Keysers. Tangent distance kernels for support vector machines. In *Proc. 16th Int. Conf. Pattern Recognition (ICPR)*, Quebec, Canada, 2002.

J. Hamaker, J. Picone, and A. Ganapathiraju. A sparse modeling approach to speech recognition based on relevance vector machines. In *Int. Conf. on Speech and Language Processing (ICSLP)*, 2002.

C. W. Hsu and C. J. Lin. A comparison on methods for multi-class support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2001.

J. Hu, S. G. Lim, and M. K. Brown. Writer independent on-line handwriting recognition using an HMM approach. *Pattern Recognition*, 33:133–147, Jan. 2000.

S. G. Hyberts, M. S. Goldberg, T. F. Havel, and G. Wagner. The solution structure of eglin c based on measurements of many NOEs and coupling constants and its comparison with X-ray structures. *Protein Science*, 1:736–751, 1992.

T. Jaakkola, M. Diekhans, and D. Haussler. Using the Fisher kernel method to detect remote protein homologies. In *Proc. 8th Int. Conf. on Image Anal. and Processing*, 1999.

S. Jäger. *Recovering Dynamic Information from Static, Handwritten Word Images*. PhD thesis, University of Freiburg, 1998.

S. Jäger, S. Manke, and A. Waibel. NPEN++: An on-line handwriting recognition system. In *Proc. 7th Int. Workshop Front. Handwriting Recognition (IWFHR)*, pages 249–260, Amsterdam, The Netherlands, 2000.

A. K. Jain, R. P. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Trans. Pattern Anal. and Mach. Intell.*, 22(1):4–37, Jan. 2000.

F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, 1998.

B. H. Juang and L. R. Rabiner. A probabilistic distance measure for hidden Markov models. *AT&T Tech. J.*, 64(2):391–408, Feb. 1985.

B. H. Juang and L. R. Rabiner. The segmental k-means algorithm for estimating parameters of hidden Markov models. *IEEE Trans. Acoust. Speech Signal Process.*, 38:1639–1641, 1990.

B. H. Juang, W. Chou, and C. H. Lee. Minimum classification error rate methods for speech recognition. *IEEE Trans. Speech & Audio Proc. T-SAP*, 5(3):257–265, May 1997.

E. Kavallieratou, N. Fakotakis, and G. Kokkinakis. New algorithms for skewing correction and slant removal on word-level. In *Int. Conf. on Electronics, Circuits and Systems*, volume 2, pages 1159–1162, Pafos, Cyprus, 1999.

A. Kosmala. *HMM-basierte Online Handschrifterkennung*. PhD thesis, Faculty of Electrical Engineering, Gerhard-Mercator-University Duisburg, Dec. 2000.

A. Kosmala, G. Rigoll, S. Lavirotte, and L. Pottier. On-line handwritten formula recognition using hidden markov models and context dependent graph grammars. In *Proc. 5th Int. Conf. Doc. Anal. and Recognition (ICDAR)*, page 107, Bangalore, India, 1999.

S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

S. Kwong, Q. H. He, K. F. Man, and K. S. Tang. A maximum model distance approach for HMM-based speech recognition. *Pattern Recognition*, 31(3):219–229, 1998.

P. M. Lallican, C. Viard-Gaudin, and S. Knerr. From off-line to on-line handwriting recognition. In *Proc. 7th Int. Workshop Front. Handwriting Recognition (IWFHR)*, pages 303–312, Amsterdam, The Netherlands, 2000.

J. J. Lee, J. Kim, and J. H. Kim. Data driven design of HMM topology for on-line handwriting recognition. In *Proc. 7th Int. Workshop Front. Handwriting Recognition (IWFHR)*, pages 239–248, Amsterdam, The Netherlands, 2000.

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. *AT&T Tech. J.*, 62(4):1035–1074, 1983.

J. Lin. Divergence measures based on the Shannon entropy. *IEEE Trans. Inform. Theory*, 37 (1):145–151, Jan. 1991.

B. C. Lovell, P. J. Kootsookos, and R. C. Williamson. The circular nature of discrete-time frequency estimates. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, volume 5, page 3369, 1991.

R. B. Lyngsø, C. N. S. Pedersen, and H. Nielsen. Metrics and similarity measures for hidden Markov models. In *Proc. 8th Int. Conf. on Image Anal. and Processing*, pages 178–186, 1999.

I. S. MacKenzie and R. W. Soukoreff. Text entry for mobile computing: Models and methods, theory and practice. In *Human-Computer Interaction*, volume 17, pages 147–198. Lawrence Erlbaum Associates, Inc., 2002.

S. Manke, M. Finke, and A. Waibel. Combining bitmaps with dynamic writing information for on-line handwriting recognition. In *Proc. 12th Int. Conf. Pattern Recognition (ICPR)*, pages 596–598, Jerusalem, Israel, 1994.

S. Manke, M. Finke, and A. Waibel. NPen++: A writer independent, large vocabulary on-line cursive handwriting recognition system. In *Proc. 3rd Int. Conf. Doc. Anal. and Recognition (ICDAR)*, Montreal, Canada, 1995.

S. Manke, M. Finke, and A. Waibel. A fast search technique for large vocabulary on-line handwriting recognition. In *Proc. 5th Int. Workshop Front. Handwriting Recognition (IWFHR)*, Colchester, UK, 1996.

K. V. Mardia. *Statistics of Directional Data*. Academic Press, 1972.

N. Merhav and Y. Ephraim. Hidden Markov modeling using a dominant state sequence with application to speech recognition. *Computer Speech & Language*, 5(4):327–339, 1991.

H. Ney. The use of a one-stage dynamic programming algorithm for connected word recognition. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 263–271, 1984.

H. Ney. Stochastic grammars and pattern recognition. In P. Laface and R. De Mori, editors, *Speech Recognition and Understanding. Recent Advances, Trends and Applications*, pages 319–344. Springer Verlag, 1992.

T. Oates, L. Firoiu, and P. Cohen. Clustering time series with hidden Markov models and dynamic time warping. In *Proc. IJCAI-99 Workshop on Neural, Symbolic and Reinforcement Learning Methods for Sequence Learning*, pages 17–21, 1999.

T. Oates, M. D. Schmill, and P. Cohen. A method for clustering the experiences of a mobile robot that accords with human judgments. In *Proc. of the 17th National Conf. on Artificial Intell.*, pages 846–851, 2000.

M. Parizeau, A. Lemieux, and C. Gagné. Character recognition experiments using UNIPEN data. In *Proc. 6th Int. Conf. Doc. Anal. and Recognition (ICDAR)*, pages 481–485, Seattle, WA, 2001.

M. P. Perrone and S. D. Connell. K-means clustering for hidden Markov models. In *Proc. 7th Int. Workshop Front. Handwriting Recognition (IWFHR)*, pages 229–238, Amsterdam, The Netherlands, 2000.

R. Plamondon and S. N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Trans. Pattern Anal. and Mach. Intell.*, 22(1):63–84, Jan. 2000.

R. Plamondon, D. Lopresti, L. R. B. Schomaker, and R. Srihari. On-line handwriting recognition. In J. G. Webster, editor, *Encyclopedia of Electrical and Electronics Engineering*, volume 15, pages 123–146. Wiley InterScience, 1999.

J. C. Platt. Probabilities for sv machines. In *Advances in Large Margin Classifiers*, pages 61–73. MIT Press, 2000.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Scholkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods—Support Vector Learning*, chapter 12, pages 185–208. MIT Press, 1999.

J. C. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGS for multiclass classification. In *Advances in Neural Information Processing Systems 12*. MIT Press, 2000.

L. Prevost and M. Milgram. Non-supervised determination of allograph sub-classes for on-line omni-scriptor handwriting recognition. In *Proc. 5th Int. Conf. Doc. Anal. and Recognition (ICDAR)*, Bangalore, India, 1999.

L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–286, 1989. Advanced Artikel ueber HMM vom Erfinder selbst; Lernen von HMM mit Baum-Welch, oft zitiert.

L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3:4–16, 1986.

L. R. Rabiner and B. H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

M. Schenkel, I. Guyon, and D. Henderson. On-line cursive script recognition using time delay neural networks and hidden Markov models. In R. Plamondon, editor, *Machine Vision and Applications 8*, pages 215–223. Springer Verlag, 1995.

B. Schölkopf. *Support Vector Learning*. PhD thesis, Technische Universität Berlin, Fachbereich 13 — Informatik, 1997.

B. Schölkopf and A. J. Smola. *Learning with Kernels — Support Vector Machines, Regularization, Optimisation, and Beyond*. MIT Press, 2002.

B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. J. Smola. Input space versus feature space in kernel-based methods. *IEEE Trans. Neural Networks*, 10(5):1000–1017, Sept. 1999.

L. R. B. Schomaker. Using stroke- or character-based self-organizing maps in the recognition of on-line, connected cursive script. *Pattern Recognition*, 26(3):443–450, 1993.

E. G. Schukat-Talamazzini. *Automatische Spracherkennung—Grundlagen, statistische Modelle und effiziente Algorithmen*. Friedr. Vieweg & Sohn, 1995.

J. Schürmann. *Pattern Classification: a unified view of statistical and neural approaches*. Wiley InterScience, 1996.

G. Seni, R. K. Srihari, and N. Nasrabadi. Large vocabulary recognition of on-line handwritten cursive words. *IEEE Trans. Pattern Anal. and Mach. Intell.*, pages 757–762, 1996.

H. Shatkay and L. P. Kaelbling. Heading in the right direction. In *Proc. 15th Int. Conf. on Machine Learning*, pages 531–539. Morgan Kaufmann Publishers, Inc., 1998.

M. Shilman, Z. Wei, S. Raghupathy, P. Simard, and D. Jones. Discerning structure from freeform handwritten notes. In *Proc. 7th Int. Conf. Doc. Anal. and Recognition (ICDAR)*, pages 60–65, Edinburgh, Scotland, 2003.

H. Shimodaira, K. Noma, M. Nakai, and S. Sagayama. Support vector machine with dynamic time-alignment kernel for speech recognition. In *Proc. Eurospeech*, 2001a.

H. Shimodaira, K. Noma, M. Nakai, and S. Sagayama. Dynamic time-alignment kernel in support vector machine. In *Advances in Neural Information Processing Systems 13*. MIT Press, 2001b.

K. Simon. Vorverarbeitung und Merkmalsextraktion in der Online-Handschrifterkennung. BS thesis, Albert-Ludwigs-Universität Freiburg, Institut für Informatik, 2002. URL `http://lmb.informatik.uni-freiburg.de/`.

K. Simon. Erkennung von handgeschriebenen Wörtern mit CSDTW. Master's thesis, Albert-Ludwigs-Universität Freiburg, Institut für Informatik, 2003. URL `http://lmb.informatik.uni-freiburg.de/`.

Y. Singer and M. K. Warmuth. Training algorithms for hidden Markov models using entropy based distance functions. In *Advances in Neural Information Processing Systems 11*, page 641. MIT Press, 1999.

N. Smith and M. Gales. Speech recognition using SVMs. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

P. Smyth. Clustering sequences with hidden Markov models. In *Advances in Neural Information Processing Systems 9*, pages 648–654. MIT Press, 1997.

C. Y. Suen, J. Kim, K. Kim, Q. Xu, and L. Lam. Handwriting recognition—the last frontiers. In *Proc. 15th Int. Conf. Pattern Recognition (ICPR)*, volume 4, pages 1–10, Barcelona, Spain, 2000.

C. C. Tappert, C. Y. Suen, and T. Wakahara. The state of the art in on-line handwriting recognition. *IEEE Trans. Pattern Anal. and Mach. Intell.*, 12(8):787–808, Aug. 1990.

S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, 1999.

M. E. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, June 2001.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

A. Vinciarelli and M. Perrone. Combining online and offline handwriting recognition. In *Proc. 7th Int. Conf. Doc. Anal. and Recognition (ICDAR)*, page 844, Edinburgh, Scotland, 2003.

R. von Mises. Über die „Ganzzahligkeit“ der Atomgewichte und verwandte Fragen. *Phys. Z.*, 19:490–500, 1918.

H. Vullings, M. Verhaegen, and H. Verbruggen. ECG segmentation using time-warping. In *Advances in Intell. Data Analysis*, pages 275–285, 1997.

V. Vuori, J. Laaksonen, E. Oja, and J. Kangas. Experiments with adaptation strategies for a prototype-based recognition system for isolated handwritten characters. *Internat. J. on Document Anal. and Recognition*, 3(2):150–159, 2001.

L. G. Vuurpijl and L. R. B. Schomaker. Finding structure in diversity: A hierarchical clustering method for the categorization of allographs in handwriting. In *Proc. 4th Int. Conf. Doc. Anal. and Recognition (ICDAR)*, pages 387–393, Ulm, Germany, 1997.

T. Wakahara, H. Murase, and K. Odaka. On-line handwriting recognition. *Proc. IEEE*, 80(7): 1181–1194, July 1992.

C. Watkins. Dynamic alignment kernels. In *Advances in Large Margin Classifiers*, pages 39–50. MIT Press, 2000.

L. Wenyin, W. Qian, R. Xiao, and X. Jin. Smart sketchpad - an on-line graphics recognition system. In *Proc. 6th Int. Conf. Doc. Anal. and Recognition (ICDAR)*, page 1050, Seattle, WA, 2001.

A. Wintner. On the stable distribution laws. *Amer. J. Math.*, 55:335–339, 1933.

F. Zernike. Wahrscheinlichkeitsrechnung und mathematische Statistik. In H. Thirring, editor, *Handbuch der Physik*, volume 3, chapter 12, pages 419–492. Springer Verlag, 1928.

R. Zhang and A. I. Rudnicky. A large scale clustering scheme for kernel k-means. In *Proc. 16th Int. Conf. Pattern Recognition (ICPR)*, Quebec, Canada, 2002.