ADAPTIVE COMBINATIONS OF CLASSIFIERS WITH APPLICATION TO ON-LINE HANDWRITTEN CHARACTER RECOGNITION

Matti Aksela

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Department of Computer Science and Engineering for public examination and debate in Auditorium TU2 at Helsinki University of Technology (Espoo, Finland) on the 29th of March, 2007, at 12 o'clock.

Helsinki University of Technology
Department of Computer Science and Engineering
Laboratory of Computer and Information Science
P.O.Box 5400
FIN-02015 HUT
FINLAND

ABSTRACT

Classifier combining is an effective way of improving classification performance. User adaptation is clearly another valid approach for improving performance in a user-dependent system, and even though adaptation is usually performed on the classifier level, also adaptive committees can be very effective. Adaptive committees have the distinct ability of performing adaptation without detailed knowledge of the classifiers. Adaptation can therefore be used even with classification systems that intrinsically are not suited for adaptation, whether that be due to lack of access to the workings of the classifier or simply a classification scheme not suitable for continuous learning.

This thesis proposes methods for adaptive combination of classifiers in the setting of on-line handwritten character recognition. The focal part of the work introduces adaptive classifier combination schemes, of which the two most prominent ones are the Dynamically Expanding Context (DEC) committee and the Class-Confidence Critic Combining (CCCC) committee. Both have been shown to be capable of successful adaptation to the user in the task of on-line handwritten character recognition. Particularly the highly modular CCCC framework has shown impressive performance also in a doubly-adaptive setting of combining adaptive classifiers by using an adaptive committee.

In support of this main topic of the thesis, some discussion on a methodology for deducing correct character labeling from user actions is presented. Proper labeling is paramount for effective adaptation, and deducing the labels from the user's actions is necessary to perform adaptation transparently to the user. In that way, the user does not need to give explicit feedback on the correctness of the recognition results.

Also, an overview is presented of adaptive classification methods for single-classifier adaptation in handwritten character recognition developed at the Laboratory of Computer and Information Science of the Helsinki University of Technology, CIS-HCR. Classifiers based on the CIS-HCR system have been used in

the adaptive committee experiments as both member classifiers and to provide a reference level.

Finally, two distinct approaches for improving the performance of committee classifiers further are discussed. Firstly, methods for committee rejection are presented and evaluated. Secondly, measures of classifier diversity for classifier selection, based on the concept of diversity of errors, are presented and evaluated.

The topic of this thesis hence covers three important aspects of pattern recognition: on-line adaptation, combining classifiers, and a practical evaluation setting of handwritten character recognition. A novel approach combining these three core ideas has been developed and is presented in the introductory text and the included publications.

To reiterate, the main contributions of this thesis are: 1) introduction of novel adaptive committee classification methods, 2) introduction of novel methods for measuring classifier diversity, 3) presentation of some methods for implementing committee rejection, 4) discussion and introduction of a method for effective label deduction from on-line user actions, and as a side-product, 5) an overview of the CIS-HCR adaptive on-line handwritten character recognition system.

TIIVISTELMÄ

Luokittimien yhdistäminen komitealuokittimella on tehokas keino luokitustarkkuuden parantamiseen. Laskentatehon jatkuva kasvu tekee myös useiden luokittimien yhtäaikaisesta käytöstä yhä varteenotettavamman vaihtoehdon. Järjestelmän adaptoituminen (mukautuminen) käyttäjään on toinen hyvä keino käyttäjäriippumattoman järjestelmän tarkkuuden parantamiseksi. Vaikka adaptaatio yleensä toteutetaan luokittimen tasolla, myös adaptiiviset komitealuokittimet voivat olla hyvin tehokkaita. Adaptiiviset komiteat voivat adaptoitua ilman yksityiskohtaista tietoa jäsenluokittimista. Adaptaatiota voidaan näin käyttää myös luokittelujärjestelmissä, jotka eivät ole itsessään sopivia adaptaatioon. Adaptaatioon sopimattomuus voi johtua esimerkiksi siitä, että luokittimen totetutusta ei voida muuttaa, tai siitä, että käytetään luokittelumenetelmää, joka ei sovellu jatkuvaan oppimiseen.

Tämä väitöskirja käsittelee menetelmiä luokittimien adaptiiviseen yhdistämiseen käyttäen sovelluskohteena käsinkirjoitettujen merkkien on-line-tunnistusta. Keskeisin osa työtä esittelee uusia adaptiivisia luokittimien yhdistämismenetelmiä, joista kaksi huomattavinta ovat Dynamically Expanding Context (DEC) -komitea sekä Class-Confidence Critic Combining (CCCC) -komitea. Molemmat näistä ovat osoittautuneet kykeneviksi tehokkaaseen käyttäjä-adaptaatioon käsinkirjoitettujen merkkien on-line-tunnistuksessa. Erityisesti hyvin modulaarisella CCCC järjestelmällä on saatu hyviä tuloksia myös kaksinkertaisesti adaptiivisessa asetelmassa, jossa yhdistetään adaptiivisia jäsenluokittimia adaptiivisen komitean avulla.

Väitöskirjan pääteeman tukena esitetään myös malli ja käytännön esimerkki siitä, miten käyttäjän toimista merkeille voidaan päätellä oikeat luokat. Merkkien todellisen luokan onnistunut päättely on elintärkeää tehokkaalle adaptaatiolle. Jotta adaptaatio voitaisiin suorittaa käyttäjälle läpinäkyvästi, merkkien todelliset luokat on kyettävä päättelemään käyttäjän toimista. Tällä tavalla käyttäjän ei tarvitse antaa suoraa palautetta tunnistustuloksen oikeellisuudesta.

Työssä esitetään myös yleiskatsaus Teknillisen Korkeakoulun Informaatioteknii-kan Laboratoriossa kehitettyyn adaptiiviseen käsinkirjoitettujen merkkien tunnis-tusjärjestelmään. Tähän järjestelmään perustuvia luokittimia on käytetty adap-tiivisten komitealuokittimien kokeissa sekä jäsenluokittimina että vertailutasona.

Lopuksi esitellään kaksi erillistä menetelmää komitealuokittimen tarkkuuden edel-leen parantamiseksi. Näistä ensimmäinen on joukko menetelmiä komitealuokit-timen rejektion (hylkäyksen) toteuttamiseksi. Toinen esiteltävä menetelmä on käyttää luokittimien erilaisuuden mittoja jäsenluokittimien valintaa varten. Eh-dotetut uudet erilaisuusmitat perustuvat käsitteeseen, jota kutsumme virheiden erilaisuudeksi.

Väitöskirjan aihe kattaa kolme hahmontunnistuksen tärkeää osa-aluetta: on-line-adaptaation, luokittimien yhdistämisen ja käytännön sovellusalana käsinkir-joitettujen merkkien tunnistuksen. Näistä kolmesta lähtökohdasta on kehitetty uudenlainen synteesi, joka esitetään johdantotekstissä sekä liitteenä olevissa jul-kaisuissa.

Tämän väitöskirjan oleellisimmat kontribuutiot ovat siten: 1) uusien adaptiivis-ten komitealuokittimien esittely, 2) uudenlaisten menetelmien esittely luokitti-mien erilaisuuden mittaamiseksi, 3) joidenkin komitearejektiomenetelmien esit-tely, 4) pohdinnan ja erään toteutustavan esittely syötettyjen merkkien todel-lisen luokan päättelemiseksi käyttäjän toimista, sekä sivutuotteena 5) kattava yleiskatsaus CIS-HCR adaptiiviseen on-line käsinkirjoitettujen merkkien tunnis-tusjärjestelmään.

Preface

The work presented in this thesis has been performed at the Helsinki University of Technology, Laboratory of Computer and Information Science. I would like to express my gratitude for my supervisor Professor Erkki Oja for making this research possible and making his vast knowledge and insight available. I have been given much freedom in performing the work that constitutes this thesis, but my adviser, Docent Jorma Laaksonen, has always been there for me with any problems I have encountered along the way. He has offered countless helpful suggestions and ideas as well as hands-on help with more practical issues. For this he will always have my utmost gratitude and respect.

I would also like to express my gratitude to Jari Kangas at Nokia Research Center. Collaboration with him was the starting point of the entire handwritten character recognition project, which started in a theme under TEKES. Since that project ended, funding from the ComMIT graduate school and our department and laboratory have made the completion of this work possible. I would also like to acknowledge the PASCAL network of excellence, which I have been a member of, and the grants obtained from the Nokia Foundation.

Special thanks go to all my co-authors, Professor Erkki Oja, Docent Jari Kangas, Dr. Vuokko Vuori, M.Sc. Ramūnas Girdziušas, and most of all my adviser Docent Jorma Laaksonen who has spent much time also working on our joint publications.

The pre-examiners of my thesis, Dr. David Windridge and Dr. Jarmo Hurri, truly made my thesis much better with their thorough examination and offering their knowledge through a multitude of very insightful comments – for your input I am most grateful. And I would like to thank the opponent in my dissertation, Professor Robert P.W. Duin, in advance – and hopefully in hindsight I will be able to say that I did not make a fool out of myself in the dissertation!

Last but certainly not least I would like to thank the most important thing in my life, my family. My parents both showed example and supported my aspirations all the way through and have done their best to help me in any way they could. And most importantly, my two wonderful sons Otso and Hugo and my beloved wife Kia – without your support this would not have been possible. And without my family it would not have meant anything in the end anyway.

Otaniemi, March 2007

Matti Aksela

Contents

List of symbols

$(A)R \rightarrow B$	DEC production rule with one-sided context
$B_i(x, c)$	Sum of classes ranked below class c by classifier i for input sample x, Eq. (2.7)
$BC(x, c)$	Borda count for input sample x and class c, Eq. (2.7)
b	kernel bandwidth for CCCC kernel-based distribution models, Eq. (5.16–5.18)
C	number of classes
$c^i(x)$	output label of ith classifier for input sample x
$c^{\text{true}}(x)$	true label of input sample x
$c_i, i \in \{1, \dots, C\}$	class labels
c_r	reject class identifier ($r = C + 1$)
$c_m(x)$	majority voting result for input sample x, Eq. (2.6)
$c_{mcbl}(x)$	MCBL result for input sample x, Eq. (5.7)
$c_p(x)$	plurality voting result for input sample x, Eq. (2.5)
$c_{wp}(x)$	weighted plurality voting result for input sample x, Eq. (5.3)
$c_{\text{prod}}(x)$	product rule decision for input sample x, Eq. (5.28)
$c_{\text{sum}}(x)$	sum rule decision for input sample x, Eq. (5.29)
$c_{\text{min}}(x)$	min rule decision for input sample x, Eq. (5.30)
$c_{\text{max}}(x)$	max rule decision for input sample x, Eq. (5.31)
$Cov(\cdot)$	covariance
$d^k(x)$	vector of distances obtained from classifier k for input x
$d_c^k(x)$	c:th component of $d^k(x)$
$d_{\text{DR}}(x)$	distance rejection averaged distance, Eq. (6.1)
$D_{\text{DTW}}(\cdot)$	DTW distance, Eq. (2.2)
$DIS(\cdot)$	Disagreement diversity measure, Eq. (7.6)
$DF(\cdot)$	Double-Fault diversity measure, Eq. (7.7)
$DFD(\cdot)$	Distinct Failure Diversity, Eq. (7.10)

$DP_x(v,s)$ (v,s):th element of the $K \times C$ dimensional Decision Profile matrix for input x, Eq. (5.6)

$DT_c(v,s)$ (v,s):th element of the $K \times C$ dimensional Decision Template matrix for class c, Eq. (2.8)

$EEC(\cdot)$ Exponential Error Count diversity measure, Eq. (7.13)

$f^k(c^k(x))$ MCBL class-wise confidence value for class c of classifier k and input sample x

$I(\cdot)$ mutual information of classifiers, Eq. (7.3)

$IE(\cdot)$ mutual information of classifier errors, Eq. (7.8)

K number of classifiers

$l^k(x)$ MCBL distance ratio for classifier k and input x, Eq. (5.8)

$L(A)R \rightarrow B$ DEC production rule with two-sided context

$MGR(\cdot)$ MGR score function (2.9)

N total number of samples

N_i number of samples collected into CCCC distribution i

$n_f(z,i)$ number of points further from the mean of distribution i than the the input z, Eq. (5.13)

$N_{yy}^{xx}(\cdot)$ various notations for counts of correctness in classifier sets (Section 7.2.3)

$r_i(x)$ rank given by classifier i for the input sample x belonging to class c, Eq. (2.9)

$p^i(\cdot)$ confidence from CCCC distribution model i

$p^i_{\mathrm{gaussian}}(\cdot)$ confidence from gaussian distribution model i, Eq. (5.12)

$p^i_{\mathrm{nonparam}}(\cdot)$ confidence from nonparametric distribution model i, Eq. (5.14)

$p^i_{\mathrm{NN}}(\cdot)$ confidence from nearest neighbor distribution model i, Eq. (5.15)

$p^i_{\mathrm{trikernel}}(\cdot)$ confidence from triangular kernel distribution model i, Eq. (5.16)

$p^i_{\mathrm{gausskernel}}(\cdot)$ confidence from gaussian kernel distribution model i, Eq. (5.17)

$p^i_{\mathrm{expkernel}}(\cdot)$ confidence from exponential kernel distribution model i, Eq. (5.18)

$P(\cdot)$ probability of event

$P(i|j)$ conditional probability of event i given event j

$q_c^k(x)$ normalized distance to the nearest prototype of class c for classifier k and input sample x, Eq. (5.11)

$Q(\cdot)$ Q statistic diversity measure, Eq. (7.5)

$SF(\cdot)$ Same Fault diversity measure, Eq. (7.11)

$s_c^k(x)$ Support from classifier k for class c for input x, Eq. (5.4)

$t_c^k(x)$	overall CCCC confidence value for input x, class c and classifier k, Eq. (5.25–5.27)
$t_{\mathrm{mcbl}}^k(x)$	overall CCCC confidence for input x and classifier k using the CCCC MCBL decision rule, Eq. (5.32)
$t_n(\cdot)$	number of times n versions fail in the Distinct Failure Diversity measure(Eq. (7.9)
T_{vote}	Voting Rejection threshold (Section 6.1)
T_r	Distance Rejection threshold (Section 6.1)
T_{step}	Learning Distance Rejection threshold step value, Eq. (6.2,6.3)
$T_r(i)$	Learning Distance Rejection threshold at time step i, Eq. (6.2,6.3)
$u_c^k(x)$	CCCC classification confidence value for input x, class c and classifier k, Eq. (5.23,5.24)
$\mathbf{v}^a$	binary vector of correctness for classifier a
$V(\cdot)$	variance of classifier set diversity measure, Eq. (7.2)
$Var(\cdot)$	variance
w_i	weight assigned to item i
$w_i(z)$	weight from CCCC weighting algorithm for z, Eq. (5.19–5.22)
$w(k)$	weight for classifier k from weighting function, Eq. (5.2)
$WCEC(\cdot)$	Weighted Count of Errors and Correct results diversity measure, Eq. (7.12)
x	input sample
$x_j, j \in \{1, \ldots, N\}$	jth input sample
$y_k(\cdot)$	neural network output
z	shorthand notation for CCCC, $z = q_c^k(x)$
z_j^i	shorthand notation for CCCC, a previously collected $q_c^k(x)$ value number i in distribution j
$\Delta_k(x, c)$	delta function equaling 1 if classifier k suggests the class c for sample x and zero otherwise
$\phi(x, i)$	angle estimate for discretization at point i of input sample x
μ_i	mean of CCCC distribution i
σ_i^2	variance of CCCC distribution i
$\rho(\cdot)$	correlation of classifier errors
$\theta(\cdot)$	neural network activation function

List of abbreviations

AIME	Adaptive Integration of Multiple Experts
ASSOM	Adaptive-Subspace Self-Organizing Map
ANN	Artificial Neural Network
ART	Adaptive Resonance Theory
BKS	Behavior-Knowledge Space
BP	Back-Propagation
CBL	Current-Best-Learning
CCCC	Class Confidence Critic Combining
CIS-HCR	Helsinki University of Technology Laboratory of Computer and Information Science Handwritten Character Recognition system
CR	Class Rejection
DEC	Dynamically Expanding Context
DR	Distance Rejection
DTW	Dynamic Time Warping
EFCL	Elliptical Fuzzy Competitive Learning
EM	Expectation Maximization
GA	Genetic Algorithm
HCR	Handwritten Character Recognition
HMM	Hidden Markov Model
k-NN	k Nearest Neighbor
KA	Kind-of-Area (DTW dissimilarity measure)
KR	Knowledge-based Rejection
LDR	Learning Distance Rejection
LKR	Learning Knowledge-based Rejection
LSC	Local Subspace Classifier
LSC-A	Local Subspace Classifier with adaptation Always
LSC-E	Local Subspace Classifier with adaptation on Errors
LVQ	Learning Vector Quantization
MCBL	Modified Current-Best-Learning

MDL	Minimum Description Length
MGR	Mixed Group Rank
ML	Maximum Likelihood
MLP	Multilayer Perceptron
NN	Neural Network
NPL	Normalized Point-to-Line (DTW dissimilarity measure)
NPP	Normalized Point-to-Point (DTW dissimilarity measure)
PDA	Personal Digital Assistant
PDL	Picture Description Language
PFAM	Probabilistic Fuzzy Artmap
PL	Point-to-Line (DTW dissimilarity measure)
PP	Point-to-Point (DTW dissimilarity measure)
SA	Simple-Area (DTW dissimilarity measure)
SCM	Supervised Clustering and Matching
SNNS	Stuttgart Neural Network Simulator
SSC	Symbol String Classifier
SSC-A	Symbol String Classifier with adaptation Always
SSC-E	Symbol String Classifier with adaptation on Errors
SVM	Support Vector Machine
TDNN	Time-Delay Neural Network
VR	Voting Rejection
1-NN-A	1 Nearest Neighbor classifier with adaptation Always
1-NN-E	1 Nearest Neighbor classifier with adaptation on Errors

Chapter 1

Introduction

1.1 Goals and scope of the thesis

The main motivation for the work presented in this thesis is the constant quest for improving the accuracy of pattern recognition. In this thesis two methods for classification performance improvement are examined in the setting of handwritten character recognition, namely on-line adaptation and classifier combination. In addition, the two are combined to form an on-line adaptive combination of classifiers for handwritten character recognition. The adaptive combination culminates in a doubly adaptive strategy of using an adaptive committee to combine member classifiers that are also adaptive by themselves, that is to say an *adaptive combination of adaptive classifiers*.

All three aspects alone, i.e. *recognition of handwriting*, *on-line adaptation* and *classifier combination*, or any two combined, are topics that have already been addressed in a multitude of publications. However, the combination of all these is something that has received very little attention. Hereby this thesis hopefully fills a gap by combining all of these three core ideas to form adaptive committee classifiers, which are shown to be effective in handwritten character recognition.

The presented methodology has been applied in a scenario where a particular writer inputs a fairly large amount of handwritten character data consecutively, and we can be aware of when the writer changes. In this setting it can be expected that it will be beneficial to use a system that is capable of adapting to the user at hand. Additionally, we have chosen not to use a specific training phase, but

all adaptation must be performed on-line, i.e. during the use of the system. This is considered to be the most user-friendly approach to adaptation, as no extra effort from the user is required.

For experiments with such on-line adaptive recognition systems one needs a database that contains a sufficient amount of data written by each particular writer. Only with enough data from each writer can the effects of the adaptation be properly observed. Even though commonly used benchmarking databases do have a huge number of character samples, there is usually a very limited amount of samples from each writer. We would desire approximately one thousand characters written by each writer to be able to examine the speed and also the possible stagnation of the adaptation. Thus the experiments in this thesis have been conducted on a database collected within our laboratory, where we could perform the data collection to these specifications.

Adaptation to the user is an effective way for improving performance of any user-centered pattern recognition system. This is the case especially with any problem where a very large amount of variation exists, but only a subset of it is being expressed during the use of the system. Just one example of this is the handwritten character recognition scenario we have explored. A practically unlimited number of ways for writing a character exist, but each writer uses only his or her own style with a significantly smaller amount of variation. Individual classifier adaptation has been shown to provide notable benefits. However, the implementation of adaptation to the user at the classifier level naturally leads to a need for accessing that particular classifier in order to tune that classifier to the style of that particular user.

The constant increase in computational power available in even the smallest consumer devices is continuously making methods that require significant amounts of computational resources more and more viable for practical application. One method that can be computationally intensive is classifier combining where several classifiers are used to recognize the same input. Although approaches that suggest combining several less powerful classifiers do exist, many combination schemes are based on the notion of combining classifiers that are proficient themselves, hence effectively multiplying the computational requirements of such systems. Our studies will examine what benefit can be obtained through combining classifiers in an adaptive fashion.

The use of adaptation on the committee level enables us, from very little information about the classifiers themselves, to significantly improve recognition performance by using a committee that learns the behavior of the classifiers, i.e. an *adaptive committee.* Even though the adaptive combination methods are pre-

sented and evaluated in the setting of on-line handwritten character recognition, all presented committee adaptation methodologies are widely applicable as very little information on the application area and the classifiers being combined is required.

Also classifiers that are adaptive in themselves can be combined using an adaptive committee classifier. It will be seen that it is possible to obtain even more benefits through the use of this doubly adaptive scheme. However, the adaptive and thus intrinsically unstable nature of the classifiers places extra requirements on the combination rules for obtaining robust behavior. Both the requirements for *adaptive combination of adaptive classifiers* and working examples will be presented.

Rejection is often implemented in classification systems for the purpose of refraining from classifying difficult samples that are likely to cause errors, or redirecting them to a part of the system specialized towards handling such samples. The same view can also be taken with committee classifiers, and some ways of implementing *committee rejection* are also addressed in this thesis as means for possible further improvement in classification performance.

The member classifiers used in classifier combination will have a significant effect on the performance of any committee. Choosing the most suitable classifiers can be a very difficult task, as the desired set is also dependent on the combination method used. Thus if a way of evaluating the classifiers in a more general fashion could be used, perhaps a more general set of classifiers, performing well for a variety of combination methods, could be discovered to speed up the process of building a classifier combination system. The concept of *classifier diversity* is often used to describe the "difference" between classifiers, but it cannot be strictly defined, and a multitude of measures have been presented. Classifier diversity and its effects for combining classifiers are also addressed in this thesis. A classification-performance-oriented approach to defining classifier diversity, *diversity of errors*, is examined and some measures based on this principle are presented.

The substance of this work is in the presented novel methods and empirical tests that show their prowess. Deep theoretical analysis of the presented combination methods would also be an issue very much worthy of attention. However, the proposed combination methods are significantly complex in nature, and theoretical analysis on their effectivity could well prove extremely challenging. Simplifying assumptions that would make the analysis feasible could also make the analysis quite valueless with respect to real world performance. This is an issue with nearly all the more complex combination methods, and the adaptive nature of

the methods causes further complexity. A more analytical investigation into the behavior of adaptive committee classifiers is still a topic that would be a very interesting subject for future work, but such discussion will not be in the scope of this thesis.

1.2   Contributions of the thesis

In this thesis, a novel framework for adaptive combination of classifiers is suggested. The field of the thesis will be first introduced via a literature survey to form an overview of classifier combination methods particularly suitable for such a setting where user-specific adaptation can be beneficial.

The most important single methodological contribution of this thesis is *a novel adaptive framework for combining classifiers*. The proposed Class Confidence Critic Combining (CCCC) scheme can be used to combine a wide variety of classifiers, adaptive or non-adaptive in themselves, in a way that provides user-dependent on-line adaptation of the system. Several *methods for individual classifier adaptation are discussed and compared*. These methods have been implemented in the CIS-HCR adaptive on-line handwritten character recognition system and used for both member classifiers of an adaptive committee and to evaluate the benefits of adaptivity on the committee level. Also *several other new adaptive classifier combination techniques* are introduced and experimented with. These methods are the novel Dynamically Expanding Context (DEC) committee, the novel Modified Current-Best-Learning (MCBL) committee, an adaptive implementations of the Behavior-Knowledge Space (BKS) and Decision Template (DT) committees, an adaptive voting committee and an adaptive best selection rule. A thorough evaluation of the performance and the benefits obtained via the presented adaptive classifier combining methods can be found in the included publications.

As classifier combining is also highly dependent on the classifiers being combined, a look at the methodology for selecting classifiers for combination is presented. It includes both an overview of existing classifier diversity methods and the introduction of a *novel classifier diversity measure* that has been found effective. Also some *rejection methods applicable for a committee classifier*, adaptive or non-adaptive, are presented along with experimental results supporting their effectivity.

A smaller, but also important topic is establishing the reasonability of the fundamental idea employed, on-line adaptation. If it is not possible to implement

a system where adaptation is possible without explicit label information, which is commonly available in batch experiments, then what is the practical value of such methodology? It is clearly necessary to illustrate the feasibility of on-line adaptation in a handwritten character recognition system without direct feedback from the user. In practice the problem is how to obtain correctness information from actual usage, without knowing the true intended class of a sample. A view on this issue is also presented in the form of a *set of label deduction rules* that has been implemented within an application developed for a Personal Digital Assistant (PDA) system. No explicit evaluation of the performance of the label deduction rules has been performed. Still, the efficiency of the adaptation when using the proposed set of rules does clearly suggest that the rules are capable of obtaining a level of labeling correctness that is sufficient for highly beneficial adaptation.

1.3 Outline of the thesis

After this introductory chapter, in Chapter 2 a literature survey on related research is presented. The literature survey is divided into sections, of which the first explores handwritten character recognition, placing extra emphasis on approaches to adaptation. The latter part focuses on methods for combining classifiers, again with special emphasis on on-line adaptive approaches in accordance with the theme of this thesis.

Next, to set the ground for the practical application of on-line adaptivity, Chapter 3 explores the feasibility of implementing on-line adaptation without explicit information on true labels for the classes. Also one method for attaining that goal is presented.

In Chapter 4 methods for individual classifier adaptation as implemented in our CIS-HCR system are examined. The methods are discussed more to illustrate the differences in adaptivity between classifiers and committees, and are, for the most part, the work of other members of our research group. Classifiers presented in this chapter are also used as member classifiers in the committee experiments.

The focal part of this thesis, the methods for adaptive combination of classifiers, are presented in Chapter 5. The developed methods are described and an extended comparison of the performance of the committee classifiers used in this work and the included publications is shown.

The following two chapters describe two approaches for further improving classifier combination performance. The first approach presented in Chapter 6 is committee rejection, to which goal several methods are presented. The second, discussed in Chapter 7, is an approach to measuring classifier diversity for selecting member classifiers for combining, based on the idea of the importance of the diversity especially among the errors made.

Finally, Chapter 8 gives the final conclusions of the work presented in this thesis. This is followed by the references and a set of publications detailing the proposed methods and showing the results of experiments.

1.4 Included publications

The following three journal articles and five conference papers have been included in this thesis.

1. Vuokko Vuori, Matti Aksela, Jorma Laaksonen, Erkki Oja, and Jari Kangas. Adaptive character recognizer for a hand-held device: Implementation and evaluation setup. In *Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition*, pages 13–22, 2000.

2. Matti Aksela, Jorma Laaksonen, Erkki Oja, and Jari Kangas. Application of adaptive committee classifiers in on-line character recognition. In *Proceedings of International Conference on Advances in Pattern Recognition*, *Lecture Notes in Computer Science*, 2013:270–279, 2001.

3. Matti Aksela, Jorma Laaksonen, Erkki Oja, and Jari Kangas. Rejection methods for an adaptive committee classifier. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 982–986, 2001.

4. Matti Aksela, Ramūnas Girdziušas, Jorma Laaksonen, Erkki Oja, and Jari Kangas. Class-confidence critic combining. In *Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition*, pages 201–206, 2002.

5. Matti Aksela, Ramūnas Girdziušas, Jorma Laaksonen, Erkki Oja, and Jari Kangas. Methods for adaptive combination of classifiers with application to recognition of handwritten characters. *International Journal of Document Analysis and Recognition*, 6(1):23–41, 2003.

6. Matti Aksela and Jorma Laaksonen. On adaptive confidences for critic-driven classifier combining. In *Proceedings of International Conference on Advances in Pattern Recognition*, *Lecture Notes in Computer Science*, 3686:71–80, 2005.

7. Matti Aksela and Jorma Laaksonen. Using diversity of errors for selecting members of a committee classifier. *Pattern Recognition*, 39(4):608–623, 2006.

8. Matti Aksela and Jorma Laaksonen. Adaptive combination of adaptive classifiers for on-line handwritten character recognition. *Pattern Recognition Letters*, 28(1):136–143, 2007.

1.5 Contribution of the author in the publications

In conference paper 1 a PDA implementation of an adaptive recognizer is presented. The system employs a questionnaire scheme, where the correctness of each recognition result is deduced from the user's following actions, without any direct feedback as to the correctness of the classification. As the first co-author of that paper, the current author devised the correctness feedback logic in cooperation with the first author and second co-author of the paper, implemented all the required methodology in the hand-held device and programmed the questionnaire program. The first author devised the adaptive classifier itself and performed the experiments with the constrained character matching algorithm, and made a tool for analyzing the results of the user experiments. The first author was mainly responsible for the reporting, but all authors participated in the reporting.

Conference paper 2 examines the behavior of the DEC committee and performance improvements obtainable. The author was responsible for the implementation of the classification system and experiments, but the original idea of the adaptive DEC committee was introduced by the first co-author. This paper also introduces the Modified Current-Best-Learning committee, which was a development of the author. All co-authors participated in the reporting.

In conference paper 3 several committee rejection schemes are examined with application to the DEC adaptive committee classifier. The author devised and implemented the rejection schemes and performed the experiments, and the co-authors participated in the reporting.

Conference paper 4 introduces the CCCC committee and provides the first experimental results to show its prowess. The CCCC committee was an original idea of the author, who also implemented the committee and performed the experiments. The second author implemented the SVM-based member classifier used in the experiments. All co-authors participated in the reporting.

Journal article 5 provides a thorough overview into several methods for adaptive combination of classifiers. The performance of the methods were evaluated and compared to several reference committee classifiers. The CCCC and DEC committees were found to perform best. The author devised and implemented the committees and performed the experiments. The second author implemented the SVM-based member classifier used in the experiments. All co-authors participated in the reporting.

Conference paper 6 introduces weighting schemes and a novel way of evaluating the confidences to the CCCC scheme. It was found that the weighting schemes enable robustness in the face of changing circumstances, which is in this paper examined with experiments that do not take advantage of user-dependent resetting of the committee. The author devised and implemented the weighting schemes and improvements to the CCCC committee and performed the experiments. The co-author participated in the reporting.

In journal article 7 several diversity measures are evaluated and their effectiveness for choosing the best possible set of classifiers for both non-adaptive and adaptive classifier combination schemes is examined. The novel exponential error count diversity measure introduced was found to be very effective. The author devised the novel "diversity of errors"-based schemes and implemented them and the reference schemes examined. The author also introduced the taxonomy of diversity schemes presented. The second author participated in the reporting.

Journal article 8 presents the culmination of our work in the sense that this paper introduces a formulation of the CCCC committee classifier that is found to work successfully within the setting of adaptive combination of adaptive classifiers. The doubly adaptive scheme is capable of improving on its adaptive member classifiers' performances. The author devised the necessary improvements to the CCCC framework to obtain the robustness for this setting, implemented the committee and performed the experiments. The second author participated in the reporting.

Chapter 2

Related research

In this section an overview of methodology related to the three main themes of this thesis, on-line adaptivity, classifier combining and on-line handwritten character recognition, is presented. However, the combined fields of handwritten character recognition, classifier combining, on-line operation and adaptivity are far too large to handle in their entirety within the scope of this thesis. Thus this discussion will focus on methods with applicability to a combination of these subjects.

This discussion is divided into two main categories. First in Section 2.1 the topic of handwritten character recognition is discussed, introducing some classification approaches while focusing on on-line methods. Special emphasis will be on methods that allow implementation of adaptation. The possibilities for adaptation are discussed in more detail in the following Section 2.2.

Then, the most central theme of this thesis, the topic of classifier combining is examined in Section 2.3, again focusing on methodology related to the focal themes of this thesis. Adaptive committee methods found in the literature, or methods that could easily be implemented in an adaptive fashion, are discussed in Section 2.4. The brevity of that section is indicative of the immaturity of the field and the necessity for more research on the topic.

The purpose of this review is not to present every distinct methodology that has been used for the studied tasks, or even every more general group of approaches, but to focus on the most important viewpoints on the subjects and present representative examples of recent research directions in the respective topics. The

discussion on actual results presented in various papers has been completely left out, as comparing results over different databases, settings and even character sets in some situations will not reveal much about the true relative effectiveness of each method; it would potentially just distract from the main focus of the investigation. For the results obtained in the experiments with each respective method the reader is directed to the references.

2.1   On-line handwritten character recognition

Handwritten character recognition has been an ongoing research problem for several decades [197, 113, 155, 15, 220]. One main division to be made within the field is in off-line versus on-line recognition of handwriting. Basically, off-line recognition means that the recognition is not performed during writing, as opposed to on-line recognition where the input is recognized immediately, i.e. on-line. In practice, data from on-line experiments can be stored in a variety of formats especially designed for on-line data collection, for example UNIPEN [65]. Such formats include integral elements of the writing process, such as the direction and speed of writing, in the stored data and hence the data can later be used as if it were being collected during recognition. Thus the actual methods of recognition differ more on the level and type of data they use than whether the actual recognition process is done on-line.

However, there is one additional distinguishing feature between on-line and off-line recognition: the possibility of interaction between the user and the system. In on-line recognition it is possible for the system to provide interactive features such as error correction or other methods of obtaining more information from the user to aid in the recognition task. The possibility of interaction can in fact make the entire setting for on-line recognition notably different from that of off-line recognition, where all available information is in the samples to be processed.

The practical distinction is often that off-line recognition works with data from images of written characters [197], whereas on-line recognition uses information collected during the writing of the characters, that is to say, information on how the characters were written and not just what the final result is. One example of how this is often implemented is via collecting the actual pen trace as a list of (x, y) coordinate pairs, possibly adding other information such as pressure and time stamps, as has been done in the recognition system described in Chapter 4. One might even say that it is more a question of off-line versus on-line data collection that makes the distinction; off-line type data could well be recognized

immediately as it comes in as well as on-line type data stored for later processing or future experiments. But still, to use the common convention, the terms off-line and on-line recognition will be used.

As an enormous amount of written documents exist, some dating back hundreds of years, it is evident that if it were possible to reliably and automatically index and store any written material, this would have great benefits with regard to both keeping all that information safe and making it more accessible. The value of off-line handwriting recognition is evident also in, for example, automatic sorting of mail and scanning documents to a computer readable form, just to name a few examples. Commercial systems for off-line recognition have been available since the middle of the 1950s [63, 187].

In contrast to off-line recognition, where quite little information is available to the system from how the characters were actually written, on-line recognition is generally performed by collecting information on the pen trace in some manner. This generally requires specific writing equipment to be used, which indicates that the writing is intended to be used with computers. Actual on-line recognition also provides the possibility of writer interaction, which in turn enables more efficient mistake correction and adaptation of the system [197]. Of course, also methods combining approaches from both off-line and on-line fields can and have been used [64, 194]. From hereon, the discussion will focus on on-line handwriting recognition.

2.1.1   Problem description

One major reason for the interest in on-line handwriting recognition is the perpetual search for effective input methods for modern-day hand-held devices, such as PDAs and mobile phones. In general, the display quality of hand-held devices has improved drastically in the recent years. Despite the constant desire to keep devices small, devices with 65 thousand colors and a resolution of 160×128 pixels and above are currently commonplace even in the lower price range mobile phones [143, 173, 137]. And surely the progress will only continue – even better quality displays will emerge in the same small space.

However, the input method is one part of the system where the small size can cause problems. The keyboard, to be a practical method for input, needs to be large enough for the keys to be pressed without excessive effort. The implementation of a regular keyboard of that size is not feasible for a hand-held device, as it would require the devices to be larger than otherwise desired. Devices that do successfully incorporate the keyboard, such as currently the Nokia Communica-

tor series [143], are much larger in size and appeal to the minority segment of the user base that places efficiency of operation and features over mobility and size. In practice the limit is that if the size of the keys becomes smaller than the size of the fingertips, the usability of the keyboard deteriorates rapidly.

There have been numerous input methods introduced with variable success, for example virtual keyboards on-screen – which have been found to produce reasonably good accuracies but very slow input rates [125] – and specialized character sets for recognition, such as the unistrokes alphabet [62] and its commercial derivation, Graffiti by Palm Computing [132]. The problem with specialized alphabets is the need for the user to learn an entirely new character set in order to interact with the machine, which raises the bar for starting to use such an interface. Therefore, perhaps the most natural and effective way of input in a hand-held device is the use of effective and adaptive handwriting recognition. The users could evoke an input method they are familiar with, and there would be no need for additional space, as a pressure-sensitive screen could be used also for input. The inclusion of on-line, transparent, adaptation into the recognition process may be the deciding factor for usability, as this would allow the user completely natural input and continuously better performance.

Of course, even though the fundamental problem remains the same, there are significant differences caused by the type of character set used. For example Chinese and Japanese Kanji characters consist of a much larger number of strokes than the Latin alphabet. Also, the Eastern alphabets consist of several thousands of characters. They thus provide challenges very different from the recognition of a subset of the Latin alphabet such as numbers with only ten possible classes. The work described in this thesis has focused on using the Latin alphabet, including the Scandinavian diacriticals, and the problem of handwritten character recognition will be examined from this viewpoint. Of course, for the classifier combination aspect, the actual symbols being recognized bear much less significance and the combination methods can be applied over a much wider range of situations than the individual classifiers used.

In general, the pen is a very natural way for humans to input information, since writing is generally taught in schools and learnt at a young age. Also for reasons based on human physiology, a pen is a very effective and precise method for input – for humans positioning the pen-tip is highly accurate due to the high number of degrees of freedom provided by the fingers and the relatively large portion of the motor-control areas in the brain dedicated to finger movement [178]. This can clearly be observed by comparing the precision of input while drawing with a pen in comparison to drawing with a regular computer mouse. Movement originating from the wrist is much more coarse as is also the quality of the final output.
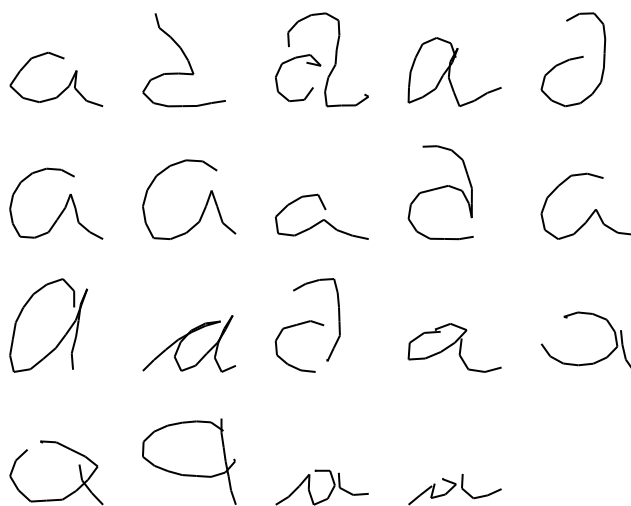
Figure 2.1: Examples of ways of writing the lowercase letter 'a'

At first glance handwritten character recognition using the Latin alphabet and recognizing characters one at a time may not seem like that difficult a problem. Basically all that is needed is to match the input character to a set of known characters, which may be stored as, for example, prototypes or model coefficients. Also the set of characters that can be input is limited to the alphabet in use. The main problem with unconstrained handwriting recognition arises from the fact that each writer has his or her own personal writing style, resulting in a huge amount of variability in the ways each character can be written. This in practice makes storing all variations, known as allographs, impossible [95]. Figure 2.1 shows just a few examples of ways of writing a lowercase 'a'. Studies have shown that numerous reasons for using a particular allograph can be identified, such as positional, contextual, dialectical, and stylistic reasons [68]. But it has also been discovered that generalizations to other allograph styles from a subset of samples is not very reliable with a small number of collected samples [217, 212, 209]. In practice this means that the way some other character will be written cannot reliably be predicted from characters of known classes.

Taking a slightly different approach, also models focusing on the variability in handwriting have been presented [219, 94, 84]. The focal idea of these approaches is to focus on what kind of variability is possible and how to incorporate that information into the model in a way that makes generalizing these variabilities over a number of models possible. However, such approaches are quite rare. It

may be that enumerating variability is in fact not a very simple task in itself. Also the forms of variability can vary between character classes, which can lead to problems if one tries to apply a more general model of variability.

Common practice is to train a recognizer with data collected from several writers thus introducing several writing styles to the system expecting generalizational ability. Then, if the writing style of the new user is similar to some style in the training set, the performance of the system for that writer can be adequate. However, the system will certainly perform poorly for writers whose styles are not in the training set – and due to the vast amount of variation, it is in practice impossible for all writing styles to be accounted for. This may lead one to the conclusion that in order to obtain satisfactory recognition accuracy with all users without constraining the allowed style of writing, one of the most practical approaches is to make the recognizer adaptive, i.e., it has to be able to learn and adjust itself to new writing styles.

2.1.2   Classification approaches

Some of the classification approaches that have been commonly applied to on-line handwritten character recognition will be discussed next. Not surprisingly, a vast amount of classification methods can be found in the literature, and they differ greatly in implementation, complexity and sometimes also effectiveness. It should also be noted that there is no single method that is known to generally perform best. The choice of optimal classifier can therefore be confidently said to depend on the task the method is to be applied to.

All classification methods have basically the same objective, to minimize the overall cost of the decisions – taking into account both correct and incorrect decisions. And since this can be viewed as being a basic property of a Bayesian decision system, one might argue that all the decision boundaries implicitly or explicitly defined by various classification approaches can be viewed as approximations of the optimal decision boundary of a Bayesian classifier. Whatever the viewpoint taken may be, the statistical reasoning certainly has fundamental value for understanding classification systems. Generally, whenever attempting to classify a sample we tend to ignore assigning specific failure costs and simply strive to choose the most likely class, the one that has the highest posterior probability. How we reach that decision, i.e. whether the approximation for that probability be directly from an application of the Bayes rule, closeness of matching to a prototype, the relative values of outputs of a neural network or distances from decision boundaries defined by a set of parameter-space vectors, is where the

classification methods differ.

All in all, any strict grouping of classification strategies is somewhat arbitrary, as the definitions of what they entail are in many cases overlapping. Some strategies could be viewed in light of both statistical and structural approaches. Prototype-based approaches commonly use some decision mechanism such as the nearest neighbor rule, which on the other hand can also be seen as a non-parametric statistical method. Thus also the grouping applied in this thesis represents only one view – and in the end, it perhaps does not really matter how each method is categorized, or if they are even grouped distinctly at all. In any case, the fundamental objective of all classification methods is the same: to correctly recognize the input. But in hopes of easier reading, one possible taxonomy has been used and methods have here been labeled under the six different categories presented below.

Statistical methods

Statistical classification approaches, by definition, take the viewpoint of assuming that the variation is of stochastic nature. Thus statistical methods more or less explicity attempt to estimate, for the input x and class c_i, class prior probabilities $P(c_i)$, and probability densities of the samples from each class $P(x|c_i)$. Bayesian decision theory is then used to obtain a prediction on the posterior probabilities $P(c_i|x)$,

$$P(c_i|x) \propto P(x|c_i)P(c_i), \tag{2.1}$$

where $P(x)$ has been left out as the equation is generally only used when observing x.

In practice, statistical methods can be divided into two categories, parametric and non-parametric. Parametric methods estimate model parameters from training data, with perhaps the most straightforward case being to use a simple distribution model for all classes, for example a Gaussian distribution [123, 14]. Non-parametric methods, on the other hand, make no assumptions on the distribution model, but attempt to use the training data directly. This can be done for example through the use of histograms or kernel functions.

One of the most common decision rules, the k Nearest Neighbor (k-NN) rule [52, 37], can be seen as a non-parametric statistical classification model. In the k-NN rule the distribution is implicitly defined by all the samples used in the set where the rule is applied. The k-NN rule has been found to be very accurate with large databases [186], although the time needed for operation is directly proportional to the number of models the input is matched against. Figure 2.2 illustrates the
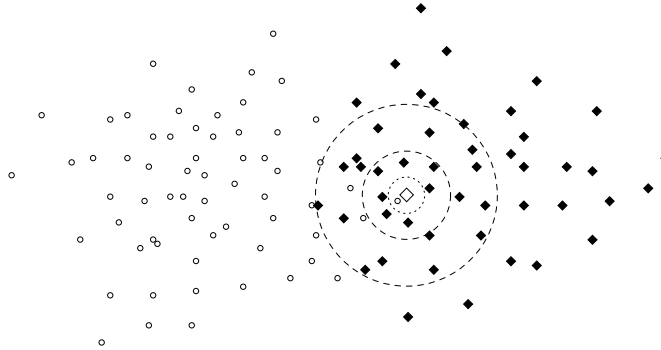
Figure 2.2: An illustration of the k-NN algorithm

k-NN rule and the benefit of using more than one neighbor; if the classification were done to just the nearest class, as illustrated by the dotted circle with the smallest radius around the input, the input marked by the diamond would be classified to the class with circle markers. Increasing the value of k, as marked by either dashed circle, would clearly direct the classification to the more appropriate class, marked with the filled diamonds. Several modifications of the k-NN algorithm have been presented to improve the operation speed and performance, for example the weighted prototype editing algorithm which calculates weights for the prototypes to enable pruning to improve performance [150].

Non-parametric statistical classification methods are often well-suited for user-specific adaptation, as the distribution is defined by the set of models, and altering a particular model can effect the implicit distribution without the need for extensive recalculations. On the contrary, when information on the distribution is collected into a more formal model, where the model parameters define the output and individual samples are not as easily taken into account, adaptation may not be as easy or effective to implement as with non-parametric models.

The concept of the Hidden Markov Model (HMM) was introduced in the late 1960's, but has become increasingly popular since the 1980's. The HMM approach stems from using a Markov model where the observation is a probabilistic function of the state, i.e. the model is a doubly embedded stochastic process with an underlying process that is not observable, but hidden, and hence the name [160]. The training time for an HMM model is usually rather long, and thus also adapting the models is often harder than for prototype-based classifiers.

In most cases a separate HMM is created for each allograph (a model for a way of writing a character) sufficiently represented in the training set [142, 21]. The

most commonly used HMM type is the continuous HMM [24]. Especially in small databases the interpolating effect of continuous HMMs can be very beneficial [167]. However, also discrete HMMs have their benefits, as the discrete parameterization helps reduce the amount of training data needed, but at the cost of loss of information during the quantization step [21].

As with speech recognition, also in handwriting recognition HMMs have been used in conjunction with language models to improve performance [74]. Some recent approaches to HMM-based recognition have used both pen direction and non-stationary pen coordinate features [148] and HMMs for cluster modeling [17]. Some variations on the basic HMM models are fuzzy HMMs [115] and applying a maximum mutual information optimization scheme with tied mixture density HMMs [144].

Prototype-based classification

One simple but often very effective approach to classification is to match the input sample against a set of prototypes whose class is known. Then the task is to determine the best match using some measure and decision mechanism, with the matchings acting as a basis for deciding the class of the input.

A significant concern in prototype-based approaches is the proper selection of the prototypes. There are some fundamental characteristics that a good set of prototypes should abide by: a prototype set should have sufficient coverage, reasonable separation, each prototype should be a good representation of a way of writing a character, and poorly representative prototypes prone to causing errors should be avoided [13]. Of course sufficient coverage poses a problem, as ideally this would mean having a prototype for each way of writing a character, which often is not feasible due to the vast amount of possible ways of writing [95]. Thus, the design of the prototype set is a task whose success has a very significant effect on the final performance, and construction of the prototype set can be viewed as an important research problem in itself [185, 35, 119, 161]. Intuitively one would desire prototypes that represent distinct ways of writing a character, as depicted by Figure 2.3 where three very distinct ways of writing the capital 'A' are shown. In practice with many classes and large amounts of data, the task is far from simple.

When using prototypes for matching against, some rule for choosing the correct class must naturally be employed. Perhaps the most commonly applied decision rule for matching prototypes is the k Nearest Neighbor (k-NN) rule [52, 37], which has already been discussed above in the section on statistical methods. In
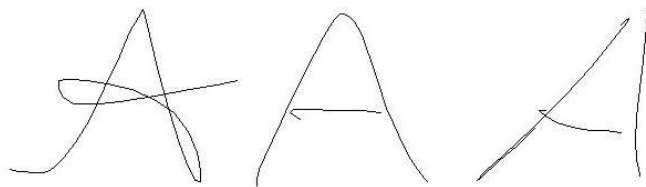
Figure 2.3: Three examples of distinct ways of writing the letter 'A'

its simplest form, a single nearest neighbor classifier, the class of the prototype that is closest to the input, according to the given distance measure, is chosen for the class of the input sample. It is easy to see why prototype-based systems are commonly employed in situations where adaptation is desired – adjusting the prototypes directly influences the classification results, and prototypes can be effectively adjusted based on even a single input sample.

There is one significant division to be seen among prototype-based methods; approaches using prototype vectors of fixed length and approaches using variable-length prototype vectors. Fixed length sample vectors are commonly obtained whenever calculating some kind of a vector of features extracted from the input. Prototypes of variable length, on the other hand, are often the result of using the data points collected for representing the samples, as the number of data points will vary between characters.

One approach for obtaining fixed-length vectors is just to resample the data to have a constant number of points and use them as the representations for the character samples. In such a setting, several types of distance calculation methods can be used, starting from simple Euclidean distances [153] to more elaborate metrics like the tangent distance [183, 150]. Another commonly used approach for obtaining fixed-length prototypes is to calculate some set of features from the input data. This is, however, more common in off-line recognition. Storing the dynamic information, speed, distance and ordering, of the points is natural when using the input coordinate streams as the representations for the characters. As a result, one often uses all available data points, which results in variable length samples. Also features dealing with directional attributes are often used [218, 145], and discussion on such approaches can be found below under structural methods.

With variable-length vectors, distance computation approaches such as elastic matching are very useful for handwritten character recognition. The need for variable-length matching techniques is evident because the number of points may

vary depending on the speed of writing, size of the characters and other similar factors. One approach to prototype-based character recognition is to use Dynamic Time Warping (DTW) for matching of elastic templates [196].

In time warping approaches a sequence of metric points of the input character is matched against those of the prototypes. The overall distance between a prototype k and the input x, can be obtained as [101]

$$D_{\text{DTW}}(k, x) = \min_{w(i)} \sum_{i=0}^{N} d(i, w(i); k, x), \qquad (2.2)$$

where $w(i)$ is a warping function that maps the time index of the input model to that of the prototype and d is the distance function. Equation (2.2) can be solved with dynamic programming by using the recursion relation [101]

$$D(i, j; k, x) = d(i, j; k, x) + \min \begin{cases} D(i-1, j; k, x) \\ D(i-1, j-1; k, x) \\ D(i-1, j-2; k, x) \end{cases}, \qquad (2.3)$$

where $D(i, j; k, x)$ is the cumulative distance to the prototype k up to the point (i, j) and $d(i, j; k, x)$ is the distance between points i and j of prototype k and input x. The definition of the cumulative distance function $D(i, j; k, x)$ is the integral point of interest for determining the effectivity of the matching. Some possibilities will be discussed in Section 4.4.1.

Several variants of different types of elastic matching have been researched. These include an elastic distance using a centroid computation algorithm [11], optimal warping of models stored using overall symbol measures such as angle and normalized height from a baseline [101], elastic matching with only one prototype per symbol [124], warping subject to two-dimensional monotonicity and continuity constraints [200], and the use of a linear combination of eigen-deformations to alter the prototype [201].

Neural networks

Neural networks (NNs), or artificial neural networks (ANNs), are methods commonly applied in a multitude of pattern recognition tasks – including handwritten character recognition. Neural networks are systems whose fundamental idea is analogous to the operation of the human brain, which is a highly complex, nonlinear and parallel computer. The human brain, and neural networks modeled in its image, consist of many simple processing units (neurons) which are combined

to form large networks that together can perform operations far more complex than individual neurons. Generally it could be said that the neural network stores information collected in a learning phase into interneuron connection strengths, known as synaptic weights. The network has intrinsic generalizational ability, meaning that it can produce reasonable outputs also for inputs not encountered in learning. The network constructs a mapping between the input and output spaces that can be highly non-linear. As a "black-box" type of classifier neural networks have been used for a wide variety of tasks, including but not limited to prediction and classification [67].

A simple mathematical expression of a neuron may be obtained via writing y_k, the output of neuron k, as a function of its m inputs x_i, bias term b_k and activation function θ. Common examples of activation functions are simple threshold functions and sigmoid functions such as the hyperbolic tangent function. Each connection is defined by a synaptic strength w_{kj}, in practice weights for the inputs. Thus the output of the neuron can be written as

$$y_k(x_1, \ldots, x_m) = \theta(\sum_{j=1}^{m} w_{kj} x_j + b_k). \qquad (2.4)$$

In practice, this "black-box" nature of neural networks means that the internal workings of a network are hard to trace. A notable benefit is that NNs can often be used without separate preprocessing of the data [139, 229]. This means that the network can perform all stages of the pattern recognition process, from feature extraction to delivering the final output. But the "black-box" nature is not entirely advantageous, as a lack of explicit knowledge on the stages of the recognition process also makes it much more difficult to incorporate *a priori* knowledge on the application domain and to break down possible sources of errors. Also the need for a very large training set is a notable drawback to NN use – this makes especially adaptivity much more challenging to implement in pure NN systems. Adaptation in an NN would require adjusting the network parameters, which is much harder to implement effectively and reliably when using only a few incoming samples.

Perhaps the most basic and best-known NN method is to use Multilayer Perceptron (MLP) networks and some variant of the Back-Propagation (BP) algorithm for training. The original BP approach is not very often used anymore since it suffers from a slow convergence rate [12]. The original BP algorithm can be considered practically obsolete, and several NN schemes allowing for faster training and better generalization results have been proposed, for example a supervised feed-forward fuzzy neural classifier used for recognition of alphanumeric characters [12], Bayesian decision based neural networks with Chinese characters [57],

enriching artificial neurons with spatio-temporal coding for on-line handwritten character recognition [139] and using an Adaptive-Subspace Self-Organizing Map (ASSOM) for handwritten digit recognition [229]. Also approaches for selecting features that enable simplification of the network structure have been found to be effective [188].

Another viable neural-net-based approach is to use a sub-sampled Time-Delay Neural Network (TDNN), a special type of temporal convolutional net [112]. Additionally, also a Space Displacement Neural Network, and a combination of these, a Space Displacement Time Delay Neural Network, have been successfully used for on-line character recognition [157]. Neural networks have been widely studied in a number of contexts. All in all, a vast amount of both different NN methods and possible application areas exists and only a few examples were covered here.

Feature space methods

Feature space methods, by definition, are approaches that base their operations on the idea of using feature extraction to reduce dimensionality and represent the data as vectors in a multi-dimensional feature space. This is done with the intention of being able to use a feature space where the classes in the data will be easily separable. This section will focus on methods attempting to define explicit decision boundaries in that feature space. Thus methods operating directly on intrinsically multi-dimensional data, without any feature-space transformations, such as featureless kernel-based methods, are examples of other methods to be discussed later.

Naturally, the success of the feature space methods depends greatly on the features and the success in their extraction – if good features are found, the classes may form separable clusters in the feature space, allowing for successful and efficient recognition with less complex classification rules. A notable problem is the ever-present "curse of dimensionality" [22] – as the dimensionality grows, the volume of the space grows exponentially, hence in practice requiring more and more extrapolation from a learning algorithm. Also along with the increase of dimensionality, as the data becomes more sparse, many algorithms have a tendency to over-fit the training data, again resulting in a loss of generalizational ability.

Perhaps the best-known feature space method is the Support Vector Machine (SVM) approach. SVMs have been widely used in a multitude of classification tasks. The concept of support vector classifiers is based on the statistical learning
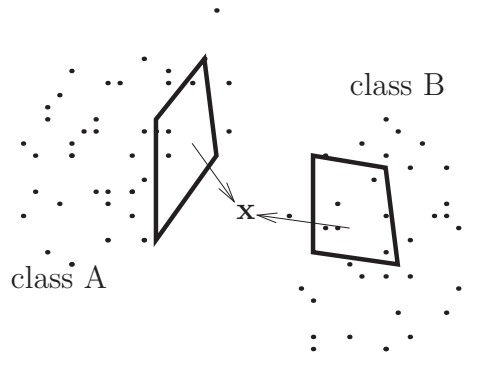
Figure 2.4: An illustration of the LSC principle

theory presented by Vapnik [205, 204] and optimization with quadratic programming. The basic idea of an SVM classifier is to define the decision boundary in the high-dimensional feature space with the support vectors – hence the name of the approach. The SVM approach has been shown beneficial, albeit computationally expensive, especially when dealing with high dimensionality combined with small sample size [120].

As the SVM approach is computationally quite intensive, a number of effective methods for speeding up SVM training with very large handwritten character datasets have been presented in e.g. [41]. Recent examples of studied SVM approaches include using feature extraction wavelet transform and elastic meshing and pair-wise SVMs for Chinese character recognition [49]. Other recent studies have proposed speeding up SVMs using the k-NN algorithm and Manhattan distance applied with respect to handwritten digit recognition [127] and using a Gaussian DTW kernel for SVMs in on-line handwriting recognition [18].

In another viewpoint on feature-space-oriented classification, specialized subspace classifiers [147, 106] have been used with feature vectors of constant dimension. The characters are recognized with methods which were earlier found to be effective for off-line recognition of numerals written on paper [103]. As the particular neural-type classification algorithm an adaptive version of the Local Subspace Classifier (LSC) is used. An illustration of the principle of the LSC for a two-class case can be seen in Figure 2.4. The classification of the input $\mathbf{x}$ is performed based on the distances, shown by arrows, to the local subspaces spanned by prototypes in the two classes [103].
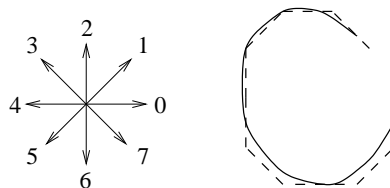
Figure 2.5: An example of 8 directional chain code and the character 'C' along with its Freeman's code word 34566701

Structural methods

The basic idea of a structural recognition system is to consider the input as consisting of several structural components that are joined in some way. In handwriting recognition, these primitives can be anything from segments of directional chain-code to very elaborate structures depicting simple handwriting elements. Structural recognition is fundamentally a top-down divide-and-conquer strategy, something that has also been considered closest to the intuition of humans, which can be considered a merit as humans clearly possess the best pattern recognition skills when dealing with handwriting recognition [218].

Syntactic methods are often used for parsing together the whole input from primitives and performing the matching. Some simple approaches for describing the structure connecting the components include Picture Description Language (PDL), tree grammars and array grammars [218].

One way to perform structure-based recognition is to define a group of entities which are first matched to the input, and then describe the order of these entities, resulting in a sequence of expressions. Simple realizations of this idea are for example chain-coding approaches, such as Freeman's code and its extensions and modifications [218]. An illustration of a simple chain-code with eight directions can be seen in Figure 2.5. One example of using a chain code is an adaptation of Freeman's code with 12 evenly spaced directions and an additional scheme for coding pen positions [145].

The actual difference calculation between the input and the primitives must of course also be performed. Due to variations in handwriting, finding absolute matches is in many cases impossible, and thus several matching schemes which allow variation have been developed. Two main approaches are elastic structural matching schemes [30, 83, 182] and deforming models [51, 32]. Also the HMM models, previously discussed in conjunction with statistical recognition, can be

used for representing some parts of the input in a structural manner. Naturally also prototypes may well be used as models of subelements of the handwriting, and similar matching methods be applied.

A different take on structural characteristics is to use structural features of the entire sample, such as horizontal, vertical and radial histograms [80]. Yet another recently published strategy uses a character representation based on joint distributions of the strokes and applies a form of neighbor selection to establish the structure of strokes [86]. Also methods based on detecting core points for structural decomposition have been proposed [190], where the core points are defined as the minimal number of designated segmentation points required to obtain the segmentation. Yet another example of recent structural recognition approaches is one that uses direction and transition features for its feature representation [121].

Other methods

Only a very brief survey of some of the fundamental and frequently used classification methods relevant to our work has been presented above. Of course, a vast array of other algorithms and methods have been applied also in the setting of on-line handwritten character recognition. Two examples of other methods with a notably different viewpoint on the recognition process include genetic algorithms and generative models.

Genetic algorithms (GAs) are modeled after the process of evolution, trial and error and survival of the fittest solutions. GAs can be applied also to handwritten character recognition, and are especially well suited for the feature-selection process. Genetic algorithms have some notable beneficial characteristics that make them very attractive for example for adjusting weights. Namely, genetic algorithms can operate with a set of solutions, are not based on derivatives and are effective in exploring and using the parameter space [136]. A variant of the Simple Genetic Algorithm has been successfully used for feature selection and weight adjustment in [85]. Genetic algorithms have been used to adjust weighting parameters for different groups of features in [29]. Also on a less handwriting-specific, but nonetheless very interesting technique, a combination of genetic algorithms and the expectation maximization (EM) algorithm for a setting of Gaussian mixture models was applied in [154]. The results showed that using a Minimum Description Length (MDL) criterion the proposed GA-EM algorithm explores the space more throughly than a standard EM algorithm and is thus less likely to stick to local minima.

Generative models represent a slightly different top-down viewpoint on the recognition problem, where the recognition process starts from a model to generate the output. The approaches that can be labeled under generative models encompass a wide array of different techniques, but since the basic approach differs notably from the one used in our research, this viewpoint will not be discussed in detail. As practical examples, generative models have been presented for a Bayesian framework [32], for deformable B-Splines with ink generators [166] and for a model based on kinematic theory of human movements [156].

2.2 On-line classifier adaptation

Since there exists practically an unlimited number of ways of writing each character, it is highly impractical, if not impossible, for a recognition system to store a model for every possibility. Still, the recognition system should perform optimally for the person that is using it, and without any prior knowledge on what style of writing the user will use. One fundamentally sound approach to solving this issue is to have the classifier adapt to the particular user.

In general, adaptation to the classification task at hand can be a very effective method for performance improvement. This is especially true when a high level of intrinsic variation in the input data exists, but a substantial part of the variation can be explained by some underlying process or phenomenon. Handwritten character recognition, as well as for example speech recognition, is such a task. In handwritten character recognition the data from different users in general varies greatly, but each user has a style that is reasonably consistent. It is this consistency of each user's particular style of writing that can be learnt by starting with a user-independent system which is then adapted for optimal performance for a particular subject.

Classifier adaptation can be performed either through the use of a separate learning phase before regular use of the device or during it. In practice the former means teaching the system how you write during a specialized training phase before starting to really use the device. While in general least prone to errors, such a policy of adaptation is not very user-friendly, as time must be spent before using the device. Also, if only a separate training phase is used, the system will not be able to adjust to subtle changes in style, but always requires complete retraining. We refer to this as off-line learning to illustrate that the adaptation is not performed during regular use. This is in fact just training the classifier further with the particular user in an additional training phase.

The alternative is on-line adaptation, or learning during use. This is an attractive approach due to its un-invasiveness and transparency to the user. If the true labels of input can be deduced without explicit feedback from the user, this information can be used to refine the performance of the system while it is being used. This also enables the system to slowly adapt to changes in the style of the writer.

One significant issue with on-line adaptation is that while the system adapts to the user, the user will often try to adapt to the system as well. The user shares the systems goal of trying to get the input recognized as effectively as possible, and is hence prone to change the style of writing into one that is easier for the system to recognize. This results in yet another cause of variation in the user's writing style, which is in fact dependant on the recognition system used.

Actually the simultaneous user and system adaptation may have some negative effects. This could be due to the fact that while the system is adapting to a style it did not previously recognize, the user might assume that the style was "bad" from the system's point of view, and hence refrain from using that style of writing a character. The problem could of course be avoided by forcing the user to correct all recognition effects, but in terms of usability this solution is generally not desirable. However, if all the user input is taken into account in the adaptation, such conflicts should not arise as the system will adapt towards all styles of writing a particular character that were input to the system. Thus if the user starts to emphasize only one allograph in the future, continuous adaptation is precisely the means for the system to adjust its own behavior accordingly.

The value of on-line adaptation is clear especially when dealing with a personal input device. Most recognition systems have at least a linear correspondence between the number of prototypes and the recognition time. Hence it is logical to limit the number of models – which also makes sense due to the limited storage space and memory allocation restrictions. Also the fact that a personal device is generally used by only one user speaks for the value of user adaptation. The most significant downside to adaptation is the possibility of having the performance for other users degrade at the expense of improving performance for the one particular subject the adaptation is performed for. This is naturally a manifestation of the general over-learning phenomenon.

We have focused our research on on-line adaptation, and hence this viewpoint is also evident in evaluating the respective methods for adaptation offered by different recognition paradigms. In the following sections, adaptation approaches applicable to different classification methods and especially on-line adaptation are examined.

2.2.1 Adaptation of a prototype set

Modification, or adaptation, of the prototype set may be the most effective method for on-line adaptation. This is due to the fact that changes to the prototype set do not require extensive recalculations of model parameters, and hence changes are easy to make. As such the adaptation can usually be performed in a very short time frame, and it is easy to perform adaptation even after every sample obtained. Also the introduction of entirely new styles is as easy as adding new prototypes to the existing set. Similarly, discarding writing styles or models that cause a great deal of erroneous results can be performed simply by removing or inactivating the prototype causing problems.

In our research group, very positive results have been obtained with prototype set adaptation [207, 208, 213, 215]. The developed classifier uses the k-NN decision rule with Dynamic Time Warping (DTW) [174] based distance calculations. The prototype set is modified through either one, or a combination, of three basic operations: (i) adding prototypes, which is the most effective way to introduce entirely new writing styles, (ii) adjustment of prototypes based on a variation of the Learning Vector Quantization (LVQ) learning rule, and (iii) inactivating prototypes. The system will be discussed in more detail in Chapter 4.

The benefits of adding prototypes have been noted also in several other studies [161, 141, 159, 118]. Also other researchers have combined similar strategies, for example the use of prototype deletion, addition, and modification through weight adjustment for a classifier based on directional features [141]. A slightly different variation on the theme, using an adaptive template cache, where templates that have contributed to a positive classification are moved to the top of the cache and preferred in classification, has also been presented [76].

In [159] a recognition system that also uses a method for adding prototypes on errors was presented. Although the recognizer works on words, the underlying classification system is based on character prototypes. Having been applied for self-supervised adaptation, the study noted the benefits of a strategy combining prototype addition and inactivation. Another method similar in nature has been presented in [195], where prototype addition is combined with a form of prototype modification called Discriminating Templating Transformation.

Overall, adaptation to the user is quite simple and effective with a classifier based on prototypes. The prototype set can be modified with simple operations that are not computationally expensive, but still have an instant impact on future recognition results. The speed and ease of modifying the prototype set make these approaches especially attractive for on-line adaptation.

2.2.2   Parameter adaptation in statistical methods

As parametric methods are, by definition, established through training on some data and storing the information in the parameters of the system, adaptation requires changing these parameters. This process requires somewhat more effort than with prototype-based approaches. The computational cost and feasibility of the adaptation is highly dependent on the parametric model in question – for example recalculating parameters of a Gaussian is far simpler than completely retraining a large array of HMMs.

Practical success has been obtained in user adaptation in the setting of traditional statistical models such as a Gaussian model for cases with continuous style variation [206]. Other examples of similar approaches are the use of tangent vectors for adaptation when using Gaussian densities, Gaussian mixture densities and Gaussian kernel densities [82].

Adaptation schemes have been successfully implemented in also more complex settings such as HMMs. One presented approach is to focus the adaptation of an HMM-based system using models for different ways of writing characters [34, 36]. In one study [25], traditional parameter estimation techniques for the setting of HMM parameter adaptation, i.e. maximum likelihood, maximum a posteriori and maximum likelihood linear regression adaptation models, were compared. As a result it was noted that the reasonably simple maximum likelihood method was surprisingly effective. In [179] a writer-adaptation mechanism based on maximum likelihood linear regression was also employed in a word recognition setting and promising results were reported. Also a method using user adaptation with sub-stroke Kanji HMM models has been shown to be successful [140].

In summary, adaptation is an important and viable option also for parametric recognition systems, although the implementation of on-line adaptation may be slightly more difficult than for prototype-based systems. Still, several successful systems based on these approaches have been presented.

2.2.3   Adaptation in neural network methods

As a neural network by nature stores the data in the model coefficients, and the coefficients as a whole store the desired information. This means that a single input effects a multitude of connections and thus has a much larger scale of effect than a single prototype, for example. Hence adaptation in a neural network setting is often significantly more difficult to implement than in either prototype-based or parametric systems. Another notable problem with neural networks in

the adaptive setting is how to alter the parameters efficiently even with small amounts of user-specific training data. This is because a complete retraining of the network with the new data included alongside the original training data is in most cases practically impossible when operating on-line. Fundamentally there are two ways of adapting a neural network, either by altering the connection weights or by altering the network structure.

One interesting approach is to use a Time-Delay Neural Network (TDNN) trained first on user-independent data and then using it without the final layer to act as a pre-processor for an optimal hyperplane classifier which can easily be adapted [130]. Thus the effective qualities of the neural network are used, but easier adaptivity is obtained through using a different final decision mechanism, hereby making retraining the whole network unnecessary.

Also using self-growing probabilistic decision-based neural networks and implementing user adaptation of the parameters as incremental reinforced and anti-reinforced learning procedures [56] has been successful. One closely related study, although within the application framework of speech recognition, shows a method of efficient adaptation of parameters through adaptive training of polynomial networks [28]. All in all, retraining neural networks for adaptive purposes is possible, but not simple.

2.2.4   Other adaptive approaches

In addition to the three major groups of recognition strategies discussed above, also other types of adaptive strategies have of course been introduced. This section presents two examples.

Fuzzy decisions is another view on the classification problem, trying to forsake the strictness of binary yes-no relationships for fuzzy membership values. In the context of a fuzzy classifier, a new adaptation technique, inspired by the Learning Vector Quantization (LVQ) and Elliptical Fuzzy Competitive Learning (EFCL), has been presented [138].

Also an adaptive structural recognizer where HMMs are used for matching against a set of primitives to form stroke-level representations has been presented in [128]. In this recognizer, the priors of the character models can be updated on every new character.

2.3   Committee methods

Combining classifiers is an approach that has been shown to be useful on numerous occasions when striving for further improvement over the performance of individual classifiers – see [87] and references listed there, or for example [175, 75, 164, 88] among many others. Many fundamentally different classifier combination structures have been shown to be beneficial, and the identification of one superior combination approach remains impossible. Different combination methods may outperform one another in different tasks.

The basic function of a committee classifier is to take the results from a set of member classifiers and attempt to combine them in a way that improves overall recognition performance. Since the committee operation is based on the members' outputs – even though some committee methods do take also the original input into account in their decisions – the behavior of the member classifiers is obviously very important for the overall performance of the committee. The two most important aspects of the member classifiers that affect a committee's performance are (i) the error rates of the classifiers and (ii) how similar the errors made by the classifiers are.

In general it could be said that the more different the mistakes made by the member classifiers are, the more beneficial their combination can be [96, 6]. Or in different words, classifier combination can be beneficial as in the outputs of several classifiers the errors are not always overlapping [73]. It can be seen for most combination methods that the level of obtainable benefit decreases as the similarity between the member classifiers increases.

There are basically two types of information that classifiers output, either plain class information or some measurement values, the latter meaning that the classification result could be a multidimensional continuous output, not a strict labeling decision. There are some fundamentally different issues in how to combine classifiers that output either type of information. We shall restrict the discussion to the framework of this thesis and thus deal only with methods where the final classification is performed into discrete classes, although the member classifiers may well output values that indicate likelihood, confidence or fuzzy membership value of belonging to a number of classes. Thus we will restrain from discussing, for example, a number of linear combination methods that can be very effective for combining results expressible as numeric values and other approaches suited only to such a setting.
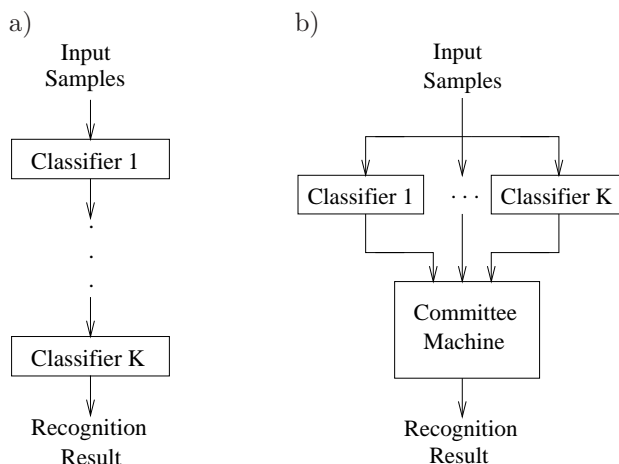
Figure 2.6: Basic a) serial and b) parallel committee structures

2.3.1 Types of classifier combination methods

Among classifier combination methods one division that can be made is between parallel and serial, also called multistage or cascade, combinators. In parallel structures the results of several, often assumedly mutually independent, classifiers are combined while multistage committees consist of several interrelated recognition stages further refining the results [87, 164]. The two basic types are illustrated in Figure 2.6, where we assume that K classifiers are used, and there are a total of C possible classes. The parallel combination approach is more common as a whole, and thus for the purpose of this discussion we will assume that a parallel structure is used unless otherwise specified. In the parallel strategy the most common approach is to use classifiers that are available and combine them, but also points of view on the importance of designing and combining the classifiers to reinforce each other should not be overlooked [192]. Yet another standpoint on categorizing classifier combination methods is a division to those that can be trained and those that cannot [44].

Perhaps still the most important distinction is between what information from the classifiers is used for the combinatory process, in other words, the level of output that the member classifiers provide the committee classifier with. On the least informative level the classifiers only output their suggested label. The next level is a list of outputs, in order of preference. On the third level are classifiers that output measurement level information, which may be for example estimates of the posterior probabilities, membership values or belief values. Finally, taking

Table 2.1: Categorizations of some classifier combination methods

Combination method	Ordering	Training	Information
Voting	parallel	no	label
Borda count	parallel	no	label
Behavior Knowledge Space	parallel	yes	label
Decision Templates	parallel	yes	label
Boosting	parallel	yes	label
Bagging	parallel	no	label
Min, max, sum and product rules	parallel	no	measurement
Expectation Maximization committee	parallel	yes	measurement
Critic-driven voting	parallel	yes	measurement
Pre-classification before classification	serial	yes	measurement
ENCORE	both	yes	measurement
Hierarchical neural gas	both	yes	measurement

a slightly different approach, the classifiers' outputs are viewed as feature vectors for the next level classifier [87].

Table 2.1 shows some common classification methods categorized by these characteristics. The column entitled 'ordering' refers to the classification stage; the ordering need not be the same for training and classification. For example the boosting algorithm is trained in a serial fashion, but can perform classification in parallel. A more detailed description of each method can be found in the following sections. For the adaptive theme of this thesis it should be noted that in general a method that is not trainable in itself is incapable of adaptation.

After a brief discussion on the importance of classifier selection in the next section, some common committee methods will be discussed. The committee methods will be divided into methods acting primarily on label information, here referred to as decision-level combining, followed by a separate group of training set alteration based methods, and methods working mainly with measurement level information. Additionally, under a separate topic, some multi-stage or serial combination strategies will be discussed. And finally, approaches where the second level is basically another classifier, and the outputs of the members are viewed as the inputs for this second level classifier, are examined.

2.3.2   Classifier selection

Even though research on committees most often focuses on methods for combining the classifiers in the most effective manner, it should not be forgotten

that the committee's performance is highly dependent on the member classifiers used. These two fundamental aspects in committee performance enhancement are sometimes referred to as *decision optimization* and *coverage optimization* [71].

The simplest way of choosing the member classifiers would be a selection based on their individual accuracies alone. The most straightforward approach is of course to take the set of the individually best classifiers. However, this is often not the optimal strategy and the gains that can be achieved may be significantly amplified through the use of classifiers that behave differently from one another in a certain sense, i.e. the set of classifiers ought to be *diverse*. The classifiers to be combined should be different from one another in a way that makes them complement each other, or else there will be no benefit from combining them. Diversity in itself, although not studied that long in the context of classifier combining, is an old concept. It has been, and continues to be, widely used in other contexts such as biology [146, 47] and evolutionary algorithms [202, 230], among others.

So instead of selecting member classifiers based solely on their accuracy, it may often be more effective to attempt to select the members based on their diversity. Measuring the diversity of the member classifiers is by no means trivial, and quantifying diversity has been considered an important research topic by several authors [100, 170, 38, 2, 110, 181, 152]. As there does not – and most likely cannot – exist any strict definition as to how diversity should be measured, in most situations a measure of diversity has some case-specific interpretation. Several different measures attempting to quantify diversity have been suggested for the purpose of classifier combining [100, 77, 168, 97, 90, 96, 184, 60, 170, 38].

Naturally there is also a trade-off between diversity and member accuracy – if all classifiers were completely correct, they would produce the same result every time. Standard statistics, such as variance or correlation, do not take into account that for classification purposes a situation where identical correct answers are given differs greatly from the situation where identical erroneous answers are suggested – the former being generally the most favorable case and the latter the worst. The concept of diversity and our approaches for improving classification performance from the diversity viewpoint will be discussed in more detail in Chapter 7.

2.3.3   Decision-level combination methods

First we will discuss some classifier combination approaches where the committee operation can be performed using only class label information from the member classifiers. It should be noted that many of these methods are simple to extend

by using some weighting scheme based on the measurement-level information obtainable – in some cases performance benefits may also ensue from this.

Class ranking methods

Arguably the most widely known method of classifier combining is *majority voting*. It has in spite of its simplicity been shown to be very effective on numerous occasions. Majority voting can be seen as a simplified *class ranking approach*, as it takes into account only the counts for the highest ranked class from each classifier. Theoretical consideration into the scheme's effectiveness has shown that majority voting does have a solid foundation [111]. In general, with classifiers that are correct on at least half of the inputs, the voting rule's performance increases as the number of classifiers involved increases [135].

Strictly speaking, a distinction should be made between majority voting, where the majority is required for a decision, and *plurality voting*, where the result obtaining most votes wins in any case. The term majority voting is often used for cases that should in fact be referred to as plurality voting – according to the aforementioned logic the majority vote rule should reject any samples for which the majority cannot reach a consensus, whereas the plurality vote would always return the most common label. A plurality voting committee has been used in the experiments presented in Publications 3, 4 and 8 of this thesis.

The plurality voting rule can be written as selecting that class $c_p(x)$ for the input sample x for which

$$c_p(x) = \arg \max_{i=1}^{C} \sum_{j=1}^{K} \Delta_i(x, j), \tag{2.5}$$

where we have a total of C classes and K classifiers. $\Delta_i(x, j)$ equals to 1 if classifier j suggests the class i as the most likely one for the input sample x and is otherwise zero. This can be seen as binary hardening of the *a posteriori* probabilities of the input belonging to a particular class [87]. The majority voting rule can be formed by using a class c_r to denote rejection and then the majority voting result for the sample x as

$$c_m(x) = \begin{cases} c_p(x), \text{ if } \sum_{j=1}^{K} \Delta_{c_p(x)}(x, j) > K/2 \\ c_r, \text{ otherwise} \end{cases}. \tag{2.6}$$

Another well-known class ranking method is the *Borda count*, which in turn can also be seen as a generalization of the majority voting rule. The Borda count for

Classifier1	Classifier 2 outputs				
outputs	1	. . .	j	. . .	C
1	(1,1)	. . .	(1,j)	. . .	(1,C)
⋮	⋮	⋱	⋮	⋮	⋮
i	⋮	⋮	(i,j)	⋮	⋮
⋮	⋮	⋮	⋮	⋱	⋮
C	(C,1)	. . .	(C,j)	. . .	(C,C)

Figure 2.7: Illustration of a 2-D Behavior-Knowledge Space model

a class is the negative of the sum of the number of classes ranked below it by each classifier. Let us use $B_i(x, c)$ to denote the number of classes ranked below class c for the input sample x by classifier i. Now the Borda count for the sample x and class c can be written as

$$BC(x, c) = \sum_{i=1}^{K} -B_i(x, c). \tag{2.7}$$

The results can then be ranked by arranging the classes in the order of their decreasing Borda counts [70, 72]. Also several variants of the Borda count method, such as averaging the ranks given by each voter for each class, or an iterative Borda elimination procedure have been suggested [203]. Both of these variants attempt to add the ability to differentiate between classifiers based on their general expertise.

Decision space methods

The *Behavior-Knowledge Space (BKS) method* [75] is based on a K-dimensional discrete space that is used to determine the class labels. Each dimension corresponds to the decision of one classifier. The committee result is obtained by first finding the *focal unit* in the K-dimensional space. The focal unit refers to the unit which is the intersection of the classifiers' decisions for the current input. The idea of the BKS is illustrated in Figure 2.7 for a two-classifier case with a total of C classes. Assuming that the classifiers output classes i and j respectively, the focal unit will be (i, j). Then during training, the true classes of the inputs are stored in their respective focal units and the classification will be based on this information.

If samples have been stored in the focal unit and for some class the ratio between the number of samples for that class and all the focal unit's samples is above a threshold, that class is selected. This threshold is used to control rejection, as unless the ratio is above the given threshold, rejection is performed. It has been experimentally seen that the BKS method performs well and expresses quasi monotonic behavior as the number of classifiers increases [89]. In that study, the addition of more classifiers did not hinder performance even though the classifiers were correlated and differed greatly in their performance. This means that the performance of the system is expected not to degrade as the number of classifiers increases. The BKS committee in its basic form has been used in Publication 8.

Another somewhat similar technique is the use of *Decision Templates* [98]. In this method, the combiner uses *decision profiles* that describe the classifiers' outputs for a sample. The decision profile DP_x is a $K \times C$ matrix where each element is one classifier's support, or confidence for the current sample to belonging to a particular class. The supports of one classifier for each class form one row in the decision profile matrix, with each column of the matrix corresponding to a particular class. During the training phase, a decision template is constructed for each class from the decision profiles obtained through processing the training samples. The overall decision template for class c can be constructed as an average over the decision profiles DP_{x_j} for the N training samples $x_j, j = 1, \ldots, N$. The (k, v)th element of the decision template matrix for class c, DT_c, is then calculated as

$$DT_c(k, v) = \frac{\sum_{j=1}^{N} \Delta_k(x_j, c) s_v^k(x_j)}{\sum_{j=1}^{N} \Delta_k(x_j, c)}, \tag{2.8}$$

where $\Delta_k(x_j, c)$ is again 1 if classifier k suggests the class c for sample x_j and $s_v^k(x_j)$ is the degree of support given by classifier k for the sample j belonging to class v. For a classifier k suggesting only crisp labels $s_v^k(x_j)$ would thus be 1 for the suggested class and 0 for all others. With classifiers suggesting fuzzy or probabilistic labels, $s_v^k(x_j)$ can be greater than 0 for more than one class. The suggested class is commonly selected by choosing the class for which $s_v^k(x_j)$ is the greatest. Thus the Decision Template approach can be used as either a decision- or measurement-level combination method.

Then a similarity measure is used to compare the decision profiles of the input samples while they are being classified to the decision templates. The similarity measure can be based on for example the ratio of the relative cardinalities of fuzzy sets. The classification decision is then made based on the similarity of the input's decision profile and the constructed decision templates [98].

2.3.4   Training set alteration based combination methods

With unstable learning algorithms small changes in the training set can cause large changes in the resulting predictors. For such learning algorithms methods that divide or alter the training data set in various ways have been found very effective. The most common example of unstable learning algorithms are neural networks, but the training set alteration methods can be applied to also other classification algorithms with similar behavior.

Perhaps the best-known committee approach in this category is *boosting*. Boosting is a method designed for converting a single learning machine with a finite error rate into an ensemble with arbitrarily low error rate [175]. It is a committee method especially suitable for increasing the performance of neural networks and other unstable learning algorithms.

The original boosting algorithm for training a neural network can be described as follows. First a set of training samples is used to train the first network. For the training set of the second network, the training samples are passed through the first network and the samples for the second network's training set are collected so that the first network has classified half of them correctly and the other half incorrectly. Then the third network will be trained with samples that the first and second network disagree on. The same training approach can then, if desired, be iterated in a recursive manner to produce 9, 27, and so on networks [43].

During the recognition phase, the samples are passed through all the three networks. If the first two networks agree, that is the output label. Otherwise the label from the third network is used. In [42] it was also shown that as the training set size increases, the training error decreases until it asymptotes to the test error rate. There have also been notable improvements presented to the boosting framework, namely AdaBoost [55] and its improvements such as using confidences assigned to predictions [177] and RankBoost [54]. The original boosting algorithm suffered from some notable problems which, for example, AdaBoost solved [176], but still the focal idea of boosting is perhaps most evident in that simple form.

Bagging, or *bootstrap aggregating* from where the acronym stems from, is another commonly used method for improving performance through training data set manipulation. The main idea for bagging is to train the members of the committee each on a random redistribution of the training data. Hence each member classifier has their training set generated by a different random sampling of the training set, with the size of the sampling remaining constant. The samplings are not exclusive, so some samples may be repeated in a number of training sets while

others are not used at all. The final decision can be obtained via averaging when the output is numerical or via voting if using discrete labels. The fundamental idea of bagging is the creation of multiple variations of a weak classifier through the redistribution of the training data. Bagging is also effective especially on unstable learning algorithms [26].

2.3.5 Measurement-level combination methods

In general it would seem logical to expect that if more information were available for the combination process than just the label information, it should be beneficial for classifier combining. There are several methods that are adept at combining classifiers which output numeric information on their approximated posterior probability of a class, or belief in correctness, or even just a distance to the nearest prototype. From a theoretic viewpoint these measures output by the classifiers pose noticeably different requirements, but they are addressed here together as combination methods that work on information on the measurement level.

Probabilistic combination methods

The task of probabilistic combination methods is rather simple in theory, as they all seek to maximize the probability (or minimize the cost) of the decision being correct. Thus the main idea of a probabilistic combination method is to obtain some kind of a probabilistic representation for the posterior probabilities of the classifiers' outputs. Then the combination method makes a decision based on these probabilities. This task is much easier if the classifiers output some meaningful measures that can be interpreted to reflect their probability of correctness, as the Bayesian probabilities can be derived from the classifier outputs that are on the measurement level [226]. For the case when measurement-level outputs are unavailable, [23] presented a method for estimating the probabilities from the data based on how often and where in the set of suggested results that particular output occurs.

Perhaps the best known rules for determining the final outcome with posterior probabilities are the *product*, *sum*, *min* and *max* rules, referring to choosing the maximum of the products of the obtained probabilities for each class, the maximum of their sums, minimums or maximums, respectively [87, 88, 45]. These rules will be discussed in more detail in Section 5.5.5, where they have been used in combination with our Class-Confidence Critic Combining committee. The

product rule is based on the assumption of independent classifiers and accurate probability estimates, and it is equivalent to the logical AND function. The sum rule, on the other hand, reflects the mean of the classifiers. Along similar lines, the min rule can be thought of as trusting the classifier who is the least confident in the decision, which could be expected to express caution. As an opposite, the max rule trusts the most confident classifier, something that can be hazardous if some classifiers give very bad estimates. Also the median rule can be applied in a similar fashion, and even the voting methods discussed above can be seen as an extension applicable through hardening the probabilities of the classes to one for the most probable and zero for others [87].

Through examination of these methods in one framework it was seen that the sum rule performed best [87], but this does not seem to be the case in all studies. An interesting result, which could partially explain this behavior, was that while the sum rule was analytically shown to perform best for Gaussian distributions of estimation errors, the voting rule can give better results for tail-heavy distributions and situations with very few experts [88].

Also linear and logistic regression have been used for classifier combining [10]. Furthermore, combinations of probabilistic and class-rank-based methods have been presented, such as the Mixed Group Ranks (MGR) method which combines rank selection and logistic regression through a linear combination of minimum functions [131]. The MGR score function can be written as

$$MGR(r_1(x,c), \ldots, r_K(x,c)) = \sum_{A \subseteq \{1,\ldots,K\}} -w_A \min\{r_j(x,c) : j \in A\}, \quad (2.9)$$

where $w_A \geq 0$ is a weight for that component of the linear combination and $r_i(x,c)$ denotes the rank given by classifier i for the input sample x belonging to class c.

Other examples of probabilistic approaches include combining classifiers by minimizing the Bayes error rate using higher-order dependencies [78], to use a Fisher discriminant function on the vector of individual expert score factors [39] (although this was compared to the sum rule and shown to perform poorer). Another recent and interesting novel approach of using a tomographic metaphor for classifier combining [223, 224] has produced very interesting results. This approach includes the application of tomographic reconstruction theory, regarding classifiers as probability density functions, and the use of a generalized inverse Radon transformation.

The Expectation Maximization (EM) algorithm, which can be used for a wide variety of tasks, including an iterative procedure for Maximum Likelihood (ML)

parameter estimation, can also be applied to classifier combining. For example a *mixture of experts* model can be optimized using the EM algorithm [227].

Also the Dempster-Schafer theory of evidence [180] can be seen as a kind of generalization to Bayesian combining. It is applicable also when handling weak evidence that does not fulfill the rather strict assumptions of probability theory. A computationally very efficient method can be derived from using binary voting. The votes are given for or against membership, by every expert for each class and feature space. Each vote can be handled as an independent source of evidence for the class membership of the input pattern. Thus it is not necessary to compute the combined belief for all of the possible subsets, but merely for the sets in focus for the final decision [53].

Critic-driven combination

An interesting enhancement to committee classification strategies is the inclusion of a *critic* into the decision scheme. Basically the task of a critic in classifier combining is just to decide whether the classifier the critic has been assigned to is correct or incorrect. Due to the fact that the critic only has two classes to decide from, its predictions can be expected to be more reliable than those of the classifiers taking on a multi-class problem [134].

Critic-driven approaches to classifier combining have been investigated for example in a situation where the critic makes its decision based on the same input data as the classifier [133] and in a case where scaling schemes and activation functions for critics were examined [66]. Also the Class-Confidence Critic Combining (CCCC) scheme, originally introduced in Publication 4 and to be discussed in detail in Section 5.5, is fundamentally an adaptive critic-based committee classifier.

Two approaches to critic-driven combinations are *critic-driven voting* and *critic-driven averaging of probabilities*. Critic-driven voting can be performed through a standard voting scheme with the exception that if the critic deems the expert's prediction to be incorrect, the expert abstains from voting. In one set of experiments [134], simple averaging was outperformed by using either geometric or arithmetic averaging, but still more information could be gained from the critic. The critics were used by giving special attention to the situation where zero probability was obtained from the critic. That was taken to mean that the expert's predicted class should be excluded.
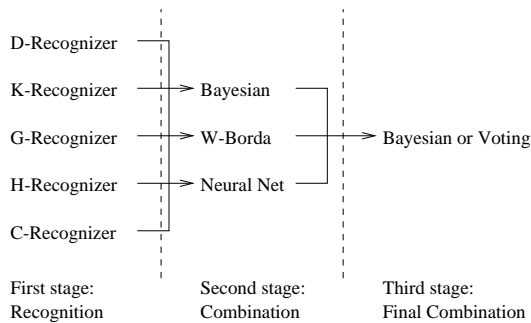
Figure 2.8: A block diagram of the combination system from [149]

2.3.6 Multi-stage combination

Multi-stage combination approaches can be used to make the classification process more efficient in terms of computational load. This is possible with systems where the input is first classified by a simple classifier using a small set of cheap features and a reject option. Then a more expensive classifier is used to handle the difficult samples that have been rejected by the first level [87]. Another possibility is to use a *pre-classification stage*, a simple classifier to prune the possible number of matches, and have a more complex approach decide the final matching [164]. Basically any method where the list of possible matches is first shortened by a simpler method and then the final decision is made without performing every matching should be considered a multi-stage classification system – for example all prototype pruning approaches [216, 221]. A method that combines a prototype-based first-level classifier that prunes the set of matches for an SVM-based system for the final decision was proposed in [33], and a method combining a first level based on a Self-Organizing Map [93] with a second level based on Learning Vector Quantization [158]. Yet another approach uses geometric, ink and directional features for pruning the prototype set before using a final elastic matching step for the final classification [222].

As an example, a two-level combination approach where there are five individual classifiers in the first stage, then three combining methods in the second, and a final combination in the last stage has been suggested in [149]. All the first-level recognizers are based on MLP neural networks using different feature vectors as input. The first recognizer uses a dynamic mesh feature (M-Recognizer), the second directional features extracted with a Kirsch mask (K-Recognizer), the third directional change features through the use of gradient vectors (G-Recognizer), the fourth histogram-based features (H-Recognizer), and the last one contour
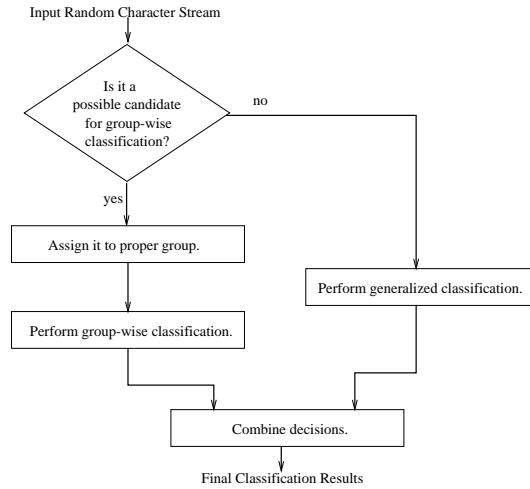
Figure 2.9: A schematic of a strategy incorporating prior knowledge to choose classes for group-wise classification [162]

chain code for boundary information (C-Recognizer). The first combination stage uses combiners based on a Bayesian method, Borda function and a neural network. In the final stage, either a Bayesian or voting combiner is used. A schematic of this committee system is shown in Figure 2.8.

Another option is to use specialized classifiers to re-process classes causing a notable amount of confusion [165]. Such methods are often tailored to the application at hand, for example by using specialized classifiers constructed for separating precisely those difficult classes. In one such method [163, 162], the basic classifier first performs an initial separation of the input characters. Based on the *a priori* knowledge, groups of character classes likely to cause confusion undergo group-wise classification, whereas structurally dissimilar characters are directed to the general classifier. This idea is illustrated in Figure 2.9. The final decision is then obtained by combining the decisions of the general and specialized classifiers. The decision is based on sufficient discrimination with either a sample, class or overall confidence index obtained from the classifiers.

One approach of combining complementary classifiers sequentially is a method whereby each classifier is designed to recognize samples misrecognized by the preceding combined classifier. The approach uses common difference principal components and difference principal components as features [81]. Another ex-

ample of a multi-stage scheme is the *ENCORE* system [46]. In ENCORE, the classifiers are first combined according to the majority rule, but then the concensus is evaluated based on past performance of each classifier in similar situations. If necessary, the decisions are modified using their confidences before obtaining the final decision.

2.3.7 Member classifiers as features

Still a different viewpoint on constructing a combination method is to use the outputs of the member classifiers simply as a feature vector input to the next classifier. Especially neural networks and similar methods have been used in this task, for example in [10], as their black-box nature and lack of necessity for preprocessing are especially suitable for this type of operation.

In a way this can be seen as combining several networks into one large network. Also other than neural classification methods could be used as either the member or the final layer – or both layers could be entirely different types of classifiers. If using neural networks, the networks need not be completely exclusive either. An example of this kind of a recognition system is the hierarchical overlapped architecture of neural gas classifiers, where overlapping networks are combined [16]. In [45] it was seen that a nearest mean and a nearest neighbor rule performed well and seemed stable when used as combination methods. We have also used a Support Vector Machine to combine classifier outputs in Publication 5 of this thesis.

2.4 On-line adaptive committee methods

As with classifier adaptation, the purpose of committee adaptation is generally to improve performance by adapting the system to a particular user. Classifier adaptation is generally capable of working closer to the data, and hence single classifier adaptation may alone be more effective than committee adaptation. This can be the case e.g. when adding entirely new ways of writing to the set of prototypes. Committee adaptation, on the other hand, is generally based on the member classifiers' outputs and their correctness.

Due to operating on a more abstract level than classifier adaptation, committee adaptation does have some significant benefits. Namely, it is possible to use an adaptive committee structure to improve the performance of classifiers that for some reason cannot be made adaptive themselves. Such reasons may include

a structure that is simply unsuitable for effective adaptation, such as a neural network that would require recalculation of interneuron weights. Another reason could be that access to the workings of the classifier is denied due to patent rights, for example. An adaptive committee can be implemented on top of a classification system and can improve over the performance of the member classifiers without even detailed knowledge of the task at hand. Also, committee adaptation can be combined with classifier adaptation, as has been done in Publication 8 and will be discussed in Chapter 5.

There are several approaches to committee adaptation, ranging from simple weighting schemes to very complex rule-based systems and structures with units specialized on predicting the member classifiers' performance. The field of adaptive committee classifiers is, however, a rather new approach. Therefore not very many effective on-line adaptive classifier combination methods applicable to our setting of handwritten character recognition have yet been presented. Here an overview of some previously published methods will be given. Our own work on the subject will be discussed in detail in Chapter 5 and Publications 2 through 8.

2.4.1   Non-neural adaptive committee methods

First, some non-neural approaches to committee adaptation will be discussed. These approaches are mostly based on weighting schemes or storing the results of previous samples for use in the decision process.

Adaptive weighting schemes

The simplest approach to adaptive combination is probably to calculate adaptive weights for the classifiers as the classification progresses. These could be based on for example the classifiers' overall performance so far or each classifier's performance for a particular class. The weights can then be used for weighting the decisions of a voting classifier or for simply selecting the output of the classifier with the highest weight so far. This approach has been taken in Publication 5.

Another weighting-based strategy is to combine the member classifiers linearly with the use of some weighting coefficients. These weighting coefficients can, for example, be dynamically acquired from a combination coefficient predictor [225]. The coefficient predictor is trained to give more precedence to certain member classifiers in situations where they have provided good results. As a result the weights enhance the effect of the best classifiers on the decision for every input

sample. Thus the resulting linear combination of classifiers should be biased towards the best classifiers in each situation.

In the context of data mining, [48] suggests ensemble weighting as the key to fast adaptation. They employ a three-step strategy to adjust the classifier set, first training a new classifier on the new data, then replacing the oldest classifier of the ensemble with it, and finally weighing the classifiers in the committee. In [50] a kind of a weighting scheme based on Arc-x4, an off-line boosting-style algorithm, is presented in a branch prediction context where online learning is very beneficial due to the time constraints.

Adaptive decision space methods

The original implementation of the Decision Template method [98] constructs the templates based on training data. However, there is no reason why the method could not be adjusted to work in an adaptive fashion by adjusting the templates during use.

Also the Behavior Knowledge Space (BKS) system discussed previously could quite well perform in an adaptive fashion by storing the made decisions during operation, although this was not included in the original implementation in [75]. An adaptive version of the BKS classifier has been presented and studied in Publication 5, where it clearly outperforms the non-adaptive version.

2.4.2 Neural adaptive committee methods

Also adaptive committees based on neural networks have been presented. Here some examples of such approaches will be discussed.

AIME

One of the adaptive committee recognition methods found in the literature is the Adaptive Integration of Multiple Experts (AIME) system [198], illustrated in Figure 2.10. The system employs a fuzzy neural logic gating network and an exceptions expert, both trained using the Supervised Clustering and Matching (SCM) algorithm [193]. The gating module learns to assess the performance of the individual experts for the situation at hand based on the experts' performance over the data space. AIME is thus capable of giving precedence to experts which
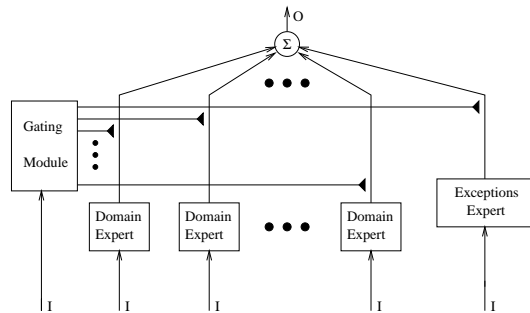
Figure 2.10: A schematic diagram of the AIME system

have previously performed well in certain situations. The AIME system is suited for both off-line and on-line learning. It can thus be initially trained and the performance then further improved through adaptation during operation.

On-line adaptive Bayesian combination of PFAMs

Adaptive Resonance Theory (ART) learning using a Probabilistic Fuzzy Artmap (PFAM) [116] has been used to combine a set of neural networks. The combination of the experts in this framework was performed in one of three ways, either simple voting, by a Bayesian method combining the confidences from a confusion matrix, or by using the BKS approach above. The experiments were conducted using both traditional training and an approach called dual-mode training, where initial training is first performed and then followed by on-line learning.

Chapter 3

Label deduction in on-line adaptation

One key component for the feasibility of using on-line adaptation in a transparent fashion is obtaining the correct labels for the recognized samples. Sufficient correctness of the labeling is necessary for the adaptation process to be effective. In batch experiments, it is often assumed that the correct labeling is available for the adaptation process. How the labeling would be obtained in practice is all too easy to overlook.

The influence of incorrectly labeled training samples depends highly on the recognition system used. For the adaptive single classifier classification framework used in our laboratory, a concerning result was noted. If the probability of incorrect labelings reached 3–4 percent, the adaptation could actually cause the performance to deteriorate to a level below initial if no counter-measures were taken [211].

Thus obtaining the correct labeling is of fundamental importance for successfully implementing an adaptive recognition system. In Publication 1 we have used a simple rule-based method that attempts to deduce the intentions of the users from their behavior after the recognition. The experiments were performed with an on-line character recognition system, implemented on a handheld device. It will be described in the following sections.

Figure 3.1: The PDA user interface of experiments in Publication 1

3.1   The handheld application

The method for obtaining correct labels for input characters presented in Publication 1 was implemented for a Personal Digital Assistant (PDA) user interface. Characters are input one-by-one into a text field and the result of the recognition is shown on the screen where text appears. The application in question was a questionnaire program, but the same strategy is valid for any setting where multiple characters are input. An illustration of the user interface of the application presented in Publication 1 can be found in Figure 3.1.

Since recognition errors cannot be entirely avoided, a relabeling option in the user interface is highly recommendable. The relabeling option also makes it possible for the system to learn entirely new ways of writing a character. By a relabeling option it is meant that by some simple means the user can give the input sample the desired label, regardless of how the sample was recognized by the system. The relabeling option can be implemented as for example pressing a dedicated relabeling button in the interface and then selecting the correct label from a list. If such an option is not available, the user may become quite frustrated if his or her way of writing some character is repeatedly misrecognized by the system. If the correct label is not obtained through recognition, it is nearly impossible to teach new writing styles unless the relabeling option exists. A relabeling button was used in our user interface.

In our application a specific submission button was used to start the adaptation. However, also other events can be used, like shift of focus away from the application, saving the text document, etc. In any case, the adaptation should be performed after the intended message is complete, as then it is quite safe to assume that a careful writer has corrected all recognition errors. This was easily implemented in a questionnaire as submitting the answer.

3.2   A method for obtaining correct labeling

For adaptation, it is desirable to obtain as much data as possible. The characters that were initially recognized incorrectly will in fact have the most information value to the adaptation. Thus we aim to discard as few samples as possible. Of course, it is also imperative to make as few labeling errors as possible. Therefore one will attempt to deduce the correct class of initially incorrectly recognized samples from the users' actions as they correct their input message to read as it was intended to.

The underlying data structure to store samples and their labels we have used is a simple linked list where each sample is stored along with its label, input index and position information. This list is modified as the writing process continues, and the first item with a given position always corresponds to the character left into the input word. If more than one input sample has been input and retained for a given position, they all are assigned the label of the topmost sample in the "pile" of that position.

A schematic example of the data structure is shown in Figure 3.2. The example depicts the user having written the first six capital letters. The first two were input successfully and in order, but the character "C" has needed three attempts for the correct recognition. Then the user has input a character that was later deleted, as the index number 6 is non-existent. Finally the letters "D" and "F" were input, and "E" was inserted between them afterwards.

The objective is to assign the correct label for each input sample while discarding only those inputs for which it is impossible to confidently suggest a label. To facilitate this, a number of cases and proper actions to be taken can be isolated:

1. A sample has been written and left unchanged after seeing the recognition result. Such a sample is considered to have been correctly recognized.

2. A single sample has been directly replaced with another sample. Thus it is deduced that the initial sample was incorrectly recognized and the label
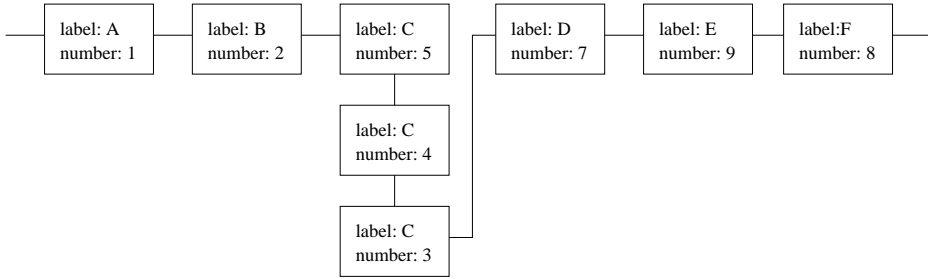
Figure 3.2: A diagram illustrating the data structure for deducing the labels

of the new sample is also assigned to the underlying sample. Both samples are kept in the input sample list and the older sample (together with all others in the same location) is relabeled according to the most recent input for that location.

3. A single sample has been relabeled using the relabeling option presented by the user interface. Then the label of the sample (and all others in the same location) is simply changed to that received from the user. The possibility of manual relabeling should be offered to the user to avoid frustration from several attempts and still not attaining the desired classification.

4. Several characters have been replaced with one input of any kind. This is thought to indicate the user's change of mind, and in such a situation nothing concerning the labels of the samples being replaced can be assumed. The replaced samples are thus discarded and removed from the list of learning samples, and only the most recent input is kept.

5. Several consecutive backspaces have been received. Also in this case the correctness of the samples being deleted cannot be established and as such they are removed from the list.

Through the use of these rules the list containing the input samples is kept up to date and the labels therein are assumed correct for the adaptation process. Naturally the possibility of incorrect labels still exists, but through these principles the labels should be correct if the user has noticed and taken care to correct all recognition errors.

The one unsolved situation is that of the user's change of mind for a single character, meaning that the user writes one character, deletes it and writes an entirely new one instead. Such situations are confused for case 2 above. It was

Figure 3.3: An example of a user input and the recognized string

thought that the error correction, the basis for case 2 above, is more important and common, so the possibility of error was deemed a reasonable risk.

Figure 3.3 shows an example, from Publication 1, of text input with character labels deduced according to this logic. As can be seen, the first character took several attempts for correct recognition, as did the third character 'l'. The latest attempt is shown on top. In this first word the adaptation would function as desired, as all inputs are clearly representative of their proper classes, and labeled correctly. In the second word, "world", the first time the user wrote the second character it was clearly an 'a'. But since the intention was to write the word "world", the user corrected this input to 'o', resulting in the replaced sample being labeled (erroneously) also as an 'o'. Such errors can remain with the logic presented, and this also gives cause for implementing other measures to enhance the robustness of the system as in real applications deduced labels really cannot be blindly trusted to always be correct.

Even though no comparative evaluation on the performance of the label deduction rules has been performed, the results of Publication 1 clearly indicate that the scheme is effective. Those results showed that while initial error rates are approximately one out of five characters needs to be resubmitted, this rate improves to roughly one out of 16 with adaptation. This is clearly indicative of the effectivity of the adaptation, which in turn requires sufficiently correct labeling. Thus the labeling provided by the presented scheme can be seen to have been effective by making the impressive recognition accuracy gains obtained through adaptation possible.

3.3   Maintaining robustness in presence of erroneous labels

Being aware of the possibility of incorrect labeling, as well as other maverick data samples, makes it possible to have the system incorporate features that attempt to adjust to their presence. The objective is to limit the effect of erroneous samples before they can cause too much harm. In the prototype-based adaptive recognizer used in our studies and to be discussed in Chapter 4, it was noted that through aggressive inactivation of mistake-causing prototypes the system could continue to improve performance through adaptation with even one in ten input samples being erroneously labeled [211]. The inactivation strategy used was to inactivate a prototype after only one or two incorrect classification results.

Another way of avoiding long-term effects of incorrect labelings is to have the adaptation method focus more on the most recent samples, giving less weight to older ones. Such a method has been implemented for the Class-Confidence Critic Combining scheme to be discussed in Section 5.5. The main idea of the weighting scheme is to have a linearly decreasing effect of older samples, with most weight being given to the most recent one. The weighting scheme is described in detail in Section 5.5.3 and was experimented with in Publications 6 and 8. This will in practice limit the effect of older data, including erroneously labeled samples, which makes corruption of the whole system much less likely.

Chapter 4

On-line classifier adaptation – The CIS-HCR system

An adaptive on-line Handwritten Character Recognition (HCR) system has been constructed by our research group in the laboratory of Computer and Information Science (CIS) of the Helsinki University of Technology [107, 106, 207, 108, 213, 1, 210, 214, 217, 215, 216, 211, 212, 209]. The system will be here referred to as CIS-HCR. The contributions of the author of this thesis have been the implementation of the symbol string creation, distance computation for the Symbol String Classifier, and the pre-processing methods for the Local Subspace Classifier, as well as implementing the system on a standard PDA. As the adaptive system is used as the main source of member classifiers for the adaptive committee experiments, an overview of the system in its entirety is given here. The experiments and results shown in this section are not from any of the included publications, but are presented to illustrate the performances of the various classification strategies and justify the selection of classifiers used for the committee experiments in the included publications and Chapters 5 through 7.

A schematic diagram of the recognition system is presented in Figure 4.1. The input, in the format described in Section 4.1, is first preprocessed and normalized as described in Section 4.2. The feature extraction is performed for the Symbol String Classifier and Local Subspace Classifier approaches as described in Section 4.3. The classification approaches are described in Section 4.4 and finally the adaptation in Section 4.5.
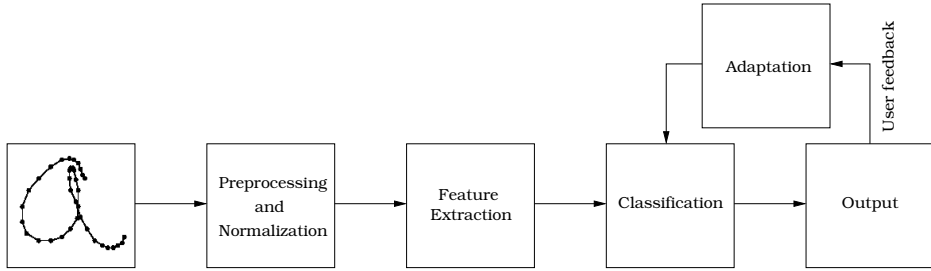
Figure 4.1: A schematic diagram of the adaptive recognition system

The adaptive recognition system is based on various forms of template matching. The system is adapted to new writing styles by either adding, inactivating or modifying the prototypes in the individual recognizers, or by utilizing some combination thereof. Three main classification techniques have been experimented with. They are Dynamic Time Warping (DTW) on the points of the strokes, the Symbol String Classifier (SSC) and the Local Subspace Classifier (LSC).

4.1 Data acquisition

In the prototype version of CIS-HCR, all character data were collected with a pressure sensitive Wacom ArtPad II tablet attached to a Silicon Graphics workstation. The collected data consists of the x- and y-coordinates, pen's pressure against the writing surface, and a time stamp. The characters were written one at a time. Writers were advised to use their natural handwriting style. The data was saved in UNIPEN format [65]. Data collection was performed on a stroke-wise basis, so that pen-up and pen-down points are stored and the characters can be classified in a stroke-wise basis when desired. Important details of collecting the databases are summarized in Table 4.1. The necessity of collecting a new database instead of using publicly available benchmarking databases for our experiments arose from the need to have a sufficient number of samples from each particular writer. This is paramount for being able to properly examine the possible benefits of on-line adaptation. Commonly used databases, while often consisting of a very large number of samples in total, are usually written by a large number of different writers and as such are poorly suited for experimentation with on-line adaptive recognition systems.

Database 1 consists of characters written without any visual feedback, following dictation given by the data collection program. The data collection was per-

formed on a workstation. The pressure level thresholding the pen movements into pen up and pen down movements was set individually for each writer. The distribution of the classes (a–z, A–Z, å, ä, ö, Å, Ä, Ö, 0–9, (,), /, +, –, %, $, @, !, ?, :, ., and ,) was similar to that of the Finnish language.

Databases 2 and 3 were collected with a program showing the pen trace on the workstation screen and recognizing the characters on-line. The minimum writing pressure for detecting pen down movements was the same for all writers. The distribution of the character classes (a–z, A–Z, å, ä, ö, Å, Ä, Ö, and 0–9) was nearly even. None of the writers of Database 1 appeared in Databases 2 or 3.

4.2   Preprocessing and normalization methods

Prior to the classification and adaptation phases, the input characters need to be preprocessed and normalized. First, an operation called *NoDuplicatePoints* is used, where sequential data points having the same coordinate values are merged.

The sampling frequency can be altered with one of two operations, *decimate(n)* or *interpolate(n)*. Of these decimate keeps every $(n+1)$th data point and discards the intermediate ones, whereas interpolate interpolates n equally-spaced points linearly between all original data point pairs.

The size variations in the characters are normalized with an operator called *Min-MaxScaling* which scales the size of the character so that the length of the longer side of the character's bounding box is the same for all characters. The aspect ratios of the characters remain unchanged. This normalization of course as a side effect makes the separation of character pairs for whom the main difference is their size, such as the 's' and 'S' or 'o' and 'O', notably more difficult. However, refraining from scaling causes significant problems as the size of writing was not enforced, and all distance measures used are sensitive to size variations. Thus it has been experimentally found that the size normalization is on average clearly beneficial for recognition accuracy and was thus used.

Table 4.1: Summary of the databases used in the experiments.

Database	Subjects	Characters
DB1	22	10 403
DB2	8	8 046
DB3	8	8 077

The original character	cdlldo	cdlldhna

Figure 4.2: Examples of symbol strings created

The unknown character and the prototypes are moved into the same location so that they can be properly matched. This is carried out by moving their center points to the origin of the coordinate system. The normalization method *MassCenter* defines the center as the mass center of the sample, while *BoundingBoxCenter* uses the center of the bounding box.

According to experiments, the best average recognition result with the DTW-based classifier can be obtained if *NoDuplicatePoints*-operation followed by *Decimate*(2)-operation is used as a preprocessing method, characters are normalized with *MinMaxScaling*- and *MassCenter*-operations. The *BoundingBoxCenter* normalization was used for the SSC classifier and to provide variation for the DTW classifiers in the committee experiments of Chapter 5.

4.3   Feature extraction

With the DTW classifier, no feature extraction methods were needed as the matching was performed directly with the normalized coordinate sequences. The SSC and LSC classification methods required specific feature extraction steps. These features will be described in the following.

4.3.1   Symbol string representations

Before the creation of the symbol string representations all strokes in the character were joined. When joining the strokes, information on where the pen was taken off the tablet was stored. The characters were normalized using the *BoundingBoxCenter* and *MinMaxScaling* operators. This resulted in a centered and scaled one-stroke character in a 1000×1000-sized box with the pen-up points marked.

The discretization of the character was performed by setting a discretization distance, or segment length, l_s for each directional symbol and following the pen-trace until this distance was reached, the trace ended, or a pen-up point was encountered. Then, the direction of the resulting vector was calculated and quantized. The number of directions d_s was varied from 4 to 32. The parts of the character with the pen up were marked with separate symbols. The actual length of the corresponding line segment was stored together with the direction symbol.

In addition, a corner detection method, which was most sensitive to changes near the center of the character, was applied [1]. An example of the produced symbol strings is shown in Figure 4.2, where the first symbol string has been formed without the corner detection algorithm and the second string with the corner detection active. The first 16 alphabets 'a' through 'p' were as direction labels used to represent 16 directions in clockwise order, with 'a' being horizontal to the right.

4.3.2   Thickened strokes

In order to form feature vectors of fixed dimensionality suitable for statistical classification, two feature extraction methods were used [104]. In the first method, the straight lines connecting the measured (x, y)-points were thickened to the width of $2r$ units in a coordinate system where the image was centered in a 1024×1024-sized frame. The original frame was then down-sampled to the size of 32×32 by averaging. The initial character sample is illustrated in Figure 4.3 (a) and the result of the first method in Figure 4.3 (b).

In the second variation, two 32×32-sized images were created instead of one. The directions of the lines connecting the sampled pen positions were used as additional information when creating the images. In the first one, the vertical component of the direction of pen movement was used in thickening the path. The filling value, represented in the image with the brightness of the corresponding part of the stroke, was obtained as $f_v = \sin\theta$, where θ is the line direction. Likewise, the horizontal part $f_h = \cos\theta$ was used in the second image. The results of this method are shown in Figure 4.3 (c) for the vertical direction image and Figure 4.3 (d) for the horizontal direction image.

The final feature vector was created through concatenating the pixel values of the grey-scale images. This gave rise to 1024-dimensional pattern vectors in the former and to 2048-dimensional vectors in the latter case. The covariance matrix of the training data set was calculated after the feature extraction, and the first

| (a) | (b) | (c) | (d) |
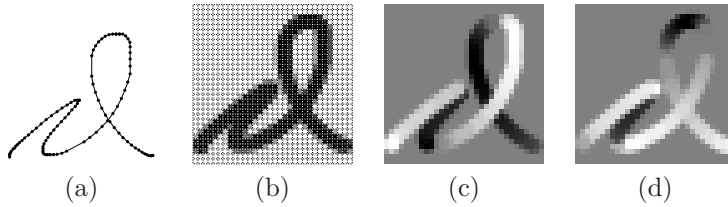
Figure 4.3: An example of the thickened stroke preprocessing

64 eigenvectors [69] of the covariance matrix [171] were used in projecting the pattern vectors using the Karhunen-Loève transform (KLT) [79]. The choice of using the first 64 eigenvectors was based on earlier experiments with off-line handwritten digit classification [103].

4.4 Classification techniques

Three different techniques have been used for the classification of handwritten characters. They are Dynamic Time Warping, Symbol String Classifier, and the Local Subspace Classifier. These methods will be described below.

4.4.1 Dynamic Time Warping

The Dynamic Time Warping (DTW) algorithm [174] has been used to match the input strokes to the prototypes through nonlinear curve matching. The algorithm finds the optimal matching of the data points which corresponds to the minimum sum of the costs and which satisfies the boundary and continuity conditions. The continuity condition requires that all data points are matched and in the same order as they have been produced. In the point-wise distance measures the boundary conditions require additionally that the first and last points of the input and the prototype strokes are matched against each other.

Classification is performed by evaluating the dissimilarity measures between the unknown character and all the prototypes and then applying the k-Nearest Neighbor rule [37]. Only characters with the same number of strokes are matched. The prototypes are also ordered on the basis of the locations of the starting and ending point of the first stroke. These two techniques improve the time efficiency of finding the nearest prototypes for the input character.
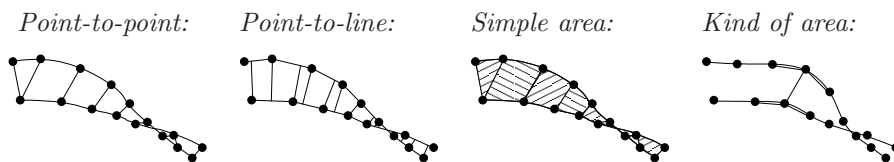
Figure 4.4: Illustrations of the DTW-based distance measures

Six different dissimilarity measures have been used. All the DTW-based dissimilarity measures are described in full detail in [207]. The main difference between the measures is the associated cost of matching a data point. All the measures are defined on stroke-wise basis. The dissimilarity measures are the *Point-to-point* (*PP*), *Normalized point-to-point* (*NPP*), *Point-to-line* (*PL*), *Normalized point-to-line* (*NPL*), *Kind-of-area* (*KA*) and *Simple-area* (*SA*) distances. The *PP*-distance uses the squared Euclidean distance between the data points as a matching cost. In the *PL*-distance, the data points are matched to lines interpolated between the data points. The *NPP*- and *NPL*-distances are otherwise similar to the *PP*- and *PL*-distances, respectively, but the sums of the matching costs are divided stroke-wise by the number of matchings.

The *KA*-distance also matches data points against data points, but the Euclidean distances from the matched data points to their neighboring data points are considered too. The *SA*-distance uses the area between the strokes approximated with triangles or quadrilaterals as the matching cost. These two distances measure the area left between the matched strokes and are therefore more sensitive to the shapes of the strokes than their data point densities. The used dissimilarity measures are illustrated in Figure 4.4.

4.4.2 Symbol String Classifier

The Symbol String Classifier (SSC) uses a distance measure based on the Levenshtein distance [114] for comparing the symbol string representations. Three kinds of operations are used: replacements, removals and additions. Each of these can be assigned a specific cost function.

In our experiments, this distance measure was somewhat modified so that information regarding the neighboring symbols was also used in determining the modification costs. Extra cost for alteration of special symbols referring to movement when the pen is off the tablet was also added. This additional penalty helps preserve the stroke information available in the original stroke-based structure, as the cost of modifying stroke-ending symbols was notably higher than that of

regular alterations. If length information was used the cost was also dependent on the lengths of the symbols. Details of the symbol string based distance can be found in [1]. Based on these costs, the actual distance between characters was calculated with a dynamic programming algorithm [174].

4.4.3   Local Subspace Classifier

The Local Subspace Classifier (LSC) method [102] models the distribution of the pattern classes in a non-parametric fashion by using existing prototypes to span lower-dimensional local subspaces in the feature space. The distance is defined between the input $\mathbf{x}$ and the local subspace nearest to it. The LSC method was also discussed in Section 2.1.2 and an illustration of its operation can be seen there in Figure 2.4.

When calculating the distance between the input vector $\mathbf{x}$ and pattern class j, the $D + 1$ prototypes belonging to class j and nearest to $\mathbf{x}$ are first searched for. A D-dimensional linear manifold $\mathcal{L}_j$ of the d-dimensional real space can then be spanned by these prototypes. When $\mathbf{x}$ is projected orthogonally onto this manifold, a residual vector $\widetilde{\mathbf{x}}_j$ results. The classification of $\mathbf{x}$ is then performed according to the shortest $\widetilde{\mathbf{x}}_j$ among classes $j = 1, \ldots, C$ where C is the number of classes. In any case, the residual length from the input vector $\mathbf{x}$ to the linear manifold is equal to or smaller than the distance to the nearest prototype $\mathbf{m}_{0j}$, i.e. $\|\widetilde{\mathbf{x}}_j\| \leq \|\mathbf{x} - \mathbf{m}_{0j}\|$. It can be seen that the LSC method degenerates to the 1-NN rule when $D = 0$. [104]

4.5   On-line adaptation

With the DTW-based classifiers four different prototype set adaptation strategies have been applied. They are *Add*, *Inactivate*, *LVQ*, and *Hybrid* [207, 213]. *Add*(k) examines the classes of the k prototypes nearest to the input character. The input character is added to the prototype set if any one of these prototypes belongs to a wrong class, even if the classification was correct. *Inactivate*(N) is used for inactivating those prototypes which are more harmful than useful. After each recognition, it is checked if the prototype nearest to the input character has been the nearest one at least N times and whether its class has been incorrect more often than correct. In that case, the prototype is removed from the set of active prototypes.

When a character written by the user is similar to a prototype of the correct class, but of slightly different shape, the existing prototype can be reshaped instead of adding the input character to the prototype set. This can be performed with an adaptation strategy called $LVQ(\alpha)$ based on a modified version of the Learning Vector Quantization (LVQ) [93, 106]. Parameter α controls the degree of reshaping. With larger values of α the modifications to the prototypes have more impact. $Hybrid(\alpha,k)$ combines the $Add(k)$ and $LVQ(\alpha)$ strategies. The k nearest prototypes are examined. If any one of them belongs to the same class as the input character, the nearest prototype is modified with $LVQ(\alpha)$. Otherwise, the input character is added to the prototype set.

In the Local Subspace Classifier experiments we started with a user-independent 1-NN classifier created with the K-means algorithm [117, 126]. For each writer, the user-dependent LSC prototype set was initially empty. The adaptation of the LSC classifier was then performed according to one of two rules. In the 'E' rule the prototype was added only if the LSC classifier had misclassified the input. The 'A' rule forced the addition of every input character. Every input character was classified with both the user-independent 1-NN classifier and the adaptive user-dependent LSC classifier. The joint classification decision of the two was given by the one with shorter distance to either to the nearest prototype or the nearest local subspace, respectively. This was possible as both types of classifiers are based on the Euclidean distance metrics and measure the residuals in the same units. If the class provided by the 1-NN classifier was incorrect, the corresponding prototype in the K-means-initialized prototype set was removed. The input character was added to the LSC prototype set according to either of the 'A' and 'E' rules depending on the experiment. As a result, the size of the user-independent 1-NN classifier decreased while the size of the user-dependent LSC classifier increased during the adaptation.

The SSC classifier used similar 'A' and 'E' rules as the LSC classifier in adaptation. A notable difference between the two methods was that the prototype set of the string-based classifier was initialized by using all available samples instead of a K-means-clustered subset. Also, the initial prototypes were never removed even when they caused false recognitions.

4.6   Experiments

Database 1 was used for forming the initial prototype set and Database 3 was used as a test set. The prototype sets were formed by first clustering character samples of Database 1 written by several subjects and then selecting the middle-

most items of the clusters to present the corresponding styles of writing. In the SSC classifier, the computational requirements were lowest and it was feasible to use the entire Database 1 as the prototype set.

In the experiments a character set containing the digits and lower case letters including three Scandinavian diacriticals (å, ä, ö) was used. The recognition error rates shown have been obtained as the average of the results on Database 3. A larger dataset that included Database 2 was used for selecting suitable values for the parameters. These parameters were k, N and α for the respective DTW adaptation strategies as well as the SSC parameters d_s and l_s and K and D for the 1-NN and LSC classifiers.

The experiments were run as batch experiments on previously collected data, and hence there was no actual feedback from the user to the recognition system. It was also assumed that the true classes of all characters were known for both the adaptation and the calculation of recognition accuracies.

4.7   Results of classifier adaptation

The results are grouped in Table 4.2 so that in every group the first line shows the results of the non-adaptive classifier. Then, results with different adaptation strategies are shown for each recognition method. Two error rates are calculated. The *total* error percentage was measured for each writer during the whole test run. The *final* error rate was evaluated for the last 200 characters of each writer. It thus gives better impression of the obtainable recognition accuracy after adaptation.

It was noted during the experiments that the writing style of some subjects got rather poor during the last characters due to fatigue and lowered motivation. This can also be observed in the results of some non-adaptive classifiers where the *final* error rate is higher than the *total* rate. The numbers of final prototypes in the last column of the table are the sums of the numbers of the remaining user-independent and the added user-dependent ones.

With the DTW classifier, the best results were obtained with $Add(k)$ when $k = 4$, and with $Hybrid(\alpha,k)$ when $k = 3$ and $\alpha = 0.3$. The same value of α worked also best with $LVQ(\alpha)$. $Inactivate(N)$ did improve the recognition accuracy only when it was applied with $Add(k)$. In that case, the best value for N was 3. The best recognition result in the whole series of experiments was obtained with DTW when adaptation strategy $Add(4)$ was used together with $Inactivate(3)$. It can

Table 4.2: The resulting error percentages of the adaptive classifier experiments

Recognizer	Errors		Units	
	total	final	start	end
DTW	14.1	14.1	273	273
DTW-$Add(4)$	3.1	1.8	273	453
DTW-$LVQ(0.3)$	9.9	8.6	273	273
DTW-$Add(4)+Inactivate(3)$	3.0	1.6	273	450
DTW-$Hybrid(3,0.3)$	4.2	2.5	273	278
DTW-$Hybrid(3,0.3)+Inactivate(16)$	4.3	2.8	273	278
SSC(d_s=32,l_s=15)	26.1	27.1	8461	8461
SSC-E(d_s=32,l_s=15)	15.2	13.4	8461	8549
SSC-A(d_s=32,l_s=15)	10.5	7.6	8461	9041
1-NN(K=10)	39.0	42.1	390	390
1-NN-E(K=7)	22.0	19.0	273	346
1-NN-A(K=7)	16.1	11.2	273	796
LSC-E(K=10,D=4)	18.6	13.9	390	483
LSC-A(K=9,D=4)	13.5	8.1	351	895

also be seen that the results of the DTW classifier are clearly superior already in the non-adaptive case when compared with the other methods here.

The SSC classifier produced its best results when using $d_s = 32$ directions and the discretization distance of $l_s = 15$. The 'A' rule of adaptation produced only about one half of the *final* errors the 'E' rule made. The non-adaptive version's error rate was approximately twice that of the 'E' rule. The SSC classifiers were outperformed by the DTW-based method, but still performed better than the LSC classifier in both adaptive an non-adaptive cases.

In the LSC experiments the two proposed stroke thickening methods for feature extraction performed equally well. Therefore it was reasonable to use the first one as it was easier to implement and use. Therefore, Table 4.2 only displays results for the first thickening method and LSC. The dimensionality of feature vectors was selected experimentally by decreasing it gradually from 64, which was the dimensionality of data after the Karhunen-Loève transform. In the reduction process, feature components were discarded starting from those corresponding to the smallest eigenvalues. The best result with the non-adaptive 1-NN classifier was obtained when the dimensionality of the feature vectors was 45. This value was then used. The optimal value for K, the number of initial user-independent

prototypes per class, was selected individually for each method between 1 and 10.

The table first shows the result for the non-adaptive 1-NN classifier and then an adaptive 1-NN classifier when both the 'E' and 'A' adaptation strategies have been used. This classifier was formed similarly to the adaptive LSC classifier, i.e. user-independent prototypes were removed and user-dependent ones added. It can be seen that the LSC-based adaptive classifier outperforms the adaptive 1-NN classifier in both the 'E' and 'A' cases. For both classifiers, the 'A' strategy seems to be better than the 'E' strategy.

In all cases the adaptive classifiers are clearly better than the non-adaptive ones. It is also evident that DTW-based classifiers are clearly the most effective ones. This led us to focus on the DTW-based classifier and provide variation through the use of different distance measures and normalizations. Chronologically the experiments with diversity measures were performed at a much later stage in the research, and thus our original determination of taking forward the most promising set of classifiers was still the deciding factor. That has been the case for most of the adaptive committee experiments that are discussed in Chapter 5. Still, in many cases a diverse set of member classifiers can be more beneficial for classifier combining, and if we had looked into this direction of research at an earlier stage, the selection of the member classifiers for the adaptive committee evaluation might well have been different. However, the use of a single type of member classifiers does help to isolate classifier-specific factors from having an effect on the adaptive committee experiments, and as such that choice does have its merits. Diversity and its effects are discussed in more detail in Chapter 7.

Chapter 5

Adaptive committee classification

In addition to using individually adaptive classifiers, committee structures that are adaptive in themselves can be used. While an adaptive committee has in general significantly less information to work with than an adaptive classifier, the higher level of abstraction also provides some benefits. Among these benefits are applicability to a wide range of scenarios and the ability to provide adaptivity to classifiers that cannot be made adaptive themselves. Also, it is much easier to store and adjust the much smaller amount of information that is provided by the member classifiers.

It is possible to combine adaptive or non-adaptive member classifiers with adaptive committee structures. In any case, adaptation on the committee level is based on some level of consistency in the decisions of the member classifiers, and it is this consistency that can be learnt by the committee. Thus combining adaptive members is a much more challenging task than combining non-adaptive member classifiers.

Experiments with adaptive committee approaches combining both adaptive and non-adaptive member classifiers have been performed and some novel committee adaptation approaches and their performance will be discussed in the following sections. More details can be found in the included publications. It should also be noted that for all these methods it is assumed that information on the correct classification result can be used for committee adaptation.

The application throughout is on-line handwritten character recognition, al-
though other applications would of course be possible as well. Any applica-
tion where a significant amount of variation is present, but only a subset of it is
likely to be expressed is an ideal candidate for performance improvement through
on-line adaptation. At least most human interface applications fall within this
category, as each person tends to have a specific style of writing, talking or ges-
turing while practically infinite amounts of possibilities exist overall. However,
also other applicable scenarios, such as process control, are not difficult to en-
vision. So, the applicability of the presented methods is by no means limited to
the present application.

5.1   Levels of classifier information for combining

One fundamental attribute in defining a classifier combination method is how
much information the committee obtains from the member classifiers. For most
classifier combination methods information on the sample being classified is only
available through the outputs of the classifiers. Even though combination meth-
ods that use also features from the original data do exist, such committees are
definitely a minority.

We will work from the assumption that the committee classifier bases its decisions
solely on information provided by the member classifiers. For these situations,
three distinct levels of information should be considered. The levels of classifier
information were also discussed in Section 2.3.

The least informative output from a classifier is just the label the classifier sug-
gests. At this level, the classifiers produce no information with regard to how
confident they are in their decisions. The next level of additional information
for a classifier is to produce a list of possible classes, in order of preference. An
even more informative alternative is that the classifiers output information on
the measurement level – examples of measurement level information are mem-
bership or belief values and confidences or estimates of the posterior probabilities
of classes. This increase in information enables much more reliable estimation of
the classifiers' confidences in their classification results.

5.2 Adaptive implementations of committee classifiers

Some adaptive committee classifiers that have been used in the work of this thesis will be discussed and explained in this section, starting from the most simple adaptation rules. All the combination methods presented in this section share the feature that they are very simple adaptive extensions to a common approach. Also, all but the Decision Template method operate solely on label-level information from the classifiers. Later in Sections 5.3, 5.4 and 5.5 three novel adaptive committee structures developed will be presented.

5.2.1 Adaptive best

Perhaps the simplest form of committee adaptation is an adjusting best committee [105], which has been used in Publications 2, 4 and 5. The idea is to select the best classifier for each individual writer by evaluating the classifiers' performances during operation and using the result from the classifier that has performed the best up to that point. The performance evaluation is conducted by simply keeping track of correct recognitions obtained from each classifier. At any given time the committee's decision is thus the result from the classifier with the highest correct answer count at that point,

$$c(x) = c^j(x), \quad j = \arg\max_{k=1}^{K} N(\text{correct classifications for classifier } k), \quad (5.1)$$

with N(correct classifications for classifier k) being the count of correct recognitions for classifier k, $c^k(x)$ the class suggested by that classifier for the present input x and K is the total number of classifiers. In the case of a draw, the result from the classifier ranked higher on the classifier evaluation data set is used.

5.2.2 Adaptive voting

Another basic approach to adaptive committee decisions is to use a weighted variation of the original plurality voting rule [87]. Adaptation has been implemented by introducing weights based on a running evaluation of correctness for each voting classifier. The weight is in the form of the ratio between the count of correct classifications from a particular classifier and all classifiers,

$$w(k) = \frac{1 + N(\text{correct classifications for classifier } k)}{1 + \sum_{j=1}^{K} N(\text{correct classifications for classifier } j)}, \quad (5.2)$$

where $w(k)$ is the weight for classifier k. The addition of one in both the nominator and denominator has been made to avoid both zero weights and divisions by zero. The adaptive voting committee has been used in Publications 4 and 5.

With this weighting, the final plurality voting decision is obtained as

$$c_{wp}(x) = \arg \max_{c=1}^{C} \sum_{k=1}^{K} w(k)\Delta_k(x,c), \qquad (5.3)$$

where C is the total number of classes. $\Delta_k(x,c)$ is 1 if classifier k suggests the class c for sample x and zero otherwise. In practice this decision scheme sums up the confidences for each suggested label and selects the one with the highest overall confidence.

5.2.3   Adaptive Behavior-Knowledge Space

The Behavior-Knowledge Space (BKS) method [75], which was briefly discussed also in Section 2.3.3, is based on using a K-dimensional discrete space, with each dimension corresponding to the decision of one classifier. That discrete space is used to determine the class labels. The knowledge space is used by first finding the *focal unit* in the K-dimensional space, the unit which is the intersection of the classifiers' decisions for the current input. In the training phase the unit in the focal unit collects the count of recognitions and counts for each true class.

For recognition, the focal unit corresponding to the classification results of the member classifiers is first identified. Then if that unit has gathered samples, the class with the highest ratio is selected. In our experiments rejection is not used and thus the output of the committee is taken to be simply the class with the highest probability in the focal point – in practice the one that has received most samples. If the focal point has not received any samples, the default rule of using the highest-ranking classifier's result is used.

The BKS method can also be used in an adaptive fashion. In the experiments of Publication 5 the BKS was trained on a separate database and it made its decisions as in the non-adaptive case described above. Adaptation of the BKS committee was implemented through adding all classified samples to the knowledge space in a fashion identical to the training phase. The new samples added to the BKS during adaptive operation are considered to be of equal value as the stored training data and as such are also treated identically during classification.

5.2.4   Adaptive Decision Templates

Another committee classification method experimented with is the Decision Template (DT) method [98], which was discussed in Section 2.3.3. The main idea of the DT method is to create decision templates for each class from the training data, compare a decision profile computed for the input sample to the decision templates, and select the best-matching class.

Each sample x's decision profile DP_x is a $K \times C$ matrix consisting of the supports of all K classifiers for all C classes. The decision template DT_c for a class c is obtained by averaging over all the decision profiles of samples belonging to that class in the training data. For implementing the DT method with classifiers that do not natively produce supports but for example distances, it is first necessary to transform the distances produced by the classifiers into supports by some means. Initial experiments revealed that the DT method does not seem to be very effective when the variability range of the supports is very small, as would often be the case if simple scaling were used. Hence, a hyperbolic tangent function was used to scale the resulting supports.

In these experiments the supports were calculated by first finding the largest finite distance produced by the classifier k for sample x, denoted as $d_{max}^k(x)$. Then the distances to the nearest prototype of each class c from classifier k were scaled and transformed to supports as

$$s_c^k(x) = \frac{\hat{s}_c^k(x)}{\sum_{v=1}^{C} \hat{s}_v^k(x)}, \qquad (5.4)$$

where

$$\hat{s}_c^k(x) = \begin{cases} 1 - \tanh(15 \frac{d_c^k(x)}{d_{max}^k(x)} + 5) & \text{, if } d_c^k(x) \text{ is finite} \\ 0 & \text{, otherwise.} \end{cases} \qquad (5.5)$$

In the unlikely event that no distance $d_c^k(c)$ was finite, supports for all classes were deemed zero. The numerical values for the hyperbolic tangent scaling, the multiplier of the ratio $\frac{d_c^k(x)}{d_{max}^k(x)}$ and the addition constant, 15 and 5 respectively, were determined experimentally from experiments on the training data set.

The input sample's decision profile was compared to the decision templates using the measure of similarity $S_1(DP_x, DT_c)$ that performed best in experiments of [98]. With K classifiers and C classes the similarity measure between the decision profile DP_x for the input x and the decision template DT_c for class c is

calculated as

$$S(DP_x, DT_c) = \frac{1}{KC} \sum_{k=1}^{K} \sum_{v=1}^{C} \frac{\min(DP_x(k,v), DT_c(k,v))}{\max(DP_x(k,v), DT_c(k,v))}. \tag{5.6}$$

After the similarity of the input's decision profile is calculated to the decision templates of all classes, the most similar class is chosen. Adaptivity was introduced to the system via adapting the decision template of the correct class after recognition by inserting the classified input's decision profile into the decision template of the corresponding class. The classified samples were thus considered equally valuable as the training data samples in defining the decision templates.

5.3   Modified Current-Best-Learning

The Current-Best-Learning (CBL) algorithm [169] is a framework for learning general logical descriptions. The CBL algorithm works by maintaining a hypothesis and adjusting it as new examples arrive. The fundamental idea is to ensure that the hypothesis is consistent for all the examples that have been presented to the system.

The CBL algorithm has been modified in our experiments to function as a committee classifier. Even though the implementation differs significantly from the original CBL algorithm, the implemented committee is referred to as the Modified Current-Best-Learning (MCBL) committee due to the source of inspiration. The MCBL committee was introduced and used in Publications 2 and 5.

In the MCBL committee, the system can be described with a two-dimensional grid, with each column, $1, \ldots, K$, representing a member classifier and each row $1, \ldots, C$ corresponding to a particular class. The values stored in the grid are simple estimates for the confidence of a member classifier's decision for classifying an input to that particular class. Thus each point of the continuous hypothesis space corresponds to a particular combination of the confidence values in the grid. The operations of the CBL algorithm, specialization and generalization, now correspond to changing the confidence values, which is equivalent to moving the point of the current hypothesis in the hypothesis space.

The decision of the committee is simply that member classifier's result which has the largest class-wise confidence value $f^k(c^k(x))$,

$$c_{mcbl}(x) = c^j(x), \quad j = \arg\max_{k=1}^{K} f^k(c^k(x)), \tag{5.7}$$

where k is the index of the classifier and $c^k(x)$ the class suggested by that classifier for the input x.

When forming the class-wise confidence values the MCBL committee uses measurement-level information from the member classifiers. In Publications 2 and 5 we used distances $d_a^k(x)$ and $d_b^k(x)$ which in a prototype-based classifier k are the distances from the input calculated to the nearest prototypes in the first and second-ranked classes, respectively. From those, we can calculate

$$l^k(x) = 1 - \frac{d_a^k(x)}{d_a^k(x) + d_b^k(x)}, \tag{5.8}$$

which is an increasing function of the unambiguity of the single classifier's decision.

By combining the values $l^k(x)$ into class-wise confidence values $f^k(c^k(x))$, a table consisting of each classifier's classification result and its confidence can be formed. To modify the hypothesis, the values $f^k(c^k(x))$ are adjusted when the committee as a whole is incorrect. For all classifiers k of the committee that are correct, the $l^k(x)$ value for that classifier is added to the confidence of the class for that classifier. Let us use the notation $c^{\text{true}}(x)$ for the true class of sample x. When a classifier produces an incorrect result, its confidence for that class is reduced by multiplying it with the value $l^k(x)$. The value $f^k(c^k(x))$ is always reduced through multiplication as $1/2 \leq l^k(x) < 1$ for all $l^k(x)$. This can be formulated as

$$\forall k \in \{1, \ldots, K\} : f^k(c^k(x)) := \begin{cases} f^k(c^k(x)) + l^k(x), \text{ if } c^k(x) = c^{\text{true}}(x) \\ f^k(c^k(x)) \cdot l^k(x) \text{ , otherwise.} \end{cases} \tag{5.9}$$

When the committee produces a correct result, the current hypothesis is considered to be effective and no changes are made. A reasonable initialization for the confidence values was found to be the inverse of the ordering of the classifiers according to their decreasing recognition performance, i.e. $f^k(c_j) = \frac{1}{k}$ for all classifiers k and class labels c_j, where the indexing of k corresponds to the classifiers' ordering.

5.4   Dynamically Expanding Context

The Dynamically Expanding Context (DEC) algorithm was originally introduced as a method for correcting coarticulation effects between adjacent phonemes in

speech recognition [91, 92, 199]. The method can be formulated as a set of context-sensitive production rules $L(A)R \rightarrow (B)$, where A and B are the input and output symbols, and L and R are the left and right contexts of the input symbol. The combined length of the L and R contexts is referred to as the level of the rule. Each time a rule is found to be in conflict with the actual transformation needed for correct output, a new higher-level rule is added.

For the setting of classifier combining, the DEC principle has been adjusted to use a set of classifier outputs as a one-sided context to create production rules to correct classification errors. We have used the DEC committee in [105, 2] as well as in included Publications 2, 3, 5 and 7.

In practice, the member classifiers are first initialized and ranked in order of decreasing performance. Then the result of the best-ranked classifier is taken as the input, and the results of the remaining classifiers are taken as a one-sided context for the input. Second-best results from the classifiers can be used too, thus categorizing the DEC committee as a method based on ordered results from its members. More than two results from each classifier could be used as well, but for our experiments we limited the context size to be taken from each classifier to the two highest-ranked results.

The DEC rules can be written symbolically as

$$(A)R \rightarrow B, \tag{5.10}$$

where $(A)R$ is a string of member classifier outputs, with (A) being the result from the highest-ranked classifier and r the one-sided context formed from the outputs of the rest of the members. The output symbol B is the desired recognition result.

When a new character is presented to the system, the input (A) along with the context R, in practice the list of member classifiers' outputs, is searched for in the existing rule base. If no matching rule is found, the default decision is applied. This default decision can be for example to use the first output of the best ranked classifier or to output a plurality voting result.

If more than one rule matches the input, the highest-level one, i.e. the most specific one, or equivalently, the one with the largest context, is used. If the recognition result is then found to be incorrect, a new rule with more context is added to the rule base. A schematic diagram of the DEC committee is presented in Figure 5.1.

As new DEC rules are being added, all the available context information will eventually be used by the rules. All error situations thereafter would call for
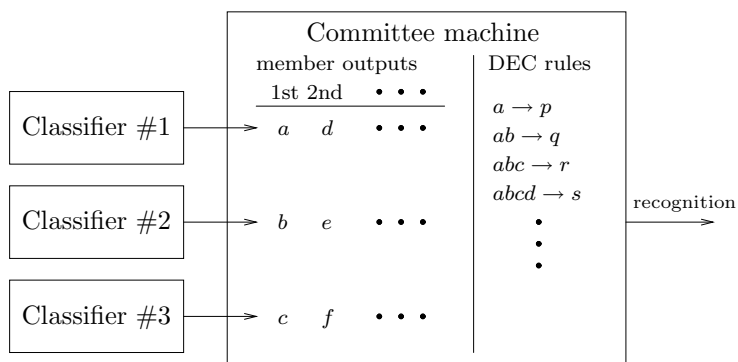
Figure 5.1: A block diagram of the DEC committee

additional rules, but the context cannot be expanded anymore. Therefore, it is allowed that there exist more than one highest-level rule for the same context. In this case, there are some methods for selecting the rule to use. First, the number of correct applications of each such rule can be maintained and the rule with the highest correctness value applied. Second, also the count of incorrect applications may be taken into account by subtracting them from the count of correct applications in order to select one of the conflicting rules. Third, a rule may be inactivated upon an incorrect result. An inactivated rule will be reactivated if a situation arises where the committee was incorrect but that rule would have produced the correct result.

In practice it often seems to be beneficial to require the output symbol (B) to be included in the context $(A)R$, which in fact means that the output of the committee must be one of its members outputs. This serves to hinder the creation of erroneous rules from bad inputs, but also makes it impossible for the committee to learn to correct situations where all classifiers are incorrect. Thus the benefit of this requirement is highly dependent on the task and member classifier set.

5.5   Class-Confidence Critic Combining

In critic-based approaches there is usually a separate expert that makes a decision on whether the classifier it is examining is correct or not. This decision must naturally be made based on information obtainable on the classifier. In our Class-Confidence Critic Combining (CCCC) approach the focal idea is to try to produce as good an estimate as possible on the classifier's correctness based on its
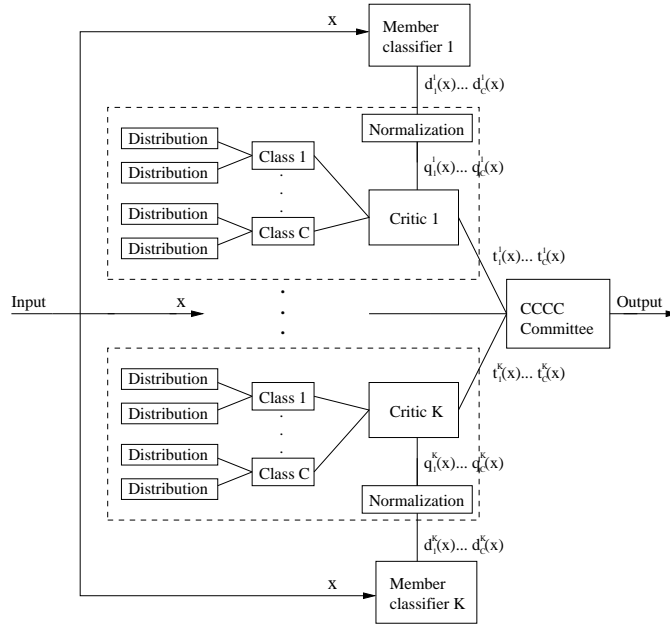
Figure 5.2: A block diagram of the CCCC committee

prior behavior for the same character class. The method produces a confidence value which is used to make the final decision.

In practice, the classifiers' behavior is modeled via collecting normalized distance values from the classifier and storing these values in distribution models. Thus the CCCC committee works with classifiers outputting measurement level information as well. The distribution models are then used to evaluate the critics' confidence in a particular decision, and the final decision on the output label is made based on these confidences. The CCCC method has been introduced, used and improved in Publications 4, 5, 6, and 8. The system is described here in its entirety as it is perhaps the most important contribution of this thesis, and also the various improvements and additions to the framework have appeared in different publications, none of which encompassed all the implemented possibilities. A schematic diagram of the CCCC committee structure is shown in Figure 5.2.

5.5.1   Distance normalization

Let there again be K classifiers, each of which uses some distance measure and determines its output by minimizing that distance. If the classifiers were to produce some evaluations of confidence or probability, those may be simply translated to values that decrease as similarity increases. For example, if we have a confidence measure $t \in [0, 1]$, we may simply use $1 - t$ as the distance. Hence we will here assume that a metric is used that decreases as similarity increases, a distance measure d.

As before, let there be C pattern classes and let it be possible for each classifier to produce a distance value for every class. This would mean for example for a prototype-based classifier that it has at least one prototype for each class. Let x be the current input sample, k the classifier index, $k \in \{1, \ldots, K\}$, and c the class index, $c \in \{1, \ldots, C\}$. Even though it would be desirable for the classifiers to output distances for all classes, that is not always the case. For example the DTW-based classifiers we have used, described in Section 4.4.1, will output an infinite distance if matching to no prototype of that class is possible. As the classifier is based on stroke-wise matching, this happens whenever the number of strokes differs between the input sample and all prototypes of a particular class. Thus in each classifier we may find the shortest distance to the nearest prototype of each class, $d_c^k(x) \in [0, \infty]$. As the possibility of infinite distances exists, we normalize the distances as

$$
q_c^k(x) = \begin{cases} \dfrac{d_c^k(x)}{\sum_{v=1}^{C} \hat{d}_v^k(x)} & , \text{ if } d_c^k(x) \text{ is finite} \\ 1 & , \text{ otherwise} \end{cases} , \tag{5.11}
$$

where $\hat{d}_c^k(x)$ used in the summation equals $d_c^k(x)$ if it is finite and is otherwise zero. However, if the distance to only one class is finite, the normalized distance to that class is defined to be zero, as the only possible class should naturally be the one chosen. Also some other scaling approaches have been experimented with in Publication 5, but they can be seen as special cases of equation (5.11) that are produced by restricting the examined classes to be either the single or two nearest classes. Later experiments have shown that the normalization is in general beneficial due to the abovementioned reasons, and hence it will be assumed to be always in use here.

5.5.2   Distribution types

In order to obtain confidences for decisions on previously unseen x, the distribution of the $q_c^k(x)$ values must be modeled in some way. The approach used here is to collect the previous values into distribution models from which a value for the confidence can be obtained as a function of $q_c^k(x)$. One key point in the effectiveness of a scheme based on confidence values calculated from distribution models is naturally the ease of creating and modifying the distribution models. The amount of data that is obtained from each true distribution for the creation of the models is quite limited, and in a real situation may vary greatly between distributions due to the fact that some classes occur much more frequently than others. Thus the methods should be capable of producing reliable estimates even with small amounts of data. We have experimented with a number of distribution models in Publications 4, 5, 6, and 8, and they are explained below.

The notation used is that the shorthand distribution index i runs over the distributions of both correct and incorrect classifications for each class c in each member classifier k. Each distribution model i contains N_i previously collected values $q_c^k(x)$, for which we use the shorthand notation $z_j^i, j = 1, \ldots, N_i$. The notation for the confidence obtained from the distribution model i stands as $p^i(q_c^k(x))$. For shortening the notation further, we shall use $q_c^k(x) = z$. The index k is used for indicating both the member classifier and the critic, as there is always exactly one critic paired with a member classifier. The weight assigned to each sample by the critic is denoted with $w_i(z_j^i)$. When no weighting scheme is in use, the constant weight of one is used for all samples, $w_i(z_j^i) = 1, \ \forall i, j$.

Gaussian normal distribution: The Gaussian normal distribution is applied through calculating the variance and mean from the already obtained samples and then calculating the values for the confidences from a Gaussian normal distribution $p_{\text{gaussian}}^i(q_c^k(x)) = p_{\text{gaussian}}^i(z)$ where,

$$p_{\text{gaussian}}^i(z) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(z-\mu_i)^2}{2\sigma_i^2}}, \tag{5.12}$$

where μ_i is the mean and σ_i^2 the variance estimated for the distribution indexed with i. Initial values are used for the mean when no samples exist and for the variance when less than two samples have been received for the particular distribution.

Non-parametric distribution: The non-parametric model is based on calculating the number $n_f(z, i)$ of points in the distribution that are further from the

mean of the distribution μ_i than the the input z,

$$n_f(z, i) = \sum_{j=1}^{N_i} v(z, i, j), \quad v(z, i, j) = \begin{cases} 1, \text{ if } |z - \mu_i| < |z_j^i - \mu_i| \\ 0, \text{ otherwise} \end{cases} . \quad (5.13)$$

The confidence is then based on the ratio between $n_f(z, i)$ and the total number of points in the distribution N_i so that

$$p_{\text{nonparam}}^i(z) = \frac{n_f(z, i)}{N_i}. \quad (5.14)$$

Nearest Neighbor approach: The nearest neighbor (NN) rule is used by calculating the distance from the input value z to the nearest value already in the distribution i containing N_i values z_j^i,

$$p_{\text{NN}}^i(z) = 1 - \min_{j=1}^{N_i} |z - z_j^i|. \quad (5.15)$$

Triangular kernel distribution estimate: The triangular kernel distribution estimate is formed through the use of a triangular kernel function,

$$p_{\text{trikernel}}^i(z) = \frac{1}{\sum_{j=1}^{N_i} w_i(z_j^i)} \sum_{j=1}^{N_i} w_i(z_j^i) \max\{0, (b - |z - z_j^i|)\}. \quad (5.16)$$

defined by the bandwidth b. The estimate is thus calculated by applying a kernel over all N_i data points z_j^i in the distribution i and normalizing.

Gaussian kernel distribution estimate: The Gaussian kernel distribution is estimated through the use of a Gaussian function with variance of b as the kernel. The evaluation of the distributions' values at specific points is performed as for the triangular kernel,

$$p_{\text{gausskernel}}^i(z) = \frac{1}{\sum_{j=1}^{N_i} w_i(z_j^i)} \sum_{j=1}^{N_i} w_i(z_j^i) e^{-\frac{(z - z_j^i)^2}{2b}}. \quad (5.17)$$

Exponential kernel distribution estimate: The two-sided exponential kernel distribution estimation is performed using an exponential kernel with the kernel bandwidth b,

$$p_{\text{expkernel}}^i(z) = \frac{1}{\sum_{j=1}^{N_i} w_i(z_j^i)} \sum_{j=1}^{N_i} w_i(z_j^i) e^{-\frac{|z - z_j^i|}{b}}. \quad (5.18)$$

5.5.3   Weighting schemes

A weight can be assigned with each distance value stored in the critic's distribution model to facilitate emphasizing newer inputs. For the second phase of adaptation, the weights will be modified to obtain more robust behavior. Three approaches for adjusting the weights of the sample points have been used, along with the constant weighting scheme used for reference, in Publications 6 and 8.

Constant weights: The weight for each sample is constant,

$$w_i(z_j^i) = 1, \ \forall i, j. \tag{5.19}$$

Class-independent weights: The weights are initially set in an increasing order by using an increasing counter (sample index) $n(z_j^i)$ scaled with a suitable constant. If known beforehand, the total number of test samples N can be used for the scaling factor to obtain the weights

$$w_i(z_j^i) = \frac{n(z_j^i)}{N}. \tag{5.20}$$

These weights do not depend on the distribution model the sample is inserted into, so within each distribution there can be large differences in the weights.

Class-dependent weights: For each distribution, the weights are scaled linearly every time a new sample is inserted. As a result, each sample has weight equal to the ratio of its index $n_i(z_i^j)$ in that particular distribution model i and the total number of samples in that model, N_i,

$$w_i(z_j^i) = \frac{n_i(z_j^i)}{N_i}. \tag{5.21}$$

This results in the first sample having the smallest weight of $1/N_i$ and the most recent sample having the weight $N_i/N_i = 1$.

Decaying weights: When a new sample is inserted to the distribution model, the weights are recalculated to decrease in accordance with a decay constant $\lambda \in [0, 1]$ so that

$$w_i(z_j^i) = \max\{0, 1 - \lambda(N_i - n_i(z_j^i))\}. \tag{5.22}$$

Effectively the inverse of the decay constant λ states how many previous samples the distribution "remembers" at any given time point, with the newest samples being given the most weight.

5.5.4 Combining confidence values

Now we obtain two confidences, one from the distribution of correct classifications denoted $p^{\text{correct}}(q_c^k(x))$ and the other from the distribution of incorrect classifications, $p^{\text{incorrect}}(q_c^k(x))$. The first approach to obtaining the classification confidence $u_c^k(x)$ given by critic k to the classification result $c^k(x)$ of classifier k is to just use the confidence from the correct distribution as the overall confidence,

$$u_c^k(x) = p^{\text{correct}}(q_c^k(x)). \tag{5.23}$$

Equation (5.23) was used in experiments of Publications 4, 5, 6, and 8.

The second approach experimented with, applied in Publications 4 and 5, is to use both the correct and incorrect classification result distribution confidences $p^{\text{correct}}(q_c^k(x))$ and $p^{\text{incorrect}}(q_c^k(x))$ by subtracting them from one another,

$$u_c^k(x) = p^{\text{correct}}(q_c^k(x)) - p^{\text{incorrect}}(q_c^k(x)). \tag{5.24}$$

It should however be noted that equation (5.24) can result also in negative confidences.

For non-adaptive member classifiers additional robustness is usually not necessary and hence the classification confidence $u_c^k(x)$ may be directly used as the overall confidence $t_c^k(x)$ given by the critic k for the input x belonging to class c,

$$t_c^k(x) = u_c^k(x). \tag{5.25}$$

However when more emphasis needs to be put on the performance of the classifiers, the overall confidence $t_c^k(x)$ can also be obtained by weighting the classification confidence $u_c^k(x)$ with a running evaluation of the classifiers' overall correctness rate as suggested in Publication 6. This rate $p(\text{classifier } k \text{ correct})$ is obtained by tracking how many times classifier k has been correct so far and dividing that by the total number of samples classified. Hence the overall confidence can be written as

$$t_c^k(x) = u_c^k(x) \cdot p(\text{classifier } k \text{ correct}). \tag{5.26}$$

If still more robustness is desired, for example when combining member classifiers that are adaptive by themselves, the original normalized distance value can be taken into account as well. In practice, to obtain the final confidence, the result of (5.26) is multiplied by one minus the normalized distance value derived from the classifier itself, $1 - q_c^k(x)$. This strategy was introduced in Publication 8 and

is reasonable as the distance value obtained from the classifier can also be seen as an indication of the classifier's own current confidence in its output. Hence in this case the overall confidence $t_c^k(x)$ becomes

$$t_c^k(x) = u_c^k(x) \cdot p(\text{classifier } k \text{ correct}) \cdot \left(1 - q_c^k(x)\right). \tag{5.27}$$

5.5.5 Decision mechanisms

For determining the final decision, a decision rule that is capable of taking advantage of the confidences provided by the critics should be used. The decision rule will take the confidences and attempt to select the best overall result. The *product*, *sum*, *min* and *max* rules were introduced in Publication 6, but also the weighted voting rule used in Publications 4 and 5 can be seen as an application of the sum rule, as well as the maximum selection of those two publications being an application of the max rule.

Product rule: For each class, the confidences of the critics are multiplied together, and then the class with the largest total confidence is chosen,

$$c_{\text{prod}}(x) = \arg\max_{j=1}^{C} \prod_{k=1}^{K} t_j^k(x). \tag{5.28}$$

Sum rule: For each class, the confidences of the critics are summed together, and then the class with the largest resulting confidence is selected,

$$c_{\text{sum}}(x) = \arg\max_{j=1}^{C} \sum_{k=1}^{K} t_j^k(x). \tag{5.29}$$

Min rule: For each class, the smallest confidence from a critic is used, and then the class with the largest minimum confidence is chosen,

$$c_{\text{min}}(x) = \arg\max_{j=1}^{C} \min_{k=1}^{K} t_j^k(x). \tag{5.30}$$

Max rule: For each class, the largest confidence from a critic is discovered, and then the class with the largest maximum confidence is selected,

$$c_{\text{max}}(x) = \arg\max_{j=1}^{C} \max_{k=1}^{K} t_j^k(x). \tag{5.31}$$

Modified Current-Best-Learning decision: The MCBL rule, as described in Section 5.3, has also been applied as a decision scheme for CCCC in Publications 4 and 5. Here the confidences originally from equation (5.8) were simply replaced with the overall confidences obtained from the critics, $t_c^k(x)$, to obtain the MCBL class-wise confidences $f^k(c^k(x))$. The MCBL decision mechanism takes into account only the confidence for the class that the critic is most confident in.

Prior to the final decision, the obtained confidences were still modified by adding the critic's confidence value $t_c^k(x)$ to the MCBL confidence value when the critic had a positive confidence and multiplying them if the critic's confidence was negative,

$$t_{\mathrm{mcbl}}^k(x) = \left\{ \begin{array}{ll} f^k(c^k(x)) + t_c^k(x) & , \text{ if } t_c^k(x) > 0 \\ f^k(c^k(x)) \cdot t_c^k(x) & , \text{ otherwise} \end{array} \right. . \qquad (5.32)$$

This last step mirrors the logic behind equation (5.27). This MCBL modification scheme was used as it was the one found to produce best results from a number of schemes experimented with. It should be noted that $t_c^k(x)$ can indeed be negative if equation (5.24) is used for combining the correct and incorrect confidences.

For the final decision from $t_{\mathrm{mcbl}}^k(x)$, both selecting the result based on the maximum value as in equation (5.31) and a scheme using the sum rule of equation (5.29) were experimented with in Publications 4 and 5, but any of the equations (5.28) – (5.31) could be used.

5.6   Committee performance

Although experiments with the committee approaches used have been presented in Publications 2, 4, 5, 6 and 8, an additional set of experiments has been performed for the purpose of being able to easily compare all the presented committee methods at once. Here they all have been evaluated in an identical setting on the same database as will be described in Sections 5.6.1–5.6.4. This is in fact a superset of the experiments of Publication 8 with now all the presented methods being evaluated in the same setting. Some conclusions from the experiments will be discussed in Section 5.6.5.

This discussion will be restricted to an empirical evaluation of the presented methods. The analytical evaluation of the presented committee methods has proven quite difficult due to the complex nature of the presented combination methods, especially without making simplifying assumptions that would lessen the applicability to real-life situations. Thus the analytical evaluation has been left out

from the scope of this thesis, but would definitely provide a very interesting topic for further work.

5.6.1   Experimental setup

The experiments were performed using a total of six different classifiers formed through the combinations of the distance measures and normalization techniques for the DTW-based classifiers described in Section 4.4.1. These classifiers have been used because they have been found to provide very impressive benefits through adaptation when used as individually adaptive classifiers, while also performing very well without adaptivity. Combining effective adaptive classifiers was considered the most difficult scenario for obtaining further improvement through adaptive combination, and thus the performance in this setting should be of much interest. Experiments with more diverse classifiers and the effects of diversity on combination performance will be discussed in Chapter 7.

The data used in the experiments were on-line handwritten characters written one-by-one. Data collection and preprocessing are covered in detail in [215]. The experiments were performed as batch runs on previously collected data. Thus there was no feedback from the user to the recognition system, and it was assumed that the true classes of all characters were known for the adaptation. All upper and lowercase letters, including the three Scandinavian diacriticals 'å', 'ä' and 'ö', and digits were used as 68 pattern classes.

It should be noted that the character set used for these experiments consisted of all 68 pattern classes whereas the data used for the experiments of Section 4.6 only included the lowercase letters and digits, a total of 39 character classes. Thus also the performances of the individual classifiers seem lower in these experiments. The use of all character classes was deemed appropriate as the committee classifiers will enhance the performance notably.

The data used was the three independent databases described in Section 4.1. Database 1 was used for forming the initial user-independent prototype set for the DTW-based member classifiers. The prototype set consisted of seven prototypes per class. Database 2 was used for estimating the values for the necessary numeric parameters for the committee and determining the performance rankings of the classifiers. Classifiers were always used in order of their rankings based on their non-adaptive performance on Database 2. Database 3 was used as an independent test set. In the experiments the adaptive member classifiers were reset to their user-independent initial state in between writers. The performances

Table 5.1: Single classifier results

Member classifier	Error percentage	
	non-adaptive	adaptive
point-to-point, mass center	20.02	9.87
point-to-point, bounding box	21.18	9.90
normalized point-to-point, mass center	20.93	10.24
normalized point-to-point, bounding box	21.18	10.70
point-to-line, mass center	20.77	15.56
point-to-line, bounding box	22.28	16.27

of the member classifiers, in adaptive and non-adaptive configuration, are shown in Table 5.1.

5.6.2   Committee configuration

In addition to the seven adaptive committee structures described in sections 5.2–5.5, also three non-adaptive committee structures, namely plurality voting, Behavior-Knowledge Space and Decision Template methods, have been used. Thus a total of ten committee structures were applied to combine both non-adaptive and adaptive member classifiers, and they appear in Tables 5.2 and 5.3.

The *Best member* refers to using the overall best-performing member classifier for establishing a performance baseline. The best classifier in this case is the point-to-point and mass center based DTW classifier on the first row of Table 5.1. The *Adaptive best* selection discussed in Section 5.2.1 keeps track of the correct recognitions of each member classifier and uses the classifier deemed best at the time.

The *Plurality voting* rule is a commonly applied committee method. It was discussed in Section 2.3.3 and is used here as a reference for the adaptive committees' performances. The use of the voting committee also illustrates the feasibility of combining the adaptive member classifiers with a non-adaptive combination method, and the level of performance enhancement achievable in this manner. The plurality voting rule applied is simply to choose the label with most occurrences among the outputs of the member classifiers as the output of the committee. In the case of a tie, classifiers are iteratively pruned away starting with the last ranked classifier until a non-tie result is obtained. In the *Adaptive voting* committee of Section 5.2.2 the correctness of each classifier is tracked and weighted voting is performed based on the classifiers' performances so far.

The *BKS* committee, described in Section 2.3.3, was trained with the data from Database 2 and operated in a writer-independent fashion. The *Adaptive BKS* variant of Section 5.2.3 simply stores all performed recognitions into the BKS in the same manner as in the training phase and uses them along with the original training data for further recognitions.

The *DT* committee, also described in its basic form in Section 2.3.3, was trained with the data from Database 2 and operated in a writer-independent fashion. The DT committee bases its decisions on measurement-level information from the member classifiers. The *Adaptive DT* committee, presented in Section 5.2.3, performs in an adaptive fashion via adapting the decision template of the class the input sample belongs to after recognition. The classified samples are thus treated identically with the training samples.

The *MCBL* committee from Section 5.3 simply operated on-line in a user-dependent fashion, collecting data while the recognition proceeded. The classification result was chosen by selecting the class with the highest confidence according to equation (5.7).

The *DEC* committee, described in Section 5.4, was applied through using plurality voting as the default decision rule and requiring the output to be included in the input. Inactivation of error-producing rules was used as the conflict solution mechanism. The rule sets created were cleared between writers.

The *CCCC* committee, discussed in Section 5.5, collected only the distributions of the correct classifications with an exponential model as in equation (5.18). The effect of the most recent sample was emphasized by using the final confidence mechanism of equation (5.27). The committee decision was based on the sum rule of equation (5.29), and the numeric parameters for the CCCC committee were established as a kernel bandwidth of $b = 0.003$ and weight decay constant $\lambda = 0.18$ through evaluation on Database 2. The decaying weights scheme of equation (5.22) was applied to further enhance robustness. The distributions were not reset between writers, as the use of the decaying weighting model causes older data to be ignored eventually in any case.

5.6.3   Performance with non-adaptive member classifiers

In Table 5.2 the performance of the classifier combination methods with non-adaptive member classifiers is examined. The overall error rate has been calculated over all the samples, while the tail error rate shows the performance for the last 200 samples of each writer. The performance of all the member classifiers

Table 5.2: Classifier combination results in error percentages with non-adaptive members

Committee	Section	Errors	Tail errors
CCCC	5.5	15.53	15.25
DEC	5.4	17.56	16.25
Adaptive DT	5.2.4	17.35	16.50
Adaptive BKS	5.2.3	18.88	17.06
MCBL	5.3	19.32	17.12
DT	2.3.3	18.26	18.31
Adaptive voting	5.2.2	19.76	19.13
Plurality voting	2.3.3	19.68	19.19
Best member	4.4.1	20.02	19.19
Adaptive best	5.2.1	20.09	19.06
BKS	2.3.3	20.72	20.25

used can be seen in the column of Table 5.1 entitled "non-adaptive error rate", and the best classifier obtained an overall error rate of 20.02%. The development of the error rates, represented through a moving average with a window size of 400 samples is shown in Figure 5.3. Note that due to the stronger smoothing used for clarity in Figure 5.3, the resulting tail errors do not exactly match those appearing in Table 5.2.

In Figure 5.3 the improvement due to the users adapting their writing style to the recognizer can be seen as decreasing error rates for also the non-adaptive committee structures. With an interactive device, it is natural to prefer a writing style that produces fewer errors. This also showcases the changing nature of writing, as even though the context did not change, most writers changed their style slightly to better suit the system. However, the opposite phenomenon can be seen at the very end, where the accuracies of the classifiers decrease. One can only speculate that the total number of characters to input for each writer in one session was slightly excessive, and motivation started to decrease near the end.

Firstly, the BKS committee performs rather poorly, although it does outperform all but the single best member classifier. Evidently the errors caused by the writers in Database 2 were not consistent enough to produce significant benefits in the classification performance for this non-adaptive combination scheme. Also the tail error rate is the worst of those in Table 5.2.

The adaptive best classifier scheme would initially seem to be outperformed by the single best member, but looking at the tail error rates we can see that using

the adaptive best scheme we can obtain better tail error rates. This is possible because the best classifier overall was not the best classifier for every writer, and this adaptive scheme aims to discover the truly best member classifier for each particular writer. This behavior is also evident in the graphs, as when comparing the best classifier (solid) and adaptive best (dash-dot) lines it can be seen that initially the adaptive best performs notably poorer as there has not yet been enough data from the users to provide a good evaluation on which classifier is best. However, as more samples are classified, that evaluation improves and the classifier that is truly best for that particular writer is given preference, resulting in better performance in the end.

This leads us to conclude that even this simplest adaptive mechanism can produce gains after the period of adjustment. The same can be seen in the voting approaches: again, even though the plurality voting approach produces a lower overall error rate, the adaptive voting obtains a better tail error rate.

With non-adaptive member classifiers the DT committee performs very well, producing an overall error rate better than the MCBL committee even though its tail error rate is slightly worse. Although the combination scheme is very effective, the use of distance information may be a deciding factor in making DT the best-performing non-adaptive committee. The MCBL committee is clearly capable of improving performance. The fact that the tail error rate is again notably better than the overall error rate is evidence that also this method of adaptation is effective. The adaptive BKS committee provides still somewhat better results. When examining the performance of these combination methods, it can be seen from Figure 5.3 that while the MCBL committee is better on the early samples, the adaptive BKS significantly improves performance after enough samples have been collected. This may be due to the adaptive BKS having all the previous data stored and hence being less robust to change, as all samples are considered equally reliable in the implemented scheme.

Practically in a league of their own are the two more sophisticated adaptive techniques, the DEC and CCCC committees. Both provide significantly more improvement than the schemes above. The DEC committee can be seen to provide consistent decrease in the error rate, but the curve for the CCCC committee is much steeper due to its inherent generalizational traits; a sample input into the distribution has always significant effect on future confidence evaluations and the influence of new inputs is emphasized as the older samples get discarded due to the adaptation scheme. Hence, even though the starting error rates are both rather similar, the resulting error rates for the CCCC committee are significantly better due to the fast adaptation.
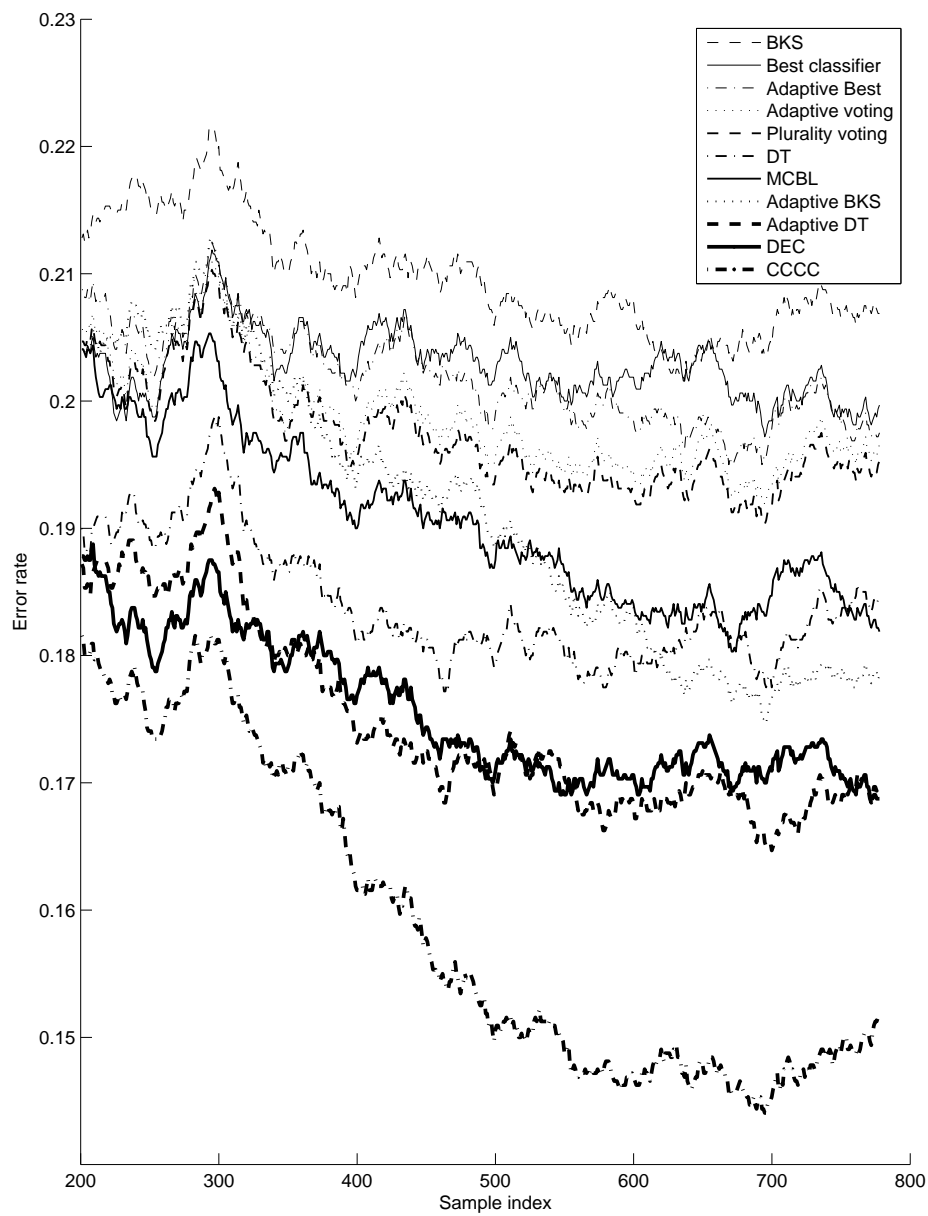
Figure 5.3: An illustration of the development of the error rate with non-adaptive member classifiers

Table 5.3: Classifier combination results in error percentages with adaptive members

Committee	Section	Errors	Tail errors
CCCC	5.5	7.97	4.13
Plurality voting	2.3.3	8.69	4.81
DEC	5.4	8.91	4.94
Adaptive voting	5.2.2	8.75	5.12
Adaptive best	5.2.1	10.00	5.75
Best member	4.4.1	9.87	5.94
Adaptive DT	5.2.4	9.36	6.38
Adaptive BKS	5.2.3	10.76	6.69
MCBL	5.3	12.17	6.94
DT	2.3.3	11.00	8.31
BKS	2.3.3	11.73	8.44

5.6.4   Performance with adaptive member classifiers

The results of the committee evaluations with adaptive member classifiers have been collected into Table 5.3. There again the overall error rate has been calculated over all samples and the tail error rate over the last 200 samples. Also here the best classifier corresponds to the point-to-point mass center DTW classifier on the first line of Table 5.1. Figure 5.4 illustrates the development of the error rates for all the evaluated methods, calculated with a moving average of window size 400. Again, please note that due to the stronger smoothing applied in Figure 5.4, the visible tail error rates are not fully compatible with those in Table 5.3.

Firstly it can be seen from Tables 5.3 and 5.1 that the MCBL committee, which was quite effective with non-adaptive member classifiers, fails to provide any improvement on the majority of the member classifiers' error rates in the overall performance. Also both the BKS approaches perform distinctly poorly with adaptive member classifiers, as do both the DT committees. These five classification schemes share the idea of basing their decisions on prior performance with a certain combination of outputs from the member classifiers, which is evidently not an effective strategy with adaptive member classifiers. The results of the adaptive member classifiers change with time, hence making predictions based on the combination of all their earlier outputs much less reliable. The committee strategies should clearly not focus too strictly on all prior classifications when dealing with adaptive classifiers, as predictions may become fragile as the

classifiers' behavior changes.

The behavior of the adaptive best scheme is very similar to that with non-adaptive classifiers, as is to be expected. Again in the overall rate it is outperformed by the best individual classifier, but in the tail error rate that is no longer the case. This simple strategy seems to be on the long run as viable for adaptive classifiers as it is for non-adaptive ones.

The overall error rate for the DEC committee produces improvements over all adaptive member classifiers, and even more clearly with the tail error rates. The rule-based nature of the DEC committee seems to be capable of successfully learning the behavior of the members, but is still slightly hindered by the effect of the adaptation altering the performance of the member classifiers.

Both the traditional plurality voting scheme and its adaptive counterpart perform notably well with adaptive classifiers. The basic, non-adaptive plurality voting rule is even more effective, and once again the strength of this simple combination strategy is obvious. Without making any effort to base its decisions on anything but the current outputs of the classifiers, the voting approach is clearly effective when the member classifiers are performing well.

The best performance is again provided by the CCCC committee. It obtains benefit from both a viable initial strategy and effective adaptation that is robust towards change, and simply outperforms all the other methods from start to finish, as can be seen in both the overall and tail error rates of Figure 5.3 and the error development in Figure 5.4.

5.6.5 Concluding remarks on the experiments

It is evident that classifier combination methods can produce significant performance improvements, especially with the non-adaptive member classifiers. However, obtaining improvements in a situation where the constantly changing behavior of adaptive member classifiers makes evaluating the classifiers' performance much more difficult is far from easy.

Some classifier combination methods that perform very well for non-adaptive classifiers, such as the MCBL, DT, adaptive DT and adaptive BKS committees, are clearly unsuitable for combining adaptive classifiers. This is most probably due to their high level of dependency on all prior data collected and lack of robustness. On the other hand, the plurality voting rule, a strategy that has no memory for the prior behavior of the classifiers, is very effective in both settings.
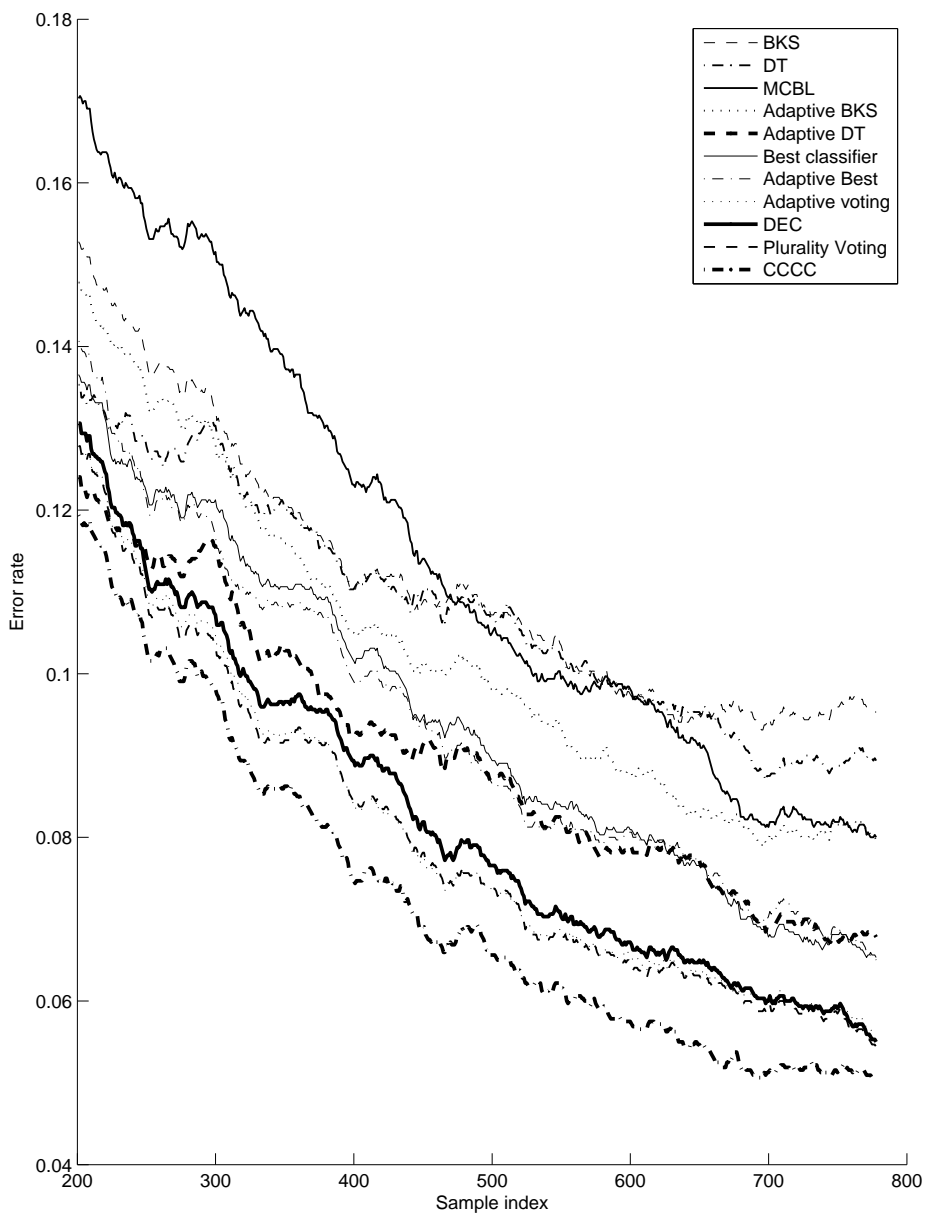
Figure 5.4: An illustration of the development of the error rate with adaptive member classifiers

Especially with adaptive member classifiers the use of minimal assumptions seems to be a viable strategy.

Additionally, it may be noted that the DT committee outperformed the BKS committee in both sets of experiments. The decision rules have fundamental differences, but in addition the DT method uses measurement-level information. This finding supports the assumption that increasing the amount of information is beneficial for the combination process.

In Publication 8 we also examined the statistical significance for the presented results. The standard error remained reasonably large due to the fact that the test set consisted of data from only eight writers, and the performances of those writers varied greatly. In practice two writers, out of the eight in the database, had notably worse error rates than the others and thus evidently caused the increase in variation.

Nevertheless, the best results can be obtained with a strategy that combines the benefits of both types by using information on the earlier behavior of the classifiers and being robust to change. This has been accomplished with the CCCC committee. As the focal result we note that the CCCC committee is clearly an approach that can successfully be applied in both settings. The CCCC committee produced the best performance in both sets of experiments, as was the case also in the experiments of the included Publications 4, 5, 6 and 8.

Chapter 6

Rejection with a committee

In addition to the development of the adaptive committee classifiers, two related issues have been studied as well. The first of these is rejection with a committee classifier, to be discussed in this chapter. The second, classifier diversity for committee member selection, will be discussed in Chapter 7.

Basically, the objective of rejection is to detect samples that would likely be misclassified, and either refrain from classifying them entirely or redirect them to a specialized unit developed for handling such samples, a *reject handler*. When a reject handler is used, it should be a classifier or combination of classifiers that is better suited for the difficult samples. In this case the final recognition accuracy is affected by both the original classification system and the reject handler, whereas without a reject handler the rejected samples are usually discarded. These two approaches to rejection are depicted in Figure 6.1. In the figure, the classification process is illustrated through the abstracted result of the recognition – a rejected result can either be accepted and a reject rate calculated, or redirected to a reject handler and thus the final result of every recognition will be either "Correct" or "Incorrect".

Two fundamental types of rejection can be identified, namely distance and ambiguity rejection [129]. Distance rejection refers to rejecting a sample that belongs to a class outside of the learning set, i.e. the sample is too different from any sample encountered previously. Ambiguity rejection as the naming suggests, is based on ambiguity, i.e. rejecting samples that cannot be clearly attributed to one particular class. The methods of rejection that will be explored here are all concerned with ambiguity rejection, as we assume that each character that is
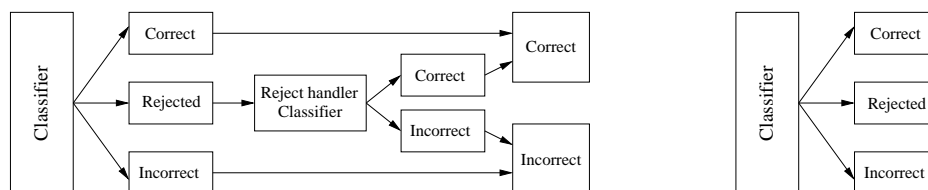
Figure 6.1: A diagram of rejection, with and without a reject handler

input into the system is supposed to belong to one of the existing classes.

One example of committee rejection is to compare the k nearest neighbors from several nearest neighbor classifiers and to perform rejection if too many different classes are suggested by the neighbors [172]. For example the majority voting rule requires the majority to agree on the classification of the sample, or else the sample will be rejected – something that can be seen as an example of inherent committee rejection.

One approach to performing committee classification in general is to base all classifiers on rejection. Thus rejection could be extensively used through iteratively rejecting classes until only one class remains as the final result [19].

6.1 Rejection methods used

In Publication 3 we experimented with a number of committee rejection methods in application with the DEC committee structure. Although the ideas behind the rejection methods can be found in literature, the actual methods were all designed and implemented by the author. Methods based on a rejection threshold for some output provided by the classifier have been suggested e.g. in [61, 189]. Rejection based on voting is just an extension of using majority voting [111, 109] instead of plurality voting. Rejection based on prior knowledge of the problem has been used for example in [191].

The DEC committee was described in detail in Section 5.4. Rejection was implemented in a way that is independent of the committee, so that if the sample is rejected it is not processed by the committee at all. Thus these rejection methods are in no way committee dependent, but applicable to a wide array of settings.

The information used for deciding whether to reject a sample x were the first and

second ranking class labels from each of a total of K member classifiers. We also used the distances to the nearest prototype of the two best result classes, $d_a^i(x)$ and $d_b^i(x)$, respectively, for every member classifier i. The two nearest prototypes were always of different classes. The proposed rejection methods were based on either the class labels or on the distances calculated in the member classifiers. External *a priori* knowledge on difficult classes was incorporated manually for some rejection methods.

Voting Rejection (VR) uses a parameter T_{vote} for determining rejection. If the number of different class labels suggested by the member classifiers is more than T_{vote}, rejection is performed.

Distance-ratio Rejection (DR) examines the averaged ratio of distances from the member classifiers. If the ratio

$$d^{\mathrm{DR}}(x) = \frac{1}{K} \sum_{i=1}^{K} \frac{d_a^i(x)}{d_a^i(x) + d_b^i(x)} \tag{6.1}$$

is greater than a given threshold T_r^{DR}, rejection is performed.

Learning Distance-ratio Rejection (LDR) is identical to the basic DR approach in operation, but the parameter T_r^{LDR} is adjusted using a step value T_{step}^{LDR}. If rejection was performed even though the result would have been correct, the value of T_r^{LDR} is increased as in

$$T_r^{\mathrm{LDR}}(i+1) = T_r^{\mathrm{LDR}}(i) + T_{step}^{\mathrm{LDR}}, \tag{6.2}$$

as unnecessary rejections can be expected to become less frequent with a higher threshold value. If a sample causing an incorrect recognition was not rejected, T_r^{LDR} is similarly decreased by

$$T_r^{\mathrm{LDR}}(t+1) = T_r^{\mathrm{LDR}}(t) - T_{step}^{\mathrm{LDR}}, \tag{6.3}$$

as in such a situation it can be expected that easier rejection could have helped in preventing the error. Thus, LDR is in fact an adaptive rejection method aimed at finding the optimal rejection threshold through adjusting the current threshold value.

Knowledge-based Rejection (KR) performs rejection based on a known set of easily confusable characters, given to the classifier as *a priori* information. In our implementation these confusion sets are defined as the groups $\{o, O, 0\}$, $\{c, C\}$, $\{s, S\}, \{x, X\}$, $\{z, Z\}$ and $\{v, V\}$, where the members of each group are very similar to each other when the size of the inputs is normalized. These groups were selected simply based on their morphological similarity. Three alternative approaches are applied, in the order of increasing total rejection:

KR1: Only the first class labels of the member classifiers are considered. If two different members of any one confusion group are found, rejection is performed.

KR2: Both the first and second class labels from the member classifiers are taken into account. If two different members of any one confusion group are found, rejection is performed.

KR3: If the first class label obtained from any one member classifier belongs to the confusion set, the result is rejected unless all the member classifiers agree on the result.

Learning Knowledge-based Rejection (LKR) keeps track of classifications that have occurred to each character class and the errors that have been made. If the ratio of errors and classifications is greater than the rejection threshold T_r^{LKR}, rejection is performed. The adjustment of T_r^{LKR} is performed as in LDR, but only towards more rejection; when the result was incorrect, the threshold is lowered in accordance with equation (6.3). Thus LKR can also be considered an adaptive rejection method.

Class Rejection (CR) is a simplistic and impractical method, but it was applied for comparison. The method simply disregards a pre-determined set of characters so that if any of the classifiers suggest a class that has been defined for rejection, the sample is rejected. The classes, in order of decreasing difficulty as estimated from the most common error types in earlier experiments, were determined as "$nmurhs0ol9adkbcefgyv1i$". As the rejection is made stricter, progressively more and more of the aforementioned classes are rejected.

6.2 Experiments and results

Detailed information on the rejection experiments can be found in Publication 3. The experiments were conducted with four DTW-based member classifiers that used either the normalized point-to-point or point-to-line distances and either the mass center or bounding box normalization, as described in Section 4.4.1. The data used in the experiments is the same as presented in Section 4.1. Database 1 was used for constructing the member classifier prototype set and Database 3 for testing. Only lowercase letters and digits were used. In hindsight this was not a good choice as for the Knowledge Rejection scheme, only the confusion pair of 'o' and '0' was then actually in effect. The configuration of the DEC committee, discussed in Section 5.4, was to use both the first and second-ranking class labels
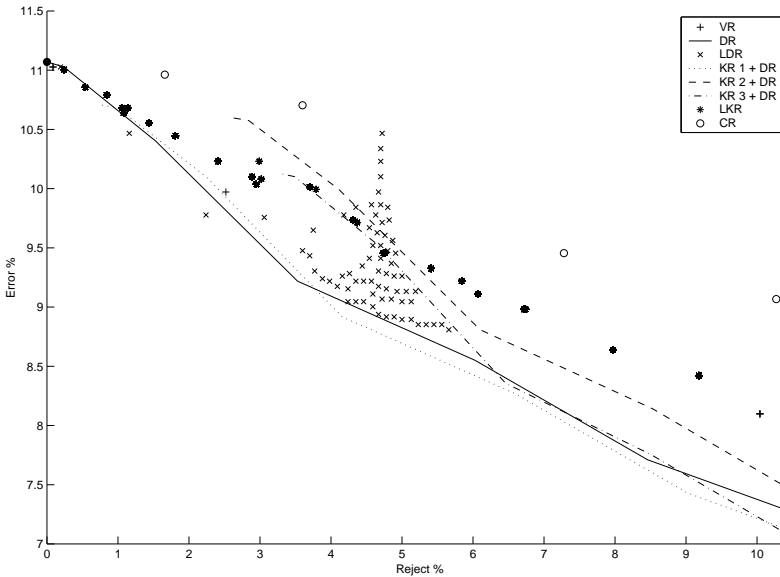
Figure 6.2: Rejection experiment results.

and to use the class label of the highest-ranking classifier as the default decision. The error percentages of the member classifiers were 14.9%, 15.1%, 18.2% and 19.6%, respectively.

The main results are referenced in Figure 6.2 and Table 6.1. The '+'-marks in Figure 6.2 correspond to Voting Rejection, the solid line to Distance-ratio Rejection and the '×'-marks to Learning Distance-ratio Rejection. The dotted line is the first Knowledge-based Rejection method, the dashed the second, and the dash-dot the third, all in combination with the Distance-ratio Rejection scheme. The '∗'-marks represent Learning Knowledge-based Rejection and the circles Class Rejection. While all the other methods were used by themselves, the Knowledge-based Rejection options were applied in conjunction with Distance Rejection. The result of the Knowledge-based Rejection alone can be seen at the beginning of the respective lines.

As is the case with most rejection methods, the error rate can be brought down at the cost of increased rejection. Without having implemented a particular reject handler, it is hard to determine the true value of the rejection. In Table 6.1 we assume that we could correctly classify 50% of the rejected samples. The error percentage after rejection is represented with e_c and the rejection percentage denoted r_r. The column e_{tot} shows the total overall error percentage if we

Table 6.1: Best results in error percentages for each rejection method assuming 50% rejection handler accuracy.

Rejection	e_c	r_r	Parameters	e_{tot}
None	11.07	0.00	—	11.07
VR	11.03	0.09	$T_{vote} = 2$	11.07
DR	9.22	3.53	$T_r^{\text{DR}} = 0.47$	10.98
LDR	9.76	2.26	$T_r^{\text{LDR}} = 0.48, T_{step}^{\text{LDR}} = 0.01$	10.89
KR 1	8.92	4.16	$T_r^{\text{DR}} = 0.47$	11.00
KR 2	8.81	6.10	$T_r^{\text{DR}} = 0.47$	11.86
KR 3	8.36	6.46	$T_r^{\text{DR}} = 0.47$	11.58
LKR	10.86	0.54	$T_r^{\text{LKR}} = 1, T_{step}^{\text{LKR}} = 0.66$	11.13
CR	10.96	1.66	reject 'n','m'	11.79

were to use a reject handler with an accuracy of 50%. We can see that Learning Distance-ratio Rejection (LDR: $T_r^{\text{LDR}} = 0.48$, $T_{step}^{\text{LDR}} = 0.01$), Distance-ratio Rejection (DR: $T_r^{\text{DR}} = 0.47$) and the first mode of Knowledge-based Rejection combined with Distance-ratio Rejection (KR1 + DR: $T_r^{\text{DR}} = 0.47$) produce the best performances. All of them would improve on the original performance if we could correctly classify 50% of the rejected samples. Also the Voting Rejection scheme shows promising behavior, producing no increase in error rate with the same reject handler accuracy. The best performance, assuming 50% reject handler accuracy, is provided with the adaptive LDR rejection scheme.

It is also evident that the plain class-based rejection methods, LKR and CR, are suboptimal for the task. It would seem that there are no specific classes that can be rejected to consistently improve the performance. More details on the experiments and performance of these rejection methods can be found in Publication 3.

However, one should keep in mind that the use of rejection should be a conscious decision. In some applications, it may actually be less user-friendly to reject inputs than to misclassify them. For example with handwritten notes, it is often intuitively easier for a human to decipher the meaning of a note if some errors have been made than if there are characters missing. And also, the need to resubmit characters can be a significant annoyance especially if in a hurry. So for the setting of this thesis, on-line handwriting recognition, it could be argued that if rejection were used, it should be done to a specialized reject handler classifier. And hence, the value of rejection would be directly proportional to the efficiency of the reject handler.

Chapter 7

Classifier selection based on diversity

While research is often focused towards developing more efficient methods for combining classifiers, it should not be forgotten that the used member classifiers of course have a significant effect on the final performance. In fact, efficient committee design can well be considered to consist of two fundamental aspects, *decision optimization* and *coverage optimization* [71]. Of these, the former refers to optimizing the combination method, and the latter the set of classifiers to be used.

The problem in coverage optimization is how to choose the classifiers. If we know beforehand the application and have fixed the combination method, then a valid option is just to use that combination method and some set of applicable data to evaluate the possible combinations of individual classifiers. However, if we have yet to decide on a combination method, or are in the beginning stages of choosing one, it may be beneficial to use a more general method of evaluating how effective combining of some particular set of classifiers could be. Such a method could well be based on the expectation that for the combination to be effective the classifiers should be different in some way, i.e. diverse.

Diversity in itself is a concept that has long been, and continues to be, used in other fields of science such as biology [146, 47] and evolutionary algorithms [202, 230]. For the purpose of combining classifiers, diversity is quite difficult to define. Therefore, it has been considered an important research topic and several

measures have been presented [100, 77, 168, 97, 90, 96, 184, 60, 170, 38, 2, 110, 181, 152]. Of course, diversity measures can also be used for generating artificial outputs of desired diversity to examine the behavior of committee methods in specific situations [99, 231], but this viewpoint was not explored in the research presented here. We have strictly focused on using diversity for classifier selection.

Optimally, a measure of diversity should indicate a set of member classifiers that are different from each other in a way that is beneficial for combining, regardless of the combination method used. Of course in reality the optimal set of classifiers is also dependent on the combination method. However, there could exist some measures of diversity that can identify member classifier sets that are consistently good over a multitude of quite different combination methods.

7.1   Diversity of errors

The most general approach to diversity is a population-theoretic viewpoint, where diversity is maximized with outputs that are always different. This means that the actual values of the outputs are not important, just how much they differ. However, in a classification task, the values of the outputs are very relevant, as there is generally only one correct class and several incorrect ones. In classifier combining, the objective is to produce a set of classifiers that provides the best overall recognition accuracy – not the set that always disagrees by each classifier suggesting a different class. The latter scheme would in practice mean that all but one classifier are always incorrect.

Clearly it is desirable to maximize the number of correct outputs suggested in the set of member classifier outputs, as combination methods generally cannot predict the correct output from receiving just incorrect ones – at least when that situation is introduced for the first time. And since there usually is only one correct result, the inherent problem of overall diversity is evident. It is desirable that all classifiers agree on the correct class, but disagree whenever making errors. Hence a diversity approach crafted for classifier combining should focus on the diversity of the errors that the classifiers make. Agreeing on the correct result should be rewarded, not penalized, but for incorrect outputs, the opposite should be true. In Publication 7 we present a view on diversity and define some measures based on this idea, referred to as *diversity of errors*.

7.2 Measures of diversity

In Publication 7 a number of diversity measures have been reviewed and a taxon-
omy has been suggested, which will also be used here. The measures have been
grouped into three categories based on their relation to the number of classes
and the type of *oracle knowledge*. First, methods using information on all classi-
fier labels, named *general diversity measures*, are presented. Then, five methods
employing oracle knowledge on the correctness of the output in a binary fashion,
binary oracle measures, are shown. Finally, some novel methods that use ora-
cle knowledge on correctness in conjunction with considering all class labels are
introduced as *measures for diversity of errors*.

An overlapping categorization of diversity measures can be made into pairwise
measures and measures that can be used on a number of classifiers simultaneously.
For pairwise measures the diversity value for a larger group can be obtained as
the average of the pairwise measures. This averaging has the notable effect of
making the selection of the optimal member classifier set size impossible. This
is because after having found the two most diverse classifiers, adding a third to
the set will always decrease the set's average diversity. Thus methods that are
based on averaging over pairs are not used for comparing over varying member
classifier set sizes.

In the following, the notation again stands as having K member classifiers to
classify N samples. The output from the ith classifier in the set of classifiers $\mathcal{K}$
for the jth sample x_j in the set of samples $\mathcal{X}$ is denoted $c^i(x_j)$. This classifi-
cation result is one of C class labels $c_i, i = 1, \ldots, C$, with the true label being
$c^{\text{true}}(x_j), j = 1, \ldots, N$.

7.2.1 General diversity measures

According to the general definition of diversity, information on the correctness of
the output is irrelevant to the diversity of the outputs. Diversity increases when
different outputs are produced and diversity is estimated simply from the outputs
produced by the classifiers. Very similar approaches have been often used also for
more general purposes, for example in [20], where approaches based on variance
and uncertainty are examined for population diversity.

Variance is a natural choice for a diversity value, as it measures how the outputs
vary. The bias-variance decomposition [90, 59, 40] can be seen as the basis for

creating a diversity measure based on variance. We will study here a simple approach derived from decomposing the mean squared error, as presented in [59].

For discrete labels, the difference between the output from the i^{th}classifier c^i and the mean output of all K classifiers, $\overline{c}$, is defined as

$$D(c^i(x_j), \overline{c}(x_j)) = \begin{cases} 1 \text{ if } c^i(x_j) \neq \overline{c}(x_j) \\ 0 \text{ otherwise} \end{cases} . \tag{7.1}$$

Furthermore, the mean is poorly defined for discrete labels, but the mode can be used as a replacement of the mean and thus we select the most frequent label for the sample x_j as $\overline{c}(x_j)$ [59]. This is in fact equivalent to the plurality voting result $c_p(x_j)$ of equation (2.5). Now we can express the variance part of the decomposition as

$$V(\mathcal{K}, \mathcal{X}) = \frac{1}{KN} \sum_{j=1}^{N} \sum_{i=1}^{K} D(c^i(x_j), \overline{c}(x_j))^2. \tag{7.2}$$

Variance is calculated simultaneously over all member classifiers, and larger variance is taken as an indication of greater diversity.

The **mutual information** criterion is another logical choice for a diversity method, as it by definition measures the amount of information shared between the classifiers [77]. The pairwise mutual information between two classifiers c^a and c^b can be calculated as

$$I(c^a, c^b, \mathcal{X}) = \sum_{i=1}^{C} \sum_{j=1}^{C} P(c^a = c_i, c^b = c_j) \log \frac{P(c^a = c_i, c^b = c_j)}{P(c^a = c_i)P(c^b = c_j)} , \tag{7.3}$$

where $P(c^a = c_i, c^b = c_j)$ is the joint probability for classifiers c^a and c^b producing the labels c_i and c_j, respectively, $P(c^a = c_i)$ is the probability of classifier c^a producing the label c_i, etc. The probabilities are estimated from the samples in $\mathcal{X}$. For a set of more than two member classifiers, the mean of the pairwise values is used and minimized.

7.2.2   Binary oracle measures

For binary oracle methods, it is assumed that the correct answer is known, and only the correctness of the classifications is considered for evaluating classifier diversity. For notation let us have for classifiers a and b the correctness of classification gathered into N-component binary vectors $\mathbf{v}^a$ and $\mathbf{v}^b$. Thus, for each

recognition, the corresponding value in the vector is one if the classification was correct and zero otherwise. Furthermore, as N is the total number of samples in $\mathcal{X}$, for a two-classifier case the measure $N^{11}(c^a, c^b, \mathcal{X})$ is used to denote the number of times both classifiers are correct, $N^{00}(c^a, c^b, \mathcal{X})$ the number of times both classifiers c^a and c^b are incorrect, and $N^{10}(c^a, c^b, \mathcal{X})$ and $N^{01}(c^a, c^b, \mathcal{X})$ the number of times when just the first or second classifier is correct, respectively. Naturally $N = N^{11}(c^a, c^b, \mathcal{X}) + N^{10}(c^a, c^b, \mathcal{X}) + N^{01}(c^a, c^b, \mathcal{X}) + N^{00}(c^a, c^b, \mathcal{X})$.

Correlation between errors is clearly a valid diversity measure if we assume that negative correlation between errors is beneficial. Here the correlation coefficient for the correctness vectors of classifiers a and b is calculated as

$$\rho(c^a, c^b, \mathcal{X}) = \frac{\mathrm{Cov}(\mathbf{v}^a, \mathbf{v}^b)}{\sqrt{\mathrm{Var}(\mathbf{v}^a)\mathrm{Var}(\mathbf{v}^b)}}, \tag{7.4}$$

where $\mathrm{Cov}(\cdot)$ refers to covariance and $\mathrm{Var}(\cdot)$ to variance. The best subset of classifiers is selected by choosing a set with the minimal mean pairwise correlation. While the measure of correlation between errors in this particular form is the contribution of the author, the benefits of negative correlation have been previously noted e.g. in [122].

The Q **statistic** [96] can be used to assess the similarity of two classifiers. It is defined for two classifiers a, b as

$$Q(c^a, c^b, \mathcal{X}) = \frac{N^{11}(c^a, c^b, \mathcal{X})N^{00}(c^a, c^b, \mathcal{X}) - N^{01}(c^a, c^b, \mathcal{X})N^{10}(c^a, c^b, \mathcal{X})}{N^{11}(c^a, c^b, \mathcal{X})N^{00}(c^a, c^b, \mathcal{X}) + N^{01}(c^a, c^b, \mathcal{X})N^{10}(c^a, c^b, \mathcal{X})}. \tag{7.5}$$

The best subset of member classifiers is selected by minimizing the value of the mean pairwise Q statistic.

The **disagreement measure** [184] is an intuitive pairwise measure counting the number of times that one of the classifiers was incorrect and the other correct. It can thus be defined for two classifiers a and b as

$$DIS(c^a, c^b, \mathcal{X}) = \frac{N^{10}(c^a, c^b, \mathcal{X}) + N^{01}(c^a, c^b, \mathcal{X})}{N}. \tag{7.6}$$

Again, the mean of the pairwise measures is used, and a larger value reflects greater diversity.

The **double-fault measure** [60] is another measure that is intuitive in the sense that it examines the rate of how often the classifiers were incorrect. The double-fault measure is defined for two classifiers a and b as

$$DF(c^a, c^b, \mathcal{X}) = \frac{N^{00}(c^a, c^b, \mathcal{X})}{N}. \tag{7.7}$$

The mean pairwise value is minimized for selecting the set of classifiers.

The **mutual information of errors** can be calculated with an equation similar to (7.3). It can be considered as an application of the idea of mutual information [77], just focusing on only two classes, correct vs. incorrect. It is defined as

$$IE(c^a, c^b, \mathcal{X}) = \sum_{i=1}^{C} \sum_{j=1}^{C} P(c^a = c^{\text{true}}, c^b = c^{\text{true}}) \log \frac{P(c^a = c^{\text{true}}, c^b = c^{\text{true}})}{P(c^a = c^{\text{true}}) P(c^b = c^{\text{true}})} ,$$

(7.8)

where $P(c^a = c^{\text{true}}, c^b = c^{\text{true}})$ is the joint probability for classifiers c^a and c^b being correct, $P(c^a = c^{\text{true}})$ is the probability of classifier c^a being correct, etc. The probabilities are estimated from the samples in $\mathcal{X}$. The mutual information of errors measure should be minimized to select the optimal subset of classifiers, again using the mean of the pairwise values for a larger set of classifiers.

7.2.3   Measures for diversity of errors

Three new measures for the diversity of errors have been introduced in Publication 7, where the distinct failure diversity measure [151] has been used for comparison. All the measures emphasize that differences within the errors made by the member classifiers truly affect performance. Agreeing on an incorrect classification is generally hazardous for classifier combination, whereas agreeing on the correct result is very beneficial. Disagreements in general have less impact on the committee performance than agreements. Of course the importance of agreeing or disagreeing is overall very much dependent on the data, classifiers and combination methods employed, and the level of information the system works on, but this basic assumption is behind the logic of the diversity measures presented in this section.

For this setting we introduce a notation where $N_{\text{same}}^{00}(c^a, c^b, \mathcal{X})$ stands for the number of times both classifiers were incorrect and suggested the same output, and $N_{\text{different}}^{00}(c^a, c^b, \mathcal{X})$, where both classifiers were incorrect, but suggested different outputs.

Distinct failure diversity was proposed in the application domain of multi-version software [151], but is also applicable for classifier combining. The measure focuses on cases where errors are coincidental but distinct, i.e. resulting in different erroneous outputs at the same time. First let us estimate the probability of

exactly n classifiers producing an incorrect result,

$$t_n(\mathcal{K}, \mathcal{X}) = \frac{\text{number of times } n \text{ classifiers produce the same incorrect result}}{\text{total number of distinct erroneous results produced by classifiers}},$$
(7.9)

where the denominator total number of distinct erroneous results produced by classifiers is the total number of incorrect classes that were suggested by at least one classifier. Now distinct failure diversity can be defined as

$$DFD(\mathcal{K}, \mathcal{X}) = \sum_{n=1}^{N} \frac{(N - n)}{(N - 1)} t_n(\mathcal{K}, \mathcal{X})$$
(7.10)

If the total number of distinct erroneous results produced by the classifiers is zero, $DFD(\mathcal{K}, \mathcal{X})$ is defined to be equal to one. The measure is maximized to obtain the optimal set of classifiers.

The **same-fault measure** is an extension to the idea of the double-fault measure. Starting from the double-fault measure, one can further restrict the simultaneous fault consideration to situations where both classifiers were incorrect and suggested the same classification result. This can be defined for two classifiers a and b as

$$SF(c^a, c^b, \mathcal{X}) = \frac{N_{\text{same}}^{00}(c^a, c^b, \mathcal{X})}{N} .$$
(7.11)

The mean of the pairwise measures is used as the measure value for a larger set. The optimal classifier set is then selected by minimizing the measure.

The **weighted count of errors and correct results** is one way to take information on correct classifications into account. It places more emphasis on the situation where classifiers agree on either the correct or incorrect result. In this approach one counts the occurrences of the different correctness combinations and gives suitable weights to the "both correct", a favorable situation, and "both same incorrect", an unfavorable situation:

$$\begin{aligned} WCEC(c^a, c^b, \mathcal{X}) = {} & N^{11}(c^a, c^b, \mathcal{X}) + \tfrac{1}{2}(N^{01}(c^a, c^b, \mathcal{X}) + N^{10}(c^a, c^b, \mathcal{X})) \\ & - N_{\text{different}}^{00}(c^a, c^b, \mathcal{X}) - 5N_{\text{same}}^{00}(c^a, c^b, \mathcal{X}) \end{aligned} .$$
(7.12)

The weighting is arbitrary, and the presented values have been chosen to penalize for errors – especially the same or identical errors. For multiple classifiers, the mean of the pairwise counts is used. The optimal subset can be selected by maximizing the measure.

The **exponential error count** is a diversity measure taking the concept of the diversity of errors one step further. As it is assumed that the member classifiers

will hinder the classification the most when they agree on the same incorrect result, that situation can be given even more emphasis in the selection criterion. The errors can be counted and weighted in an exponential fashion by the number of classifiers making the error. The count of errors made by a total of i classifiers to the same class is denoted $N_{\text{same}}^{i \times 0}(\mathcal{K}, \mathcal{X})$ and added to the sum after rising to the ith power, or

$$EEC(\mathcal{K}, \mathcal{X}) = \frac{\sum_{i=1}^{K}(N_{\text{same}}^{i \times 0}(\mathcal{K}, \mathcal{X}))^i}{N^{K \times 1}(\mathcal{K}, \mathcal{X}) + 1} \ . \tag{7.13}$$

This measure considers all member classifiers of the set at the same time, and the best combination is selected by minimizing the measure. Here also the correct classifications are taken into account by scaling the exponential sum with $N^{K \times 1}(\mathcal{K}, \mathcal{X})$, the number of samples for which every member classifier was correct. The addition of the constant value one safeguards against situations where the classifiers are never simultaneously correct.

It should be noted that one input sample can contribute to more than one $N_{\text{same}}^{i \times 0}(\mathcal{K}, \mathcal{X})$ value – if i classifiers agree on one erroneous output and j classifiers on another, both $N_{\text{same}}^{i \times 0}(\mathcal{K}, \mathcal{X})$ and $N_{\text{same}}^{j \times 0}(\mathcal{K}, \mathcal{X})$ are increased.

7.3   Experiments

In Publication 7 three different committee methods were used for evaluating the diversity methods. The first committee, plurality voting, has been discussed in Section 2.3.3. The second one, the Behavior-Knowledge Space (BKS) committee, was also described in Section 2.3.3. Finally, also an adaptive committee presented in Section 5.4, the Dynamically Expanding Context (DEC) committee classifier, was applied. The configuration of the DEC committee was such that both the first and second outputs from all the classifiers were used for the context. Additionally, it was required that the output is included in the inputs.

The data used in the experiments was the same as presented in Section 4.1. Database 1 was used for member classifier construction and training. Database 2 was used for training the BKS, and Database 3 for testing. All databases featured 68 character classes; digits and both lower and uppercase letters, including three Scandinavian characters ä, ö and å.

Table 7.1: Member classifiers and their error percentages

Classifier index	Member classifier	Errors (ranking)	Errors (test)	Classifier rank
1	DTW PL Bounding box	30.96	23.06	4
2	DTW PL Mass center	25.32	20.02	2
3	DTW PP Bounding box	27.45	21.16	3
4	DTW PP Mass center	23.31	19.30	1
5	Point-sequence SVM	46.21	23.93	7
6	Grid SVM	39.15	26.49	5
7	Point-sequence NN	55.75	50.22	8
8	Grid NN	45.54	35.74	6

The member classifier construction can be found in detail in Publication 7. Four member classifiers were based on the DTW-based stroke matching classifier discussed in Section 4.4.1, using either the point-to-point (PP) or point-to-line (PL) distance and either mass center or bounding box center normalization.

Two member classifiers based on Support Vector Machines (SVMs) [27] were constructed. The SVM classifiers were implemented using the libsvm package version 2.36 [31]. The first of the SVM member classifiers takes its data as a list of points from the character. The second SVM member classifier takes a feature vector of values calculated from a spatial 3×3 grid representation of the character. Also two member classifiers based on Neural Networks (NNs) were included in the experiments. A fully connected feed-forward network structure [67] was created using the Stuttgart Neural Network Simulator (SNNS) version 4.2 [228]. The first NN classifier used the same preprocessing and feature vector type as the first SVM classifier and the second used the same grid-based approach as the second SVM classifier, but with a larger 5×5 grid. Details on the SVM and NN based member classifiers can be found in Publication 7. The member classifiers and their performances have been collected into Table 7.1.

7.4   Experiment results

The most important results from Publication 7 are shown also here. The experiment aimed to determine the best selection of $k = 4$ member classifiers, as the diversity result of several metrics is dependent on the number of member classifiers. Thus the methods could not be directly fairly compared when freely adjusting the number of member classifiers. Additional experiments with

Table 7.2: Comparison of the diversity measures, the optimal classifier selection for each measure and the error percentage obtained with each respective committee combiner

Criterion	Members	Vote	BKS	DEC
Variance	1,6,7,8	20.39	20.68	16.07
Mutual information	1,6,7,8	20.39	20.68	16.07
Correlation of errors	4,6,7,8	18.64	18.23	15.35
Q statistic	4,6,7,8	18.64	18.23	15.35
Disagreement measure	4,6,7,8	18.64	18.23	15.35
Double-fault measure	3,4,5,6	18.21	19.16	15.06
Mutual information of errors	3,5,7,8	18.60	21.00	16.35
Distinct Failure Diversity	4,6,7,8	18.64	18.23	15.35
Same-fault measure	4,6,7,8	18.64	18.23	15.35
Weighted count of errors and corr.	1,4,5,6	17.92	20.31	15.21
Exponential error count	4,5,6,8	17.67	18.13	14.54
Best individual rates	1,2,3,4	19.34	20.07	18.17
All 8 members	1–8	17.89	22.36	16.17

both varying member classifier size and the effect of leaving the worst-performing member classifier out have been presented and discussed in Publication 7.

The best combinations of the eight available member classifiers produced by the different selection criteria, and the resulting accuracies from the three committee structures used for evaluation, have been collected into Table 7.2. Also the recognition rate obtained by using the four individually best member classifiers has been included for comparison as 'Best individual rates', as well as the performance with all eight member classifiers. Additionally, for comparing the results with the best accuracies obtainable with the fixed classifier set size of four member classifiers, the *brute force* approach of evaluating all the 70 possible combinations with each of the three combination methods was also applied. These results are shown in Table 7.3.

The most prominent difference between the selections of the general diversity measures and those that emphasize also correctness is that almost all the general methods use the poorest-performing classifier number 7. The one exception to this is the double-fault measure. The methods that emphasize also correctness, i.e. the weighted count of errors and correct results and the exponential error count, did not select classifier 7. In a general view of diversity it is logical

Table 7.3: Three best brute force error percentages for each committee

	Vote		BKS		DEC	
	Members	**Errors**	**Members**	**Errors**	**Members**	**Errors**
Best	4,5,6,7	17.14	2,3,4,7	17.46	2,5,6,8	14.05
2nd	2,5,6,7	17.31	2,3,6,8	17.83	4,5,6,8	14.54
3rd	3,5,6,7	17.57	3,4,6,8	17.85	2,3,6,8	14.70

to use classifier 7 as it does produce the most different results due to its low accuracy. The general measures, variance and mutual information, show consistently the poorest performance, because their selection encompasses mostly poor-performing classifiers – as is to be expected.

The correlation of errors, Q statistic, and the disagreement measure selected exactly the same set of member classifiers, and produced notably better committee results than the most general methods. The double-fault measure avoids making errors on the same sample, which seems to be a reasonable base strategy. As for the error diversity approaches, neither the distinct failure diversity nor the very simple same-fault measure provide any real advantage. They choose the same set as the majority of the binary-oracle methods.

The weighted count of errors and correct results criterion provides a combination that is the second best for both the voting and DEC committees, but notably poor for the BKS committee. The exponential error count approach finds the best-performing selection of all the criteria and for all combination methods in Table 7.2. This is in accordance with the initial assumption about the importance of the classifiers not making exactly the same mistakes too often, combined with the emphasis on correct outputs.

However, as can be seen from Table 7.3, none of the diversity measures were capable of discovering the truly best set of member classifiers for any of the combination methods. The exponential error count measure's selection {4,5,6,8}, the true second-best set for the DEC committee, is the only one in the top three for any of the combination methods. As is evident from the varying content of the best *brute force* selections, the truly best member classifier set is highly dependent on the combination method.

An interesting difference of behavior can be noted with the BKS committee in comparison to the two other combination methods. The best member classifier set from Table 7.3 for the BKS committee is $\{2, 3, 4, 7\}$, three members with the

best individual accuracies and one with the worst. This could be at least partly due to the fact that the separate training phase teaches the committee the types of errors that commonly occur. Hence the overall diversity of the set becomes a less important factor.

The behavior of the DEC committee is in line with the nature of the rule creation process: it is highly beneficial that the first classifier is as accurate as possible for the lowest-level rules to be maximally general. Correspondingly, one of the two best classifiers is consistently chosen for the first member. The increasing context size of the rules is most beneficial when there is variability in the member classifiers results, but consistently incorrect results are of no benefit.

In the experiments of Publication 7 we also found that the resulting accuracies of the combinations selected by many diversity measures increased when we left the one worst-performing member classifier out of the pool of classifiers. The methods that had chosen the worst-performing classifier as the most "different" classifier could no longer select it as it was not available. This in practice resulted in selecting combinations that provided better overall recognition accuracy in spite of the decrease in diversity according to the majority of the more general measures. This again supports the line of thinking that, for classifier combination purposes, both the diversity of the errors and the frequency of correct outputs are important.

As the final note, it must not be forgotten that the selection of member classifiers is dependent on the combination method's characteristics. This fact was also concluded in [58] among others and can also be clearly seen in our experiments. A particular set of classifiers, while optimal in some sense, does not guarantee the best results for all combination methods. Naturally also the data set used has a very significant effect. However, a suitable measure may still provide useful generalizational ability. The exponential error count, a novel diversity measure focusing on the diversity of errors, was seen to produce a selection that consistently provided good results, at least for this classifier pool and data set.

Chapter 8

Conclusions

In this thesis a novel framework for on-line adaptive combination of classifiers has been presented in the setting of on-line handwritten character recognition. The classifiers to be combined can be adaptive or non-adaptive themselves. The main contribution on the methodological level is the Class-Confidence Critic Combining (CCCC) technique, but also other adaptive committee structures have been presented and they showcase distinct approaches to implementing committee adaptation. These different approaches include a rule-based system in the Dynamically Expanding Context (DEC) committee and a scheme working with and updating a table of classifier and label-specific values in the Modified Current-Best-Learning (MCBL) committee. These classifier combination schemes were evaluated and compared to several reference committees. With non-adaptive member classifiers it is evident that adaptive committee classifiers can produce notable benefits.

The proposed CCCC technique is a very modular classifier combination scheme that can be applied to a wide variety of settings. The distribution models, decision rules, and other components of the system can be altered or changed to better suit the application. The performance of the CCCC committee has been found to be impressive, as it produced the best results in all experiments.

In addition to the actual problem of how to combine the classifiers, the thesis has addressed the question of how to select the classifiers to combine. The experiments also showed that a diverse set of classifiers can clearly outperform another set that consists of the classifiers that are best in their own right. Therefore, how the classifier set is selected is of utmost importance. Some diversity measures have the tendency of preferring classifiers that have high error rates as

this policy produces the most different inputs to the committee. This is often suboptimal for classifier combining, where the overall objective is to obtain the best recognition accuracy. The concept of diversity of errors has been successfully used to form measures that can be used to compute the diversity between member classifiers. In the performed experiments this strategy has shown good performance with a number of different combination strategies. Measures based on the diversity of errors were found to produce a classifier set that resulted in the best combined accuracy among the diversity measures examined.

Also committee rejection and the process of deducing true labels from the user's actions have been given attention. These both are integral components for the practical implementation of an on-line handwritten character recognition system. In our experiments with committee rejection the most effective scheme used was also adaptive by nature. This rejection method, based on an adaptive threshold, outperformed static methods, even ones making use of task-specific prior information. Also label deduction was shown to be effective, as the labeling provided by the suggested scheme was clearly sufficient for effective adaptation.

All in all, combining classifiers can be justified from several viewpoints. Perhaps most notably because in most recognition tasks one cannot find a single globally best classifier. Therefore it can be beneficial to combine the benefits of several classifiers. The main point is of course that the classifiers should be combined in a way that results in their errors becoming collectively canceled. Such a committee could be obtained in many ways, for example through combination of weaker classifiers which each are successful for some part of the data, or with combination schemes where the committee attempts to learn to correct the mistakes made by its members. Computational resources may have earlier been an issue limiting the use of several classifiers simultaneously, but with the constant increase of computational power, using multiple classifiers has become all the more feasible.

Adaptation to the user is a viable approach for improving performance in any setting where a large amount of variation is possible, but only a subset of it is being presented. Our work in handwritten character recognition is a good example of such an application: everyone writes a particular character in only a few of the practically infinite number of possible ways. Hence combining the approaches of adaptivity and classifier combining is well motivated – but far from simple.

Especially when dealing with member classifiers that are adaptive and thus change their performance over time, the implementation of an effective adaptive committee is challenging. In the experiments it was seen that two very different types of committee classifier could provide additional benefits – very simple com-

bination schemes that do not rely on prior information, such as plurality voting, and methods especially designed to function in a setting where classifier behavior is expected to change. In particular the plurality voting rule was surprisingly effective in this changing environment – in the adaptive member classifier experiments the performance of the plurality voting rule was second only to that of the CCCC committee. This is most probably due to the voting strategy's complete lack of dependence on prior behavior. With adaptive member classifiers, the worst performance was obtained with classifier combination schemes heavily dependent on prior training.

The work of this thesis is clearly indicative of the fact that it is possible to provide significant benefit using a combination scheme, such as the CCCC, in a variety of settings. This has been the case both with very good member classifiers and also with ones that are not as exceptional on their own, as well as with adaptive member classifiers. In summary of the adaptive committee experiments it can be stated that when the best non-adaptive classifier provided an error percentage of slightly over 20%, combining them using the CCCC committee yielded an error percentage of approximately 15%. With adaptive member classifiers, the best individual member produced roughly 10% errors, with a tail error percentage of 6%, and the adaptive combination lowered the overall error percentage to slightly below 8%, with the tail error percentage dropping to roughly 4%. Additionally, the computational cost associated with any of the examined combination schemes is negligible in comparison to the cost of classification in the member classifiers.

The substance of this thesis is in the novel methods presented and in the thorough empirical evaluation on their effectiveness. No theoretical analysis of the methods' effectiveness has been presented, and the author is aware of the limits of using only empirical evidence as a proof of concept. The empirical experiments performed were not actual on-line user experiments with adaptivity, but data collected on-line and later evaluated repeatedly in batch runs. The issue is two-fold – on one hand we now do not see the effects of the adaptation of the user to the system, but without a fixed data set it would have been impossible to compare methods. The value of comparable results with different algorithms was seen as more important at this point, but the value of interactive experiments is certainly recognized. Theoretical analysis of the effectivity of such complex combination methods proved to be a more difficult task than expected, at least without making so many simplifying assumptions that would have seriously compromised the applicability of the analysis to a real-world scenario. Theoretical studies were thus left out of the scope of this thesis and will remain an interesting problem for future work.

The research presented in this thesis has successfully combined the ideas of adaptivity and classifier combining to form effective adaptive classifier combination schemes for the application at hand. The methodology is likewise suited for other application areas where there exists a significant amount of variation, but only a subset of it is being expressed. Such applications include, for example, speech recognition and other human interface tasks. The methodology can of course be applied to a wide variety of tasks, but as adaptive classifier combining is based on the idea of on-line learning from samples, it will inevitably be suboptimal with very unstable or even randomly occurring input patterns.

The main points of this thesis, i.e. recognition of handwriting, on-line adaptation and classifier combination, have all been studied before. The use of all three together, however, is something that has so far gained very little attention. Hopefully this novel combination of approaches will provide a new view on an old problem.

Bibliography

[1] M. Aksela. Handwritten character recognition: A palm-top implementation and adaptive committee experiments. Master's thesis, Helsinki University of Technology, 2000.

[2] M. Aksela. Comparison of classifier selection methods for improving committee performance. In *Proceedings of the Fourth International Workshop on Multiple Classifier Systems*, pages 84–93, Guildford, United Kingdom, June 11–13 2003.

[3] M. Aksela, R. Girdziušas, J. Laaksonen, E. Oja, and J. Kangas. Class-confidence critic combining. In *Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition*, pages 201–206, Niagara-on-the-Lake, Canada, August 6–8 2002.

[4] M. Aksela, R. Girdziušas, J. Laaksonen, E. Oja, and J. Kangas. Methods for adaptive combination of classifiers with application to recognition of handwritten characters. *International Journal of Document Analysis and Recognition*, 6(1):23–41, 2003.

[5] M. Aksela and J. Laaksonen. On adaptive confidences for critic-driven classifier combining. In *Proceedings of International Conference on Advances in Pattern Recognition*, volume 2, pages 71–80, Bath, United Kingdom, August 22–25 2005.

[6] M. Aksela and J. Laaksonen. Using diversity of errors for selecting members of a committee classifier. *Pattern Recognition*, 39(4):608–623, 2006.

[7] M. Aksela and J. Laaksonen. Adaptive combination of adaptive classifiers for handwritten character recognition. *Pattern Recognition Letters*, 28(1):136–143, 2007.

[8] M. Aksela, J. Laaksonen, E. Oja, and J. Kangas. Application of adaptive committee classifiers in on-line character recognition. In *Proceedings of International Conference on Advances in Pattern Recognition*, pages 270–279, Rio de Janeiro, Brazil, March 11–14 2001.

[9] M. Aksela, J. Laaksonen, E. Oja, and J. Kangas. Rejection methods for an adaptive committee classifier. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 982–986, Seattle, Washington, USA, September 10–13 2001.

[10] F. M. Alkoot and J. Kittler. Multiple expert system design by combined feature selection and probability level fusion. In *Proceedings of the International Conference Information Fusion*, volume 2, pages THC5/9 – THC5/16, Paris, France, July 10–13 2000.

[11] F. Andrianasy and M. Milgram. A new learning scheme for the recognition of dynamical handwritten characters. In *IEEE Workshop on Neural Networks for Signal Processing*, pages 371–379, Cambridge, Massachusetts, USA, August 31 – September 2 1995.

[12] S. Annadurai and A. Balasubramaniam. Classification of handwritten alphanumeric characters: a fuzzy neural approach. In *International Conference on High Performance Computing*, pages 36–41, Trivandrum, India, December 19–22 1996.

[13] Anonymous. Global prototype establishment procedure for handwritten character recognition. *IBM Technical Disclosure Bulletin*, 31(9):114–116, February 1989.

[14] H. Arakawa. On-line recognition of handwritten characters – alphanumerics, Hiragana, Katakana, Kanji. *Pattern Recognition*, 16(1):9–16, 1983.

[15] N. Arica and F. T. Yarman-Vural. An overview of character recognition focused on off-line handwriting. *IEEE Transactions on Systems, Man and Cybernetics*, 31(2):583–592, 2000.

[16] A. S. Atukorale and P. N. Suganthan. Combining classifiers based on confidence values. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 37–40, Bangalore, India, September 20–22 1999.

[17] C. Bahlmann and H. Burkhardt. The writer independent online handwriting recognition system frog on hand and cluster generative statistical dynamic time warping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):299–310, 2004.

[18] C. Bahlmann, B. Haasdonk, and H. Burkhardt. On-line handwriting recognition with support vector machines - a kernel approach. In *Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition*, pages 49–54, Niagara-on-the-Lake, Canada, August 6–8 2002.

[19] S. Baker and S. K. Nayar. Pattern rejection. In *Proceedings of the conference on Computer Vision and Pattern Recognition*, pages 544–549, Los Alamitos, California, USA, June 18–20 1996.

[20] M. Bedau, M. Zwick, and A. Bahm. Variance and uncertainty measures of population diversity dynamics. *Advances in Systems Science and Applications Special Issue I*, pages 7–12, 1995.

[21] E. J. Bellegarda, J. R. Bellegarda, D. Nahamoo, and K. S. Nathan. A discrete parameter HMM approach to on-line handwriting recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 2631–2622, Detroit, Michigan, USA, May 9–12 1995.

[22] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[23] D. Bouchaffra and V. Govindaraju. A methodology for mapping scores to probabilities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):923–927, 1999.

[24] A. Brakensiek, A. Kosmala, and D. Willett. Performance evaluation of a new hybrid modeling technique for handwriting recognition using identical on-line and off-line data. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 446–449, Bangalore, India, September 20–22 1999.

[25] A. Brakensiek, A. Kosmala, and G. Rigoll. Comparing normalization and adaptation techniques for on-line handwriting recognition. In *Proceedings of the 16th International Conference on Pattern Recognition*, volume 3, pages 73–76, Quebec, Canada, August 11–15 2002.

[26] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[27] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[28] W. M. Campbell and C. C. Broun. Method for adaptive training of polynomial networks with applications to speaker verification. In *Proceedings of International Joint Conference on Neural Networks*, volume 2, pages 1510–1515, Washington, DC, USA, July 15–19 2001.

[29] S.-H. Cha, C. C. Tappert, and S. N. Srihari. Optimizing binary feature vector similarity measure using genetic algorithm and handwritten character recognition. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 662–665, Edinburgh, Scotland, August 3–6 2003.

[30] K.-F. Chan and D.-Y. Yeung. Elastic structural matching for online handwritten alphanumeric character recognition. In *Proceedings of International Conference on Pattern Recognition*, volume 2, pages 1508–1511, Brisbane, Australia, August 17–20 1998.

[31] C.-C. Chang and C.-J. Lin. Libsvm : a library for support vector machines version 2.36. http://www.csie.ntu.edu.tw/~cjlin/libsvm/, October 2002.

[32] K. Cheung, D. Yeung, and R. Chin. A Bayesian framework for deformable pattern recognition with application to handwritten character recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1382–1388, 1998.

[33] C.-H. Chou, C.-C. Lin, Y.-H. Liu, and F. Chang. A prototype classification method and its use in a hybrid solution for multiclass pattern recognition. *Pattern Recognition*, 39(4):624–634, 2006.

[34] S. D. Connell and N. K. Jain. Writer adaptation of online handwriting models. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 434–437, Bangalore, India, September 20–22 1999.

[35] S. D. Connell and A. K. Jain. Learning prototypes for on-line handwritten digits. In *Proceedings of the 14th International Conference on Pattern Recognition*, pages 182–184, Brisbane, Australia, August 17–20 1998.

[36] S. D. Connell and A. K. Jain. Writer adaptation for online handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):329–346, 2002.

[37] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, IT-13(1):21–27, January 1967.

[38] P. Cunningham and J. Carney. Diversity versus quality in classification ensembles based on feature selection. In *Proceedings of the 11th European Conference on Machine Learning*, volume 1810, pages 109–116, Barcelona, Spain, May 30 – June 2 2000.

[39] J. Czyz, J. Kittler, and L. Vandendorpe. Combining face verification experts. In *Proceedings of the 16th International Conference on Pattern Recognition*, volume 2, pages 28–31, Quebec, Canada, August 11–15 2002.

[40] P. Domingos. A unified bias-variance decomposition and its applications. In *Proceedings of the 17th International Conference on Machine Learning*, pages 231–238, Standford, California, USA, June 29 – July 2 2000.

[41] J.-X. Dong, A. Krzyzak, and C. Y. Suen. Fast SVM training algorithm with decomposition on very large data sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):603–618, 2005.

[42] H. Drucker, C. Cortes, L. D. Jackel, Y. LeCun, and V. Vapnik. Boosting and other ensemble methods. *Neural Computation*, 6(6):1289–1301, September 1994.

[43] H. Drucker, R. Schapire, and P. Simard. Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4):705–719, 1993.

[44] R. P. W. Duin. The combining classifier: to train or not to train? In *Proceedings of the 16th International Conference on Pattern Recognition*, volume 2, pages 765–770, Quebec, Canada, August 11–15 2002.

[45] R. P. W. Duin and D. M. J. Tax. Experiments with classifier combining rules. In *Proceedings of International Workshop on Multiple Classifier Systems*, pages 16–29, Santa Margherita di Pula, Italy, June 21–23 2000.

[46] M. C. Fairhurst and A. F. R. Rahman. Enhancing consensus in multiple expert decision fusion. *IEE Proceedings on Vision, Image and Signal Processing*, 147(1):39–46, 2000.

[47] D. P. Faith. Quantifying biodiversity: a phylogenetic perspective. *Conservation Biology*, 16(1):248–252, 2002.

[48] C. Fang, W. Yizhou, and C. Zaniolo. An adaptive learning approach for noisy data streams. In *Proceedings of International Conference on Data Mining*, pages 351–354, Brighton, United Kingdom, November 1–4 2004.

[49] J. Feng and D.-E. Chen. Handwritten similar chinese characters recognition based on multi-class pair-wise support vector machines. In *International Conference on Machine Learning and Cybernetics*, volume 7, pages 4405–4409, Guangzhou, China, August 18–21 2005.

[50] A. Fern and R. Givan. Online ensemble learning: An empirical study. *Machine Learning*, 53(1-2):71–109, 2003.

[51] A. Filatov, A. Gitis, and I. Kil. Graph-based handwritten digit string recognition. In *Proceedings of International Conference on Document Analysis*

and Recognition, volume 2, pages 845–848, Montreal, Canada, August 11–15 1995.

[52] E. Fix and J. L. Hodges. Discriminatory analysis—nonparametric discrimination: Consistency properties. Technical Report Number 4, Project Number 21-49-004, USAF School of Aviation Medicine, Randolph Field, Texas, 1951.

[53] J. Franke and E. Mandler. A comparison of two approaches for combining the votes of cooperating classifiers. In *Proceedings of the 11th International Conference on Pattern Recognition*, volume 2, pages 611–614, The Hague, The Netherlands, September 1992.

[54] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. In J. W. Shavlik, editor, *Proceedings of the 15th International Conference on Machine Learning*, pages 170–178, Madison, Wisconsin, USA, July 24–27 1998.

[55] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journan of Computer and System Sciences*, 55(1):119–139, 1997.

[56] H.-C. Fu, H. Y. Chang, Y. Y. Xu, and H.-T. Pao. User adaptive handwriting recognition by self-growing probabilistic decision-based neural networks. *IEEE Transactions on Neural Networks*, 11(6):1373–1384, 2000.

[57] H.-C. Fu and Y. Y. Xu. Multi-linguistic handwritten character recognition by Bayesian decision-based neural networks. In *Proceedings of IEEE Workshop on Neural Networks for Signal Processing*, pages 626–635, Amelia Island, Florida, USA, September 24–26 1997.

[58] G. Fumera and F. Roli. Performance analysis and comparison of linear combiners for classifier fusion. In *Proceedings of the Joint IAPR International Workshops on Syntactical and Structural Pattern Recognition and Statistical Pattern Recognition*, pages 424–432, Windsor, Canada, August 6–9 2002.

[59] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

[60] G. Giancinto and F. Roli. Design of effective neural network ensembles for image classification purposes. *Image Vision and Computing Journal*, 19:697–705, 2001.

[61] N. Giusti, F. Masulli, and A. Sperduti. Theoretical and experimental analysis of a two-stage system for classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):893–904, 2002.

[62] D. Goldberg and C. Richardson. Touch-typing with a stylus. In *Proceedings of International Conference on Human Factors in Computing Systems*, pages 80–87, Amsterdam, Netherlands, April 24–29 1993.

[63] V. K. Govindan. Character recognition – a review. *Pattern Recognition*, 23(7):671–683, 1990.

[64] S. Guberman. Off-line and online handwriting recognition-common approach. In *IEE European Workshop on Handwriting Analysis and Recognition*, volume 6, pages 1–2, London, United Kingdom, July 14–15 1998.

[65] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet. Unipen project of on-line data exchange and recognizer benchmark. In *Proceedings of International Conference on Pattern Recognition*, pages 29–33, Jerusalem, Israel, October 9–13 1994.

[66] H. Hao, C.-L. Liu, and H. Sako. Confidence evaluation for combining classifiers. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 755–759, Edinburgh, Scotland, August 3–6 2003.

[67] S. Haykin. *Neural Networks - a Comprehensive Foundation*. Prentice-Hall, 1998.

[68] E. M. Herrick. Letters with alternative basic shapes. *Visible Language*, 13(2):133–142, 1979.

[69] D. Hilbert. Grundzge einer allgemeinen theorie der linaren integralrechnungen. (erste mitteilung). *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, pages 49–91, 1904.

[70] T. K. Ho, J. J. Hull, and S. N. Srihari. Combination of decisions by multiple classifiers. In H. S. Baird, H. Bunke, and K. Y. (Eds.), editors, *Structured Document Image Analysis*, pages 188–202. Springer-Verlag, Heidelberg, 1992.

[71] T. K. Ho. *Hybrid Methods in Pattern Recognition*, chapter Multiple Classifier Combination: Lessons and Next Steps. World Scientific Press, 2002.

[72] T. K. Ho, J. J. Hull, and S. N. Srihari. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):66–75, 1994.

[73] L. Holmström, P. Koistinen, J. Laaksonen, and E. Oja. Neural and statistical classifiers - taxonomy and two case studies. *IEEE Transactions on Neural Networks*, 8:5–17, 1997.

[74] J. Hu, M. K. Brown, and W. Turin. HMM based on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):1039–1045, 1996.

[75] Y. S. Huang and C. Y. Suen. A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(1):90–94, 1995.

[76] N. Iwayama, K. Akiyama, and K. Ishigaki. Hybrid adaptation: Integration of adaptive classification with adaptive context processing. In *Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition*, pages 169–174, Niagara-on-the-Lake, Canada, August 6–8 2002.

[77] H. J. Kang and S. W. Lee. An information-theoretic strategy for constructing multiple classifier systems. In *Proceedings of the 15th International Conference on Pattern Recognition*, volume 2, pages 483–486, Barcelona, Spain, September 3–7 2000.

[78] H.-J. Kang and S.-W. Lee. Combining classifiers based on minimization of a Bayes error rate. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 398–401, Bangalore, India, September 20–22 1999.

[79] K. Karhunen. Zur spektraltheorie stochastischer prozesse. *Annales Academiae Scientiarum Fennicae*, 34:1–7, 1946.

[80] E. Kavallieratou, N. Fakotakis, and G. Kokkinakis. Handwritten character recognition based on structural characteristics. In *Proceedings of the 16th International Conference on Pattern Recognition*, volume 3, pages 11–16, Quebec, Canada, August 11–15 2002.

[81] T. Kawatani. Handwritten Kanji recognition using combined complementary classifiers in a cascade arrangement. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 503–506, Bangalore, India, September 20–22 1999.

[82] D. Keysers, W. Macherey, H. Ney, and J. Dahmen. Adaptation in statistical pattern recognition using tangent vectors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):259–274, 2004.

[83] N. A. Khan and H. A. Hegt. A flexible and robust matching scheme for character recognition to cope with variations in spatial interrelation among structural features. In *International Conference on Systems, Man and Cybernetics*, volume 5, pages 4166–4171, San Diego, California, USA, October 11–14 1998.

[84] N. A. Khan and H. A. Hegt. Recognition of real-life character samples using a structural variation and degradation model. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 225–228, Bangalore, India, September 20–22 1999.

[85] G. Kim and S. Kim. Feature selection using genetic algorithms for handwritten character recognition. In *Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition*, pages 103–112, Amsterdam, Netherlands, September 11–13 2000.

[86] I.-J. Kim and J.-H. Kim. Statistical character structure modeling and its application to handwritten chinese character recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11):1422–1436, 2003.

[87] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[88] J. Kittler and F. M. Alkoot. Sum versus vote fusion in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1):110–115, 2003.

[89] J. Kittler, M. Ballette, J. Czyz, F. Roli, and L. Vandendorpe. Enhancing the performance of personal identity authentication systems by fusion of face verification experts. In *IEEE International Conference on Multimedia and Expo*, volume 2, pages 581–584, Lausanne, Switzerland, August 26–29 2002.

[90] R. Kohavi and D. H. Wolpert. Bias plus variance decomposition for zero-one loss functions. In L. Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 275–283, Bari, Italy, July 3–6 1996.

[91] T. Kohonen. Dynamically expanding context, with applications to the correction of symbol strings in the recognition of continuous speech. In *International Conference on Pattern Recognition*, volume 2, pages 1148–1151, Paris, France, October 27–31 1986.

[92] T. Kohonen. Dynamically expanding context. *Journal of Intelligent Systems*, 1(1):79–95, 1987.

[93] T. Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer-Verlag, 1997. Second Extended Edition.

[94] K. Ku and P. Chiu. Variability model of a handprinted chinese character. *Electronic Letters*, 31(9):711–712, 1995.

[95] T. T. Kuklinski. Components of handprint style variability. In *Proceedings of the 7th International Conference on Pattern Recognition*, pages 924–926, Montreal, Canada, July 30 – August 2 1984.

[96] L. I. Kuncheva, C. J. Whittaker, C. A. Shipp, and R. P. W. Duin. Is independence good for combining classifiers. In *Proceedings of the 15th International Conference on Pattern Recognition*, volume 2, pages 168–171, Barcelona, Spain, September 3–7 2000.

[97] L. I. Kuncheva. That elusive diversity in classifier ensembles. In *Proceedings of First Iberian Conference on Pattern Recognition and Image Analysis*, pages 1126–1138, Mallorca, Spain, June 4–6 2003.

[98] L. I. Kuncheva, J. C. Bezdek, and R. P. W. Duin. Decision templates for multiple classifier fusion: an experimental comparison. *Pattern Recognition*, 34(2):299–314, 2001.

[99] L. I. Kuncheva and R. K. Kountchev. Generating classifier outputs of fixed accuracy and diversity. *Pattern Recognition Letters*, 23(5):593–600, 2002.

[100] L. I. Kuncheva and C. J. Whitaker. Measures of diversity in classifier ensembles. *Machine Learning*, 51:181–207, 2003.

[101] J. M. Kurtzberg and C. C. Tappert. Symbol recognition system by elastic matching. *IBM Technical Disclosure Bulletin*, 24(6):2897–2902, 1981.

[102] J. Laaksonen. Local subspace classifier. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 637–642, Lausanne, Switzerland, October 8–10 1997.

[103] J. Laaksonen. *Subspace Methods in Recognition of Handwritten Digits*. PhD thesis, Helsinki University of Technology, 1997.

[104] J. Laaksonen, M. Aksela, E. Oja, and J. Kangas. Adaptive local subspace classifier in on-line recognition of handwritten characters. In *Proceedings of International Joint Conference on Neural Networks 1999*, Washington, DC, USA, July 10–16 1999.

[105] J. Laaksonen, M. Aksela, E. Oja, and J. Kangas. Dynamically Expanding Context as committee adaptation method in on-line recognition of handwritten latin characters. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 796–799, Bangalore, India, September 20–22 1999.

[106] J. Laaksonen, J. Hurri, E. Oja, and J. Kangas. Comparison of adaptive strategies for on-line character recognition. In *Proceedings of International Conference on Artificial Neural Networks*, pages 245–250, Skövde, Sweden, September 2–4 1998.

[107] J. Laaksonen, J. Hurri, E. Oja, and J. Kangas. Experiments with a self-supervised adaptive classification strategy in on-line recognition of isolated handwritten latin characters. In *Proceedings of Sixth International Workshop on Frontiers in Handwriting Recognition*, pages 475–484, Taejon, Korea, August 12–14 1998.

[108] J. Laaksonen, V. Vuori, E. Oja, and J. Kangas. Adaptation of prototype sets in on-line recognition of isolated handwritten latin characters. In S.-W. Lee, editor, *Advances in Handwriting Recognition*, pages 489–497. World Scientific Publishing, 1999.

[109] L. Lam and S. Suen. Application of majority voting to pattern recognition: an analysis of its behavior and performance. *IEEE Transactions on Systems, Man and Cybernetics*, 27(5):553–568, 1997.

[110] L. Lam. Classifier combinations: Implementations and theoretical issues. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, pages 77–86, Santa Margherita di Pula, Italy, June 21–23 2000.

[111] L. Lam and C. Y. Suen. A theoretical analysis of the application of majority voting to pattern recognition. In *Procedings of International Conference on Pattern Recognition*, volume 2, pages 418–420, Jerusalem, Israel, October 9–13 1994.

[112] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In L. P.G.J., editor, *Neural Netwotks, current applications*, 1992.

[113] C. Leedham. Historical perspectives of handwriting recognition systems. In *IEE Colloquium on Handwriting and Pen-based input*, pages 1/1–1/3, 1994.

[114] V. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):107–710, 1966.

[115] C. Li, H. Ji, and J. Pei. Multilayer fuzzy HMM for online handwriting shape recognition. In *Proceedings of International Conference on Signal Processing*, volume 2, pages 1427–1430, Istanbul, Turkey, December 15–17 2004.

[116] C. P. Lim and R. F. Harrison. Online pattern classification with multiple neural network systems: An experimental study. *IEEE Transactions on Systems, Man and Cybernetics*, 33(2):235–247, 2003.

[117] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, COM-28(1):84–95, January 1980.

[118] C.-L. Liu, S. Jaeger, and M. Nagakawa. Online recognition of chinese characters: the state-of-the-art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):198–213, 2004.

[119] C.-L. Liu and M. Nakagawa. Prototype learning algorithms for nearest neighbor classifier with application to handwritten character recognition. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 378–381, Bangalore, India, September 20–22 1999.

[120] C.-L. Liu, K. Nakashima, H. Sako, and H. Fujisawa. Handwritten digit recognition: benchmarking of state-of-the-art techniques. *Pattern Recognition*, 36(10):2271–2285, 2003.

[121] X. Y. Liu and M. Blumenstein. Experimental analysis of the modified direction feature for cursive character recognition. In *Proceedings of the 9th International Workshop on Frontiers in Handwriting Recognition*, pages 353–358, Tokyo, Japan, October 26–29 2004.

[122] Y. Liu and X. Yao. Ensemble learning via negative correlation. *Neural Networks*, 12(10):1399–1404, 1999.

[123] W. Loy and L. Landay. An on-line procedure for recognition of handprinted alphanumeric characters. *Pattern Recognition*, 4(4):422–427, July 1982.

[124] P.-Y. Lu and R. W. Brodersen. Real-time on-line symbol recognition using a DTW processor. In *Proceedings of International Conference on Pattern Recognition*, volume 2, pages 1281–1283, Montreal, Canada, July 30 – August 2 1984.

[125] I. S. MacKenzie, B. Nonnecke, S. Riddersma, C. McQueen, and M. Meltz. Alphanumeric entry on pen-based computers. *International Journal of Human-Computer Studies*, 41:775–792, 1994.

[126] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[127] U. Markowska-Kaczmar and P. Kubacki. Support vector machines in handwritten digits classification. In *International Conference on Intelligent Systems Design and Applications*, pages 406–411, Wroclaw, Poland, September 8–10 2005.

[128] S. Marukatat, R. Sicard, T. Artieres, and P. Gallinari. A flexible recognition engine for complex on-line handwritten character recognition. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 1048–1052, Edinburgh, Scotland, August 3–6 2003.

[129] L. Mascarilla, C. Frelicot, and E.-H. Zahzah. Combining: an alternative for choosing between reject-first and accept-first classifiers. In *Proceedings of the International Conference of the North American Fuzzy Information Processing Society*, pages 253–257, New York, New York, USA, June 10–12 1999.

[130] N. Matic, I. Guyon, J. Denker, and V. Vapnik. Writer-adaptation for on-line handwritten character recognition. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 187–191, Tsukuba City, Japan, October 20–22 1993.

[131] O. Melnik, Y. Vardi, and C.-H. Zhang. Mixed group ranks: Preference and confidence in classifier combination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):973–981, 2004.

[132] A. Meyer. Pen computing, a technology overview and a vision. *ACM SIGCHI bulletin*, July 1995.

[133] D. J. Miller and L. Yan. Critic-driven ensemble classification. *IEEE Transactions on Signal Processing*, 47(10):2833–2844, 1999.

[134] D. J. Miller and L. Yan. Ensemble classification by critic-driven combining. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 1029–1032, Phoenix, Arizona, USA, March 15–19 1999.

[135] D. J. Miller and L. Yan. Some analytical results on critic-driven ensemble classification. In *Proceedings of IEEE Workshop on Neural Networks for Signal Processing*, pages 252–263, Madison, Wisconsin, USA, August 23–25 1999.

[136] M. Mitchell. Book review: Handbook of genetic algorithms (L. D. Davis). *Artificial Intelligence*, 100(1-2):325–330, 1998.

[137] Motorola. Motorola web site. http://www.motorola.com, April 2006.

[138] H. Mouchere, E. Anquetil, and N. Ragot. On-line writer adaptation for handwriting recognition using fuzzy inference systems. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 2, pages 1075–1079, Seoul, Korea, August 29 – September 1 2005.

[139] N. Mozayyani, A. Baig, and G. Vaucher. A fully-neural solution for on-line handwritten character recognition. In *Proceedings of International Joint Conference on Neural Networks*, volume 2, pages 160–164, Anchorage, Alaska, USA, May 4–9 1998.

[140] M. Nakai, N. Akira, H. Shimodaira, and S. Sagayama. Substroke approach to HMM-based on-line Kanji handwriting recognition. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 491–495, Seattle, Washington, USA, September 10–13 2001.

[141] A. Nakamura. A method to accelerate writer adaptation for on-line handwriting recognition of a large character set. In *Proceedings of the 9th International Workshop on Frontiers in Handwriting Recognition*, pages 426–431, Tokyo, Japan, October 26–29 2004.

[142] K. Nathan, J. R. Bellegarda, D. Nahamoo, and E. J. Bellegarda. On-line handwriting recognition using continuous parameter hidden Markov models. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume 5, pages 121–124, Minneapolis, Minnesota, USA, April 27–30 1993.

[143] Nokia. Nokia web site. http://www.nokia.com, April 2006.

[144] R. Nopsuwanchai and A. Biem. Discriminative training of tied mixture density HMMs for online handwritten digit recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 6–10, Hong Kong, April 6–10 (Conference not held due to SARS) 2003.

[145] F. Nouboud and R. Plamondon. A structural approach to on-line charac-
ter recognition: System design and applications. *International Journal of
Pattern Recognition and Artificial Intelligence*, 5(1&2):311–335, 1991.

[146] U. Nübel, F. Garcia-Pichel, M. Kühl, and G. Muyzer. Quantifying micro-
bial diversity: Morphotypes, 16s rRNA genes, and carotenoids of oxygenic
phototrophs in microbial mats. *Applied and Environmental Microbiology*,
65(2):422–430, 1999.

[147] E. Oja. *Subspace Methods of Pattern Recognition*. Research Studies Press,
England, and Wiley, USA, 1983.

[148] D. Okumura, S. Uchida, and H. Sakoe. An HMM implementation for on-line
handwriting recognition based on pen-coordinate feature and pen-direction
feature. In *Proceedings of International Conference on Document Anal-
ysis and Recognition*, volume 1, pages 26–30, Seoul, Korea, August 29 –
September 1 2005.

[149] J. Paik, S. b. Cho, K. Lee, and Y. Lee. Multiple recognizers system using
two-stage combination. In *Proceedings of International Conference on Pat-
tern Recognition*, volume 4, pages 581–585, Banff, Canada, August 25–29
1996.

[150] R. Paredes, E. Vidal, and D. Keysers. An evaluation of the WPE algorithm
using tangent distance. In *Proceedings of the 16th International Conference
on Pattern Recognition*, volume 4, pages 48–51, Quebec, Canada, August
11–15 2002.

[151] D. Partridge and W. Krzanowski. Distinct failure diversity in multiversion
software. *University of Exeter Technical Report*, 348, 1997.

[152] E. Pekalska, R. P. W. Duin, and M. Skurichina. A discussion on the classifier
projection space for classifier combining. In *Proceedings of the Third Inter-
national Workshop on Multiple Classifier Systems*, pages 137–148, Cagliari,
Italy, June 24–26 2002.

[153] C. Perez, C. Held, and P. Mollinger. Handwritten digit recognition based
on prototypes created by euclidean distance. In *Proceedings of Interna-
tional Conference on Information, Intelligence and Systems*, pages 320–323,
Rockville, Maryland, USA, October 31 – November 3 1999.

[154] F. Pernkopf and D. Bouchaffra. Genetic-based EM algorithm for learning
gaussian mixture models. *IEEE Transactions on Pattern Analysis and
Machine Intelligence*, 27(8):1344–1348, 2005.

[155] R. Plamondon and S. N. Srihari. On-line and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):63–84, 2000.

[156] R. Plamondon and W. Guerfali. The generation of handwriting with delta-lognormal synergies. *Biological Cybernetics*, 78(2):119–132, 1998.

[157] E. Poisson, C. V. Gaudin, and P. M. Lallican. Multi-modular architecture based on convolutional neural networks for online handwritten character recognition. In *Proceedings of the 9th International Conference on Neural Information Processing*, volume 5, pages 2444–2448, Singapore, November 18–22 2002.

[158] K. Prema and N. S. Reddy. Pattern recognition. *Sadhana*, 27(5):585–594, 2002.

[159] L. Prevost and L. Oudot. Self-supervised adaptation for on-line script text recognition. *Electronic Letters on Computer Vision and Image Analysis*, 5(2):87–97, 2005.

[160] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of IEEE*, 10(2):257–286, 1989.

[161] B. S. Raghavendra, C. K. Narayanan, G. Sita, A. G. Ramakrishnan, and M. Sriganesh. Prototype learning methods for online handwriting recognition. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 1, pages 287–291, Seoul, Korea, August 29 – September 1 2005.

[162] A. F. R. Rahman and M. C. Fairhurst. A comparative study of decision combination strategies for a novel multiple-expert classifier. In *International Conference on Image Processing and Its Applications*, volume 1, pages 131–135, Dublin, Ireland, July 14–17 1997.

[163] A. F. R. Rahman and M. C. Fairhurst. Exploiting second order information to design a novel multiple expert decision combination platform for pattern classification. *Electronics Letters*, 33(6):476–477, 1997.

[164] A. F. R. Rahman and M. C. Fairhurst. Introducing new multiple expert decision combination topologies: a case study using recognition of handwritten characters. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 2, pages 886–891, Ulm, Germany, August 18–20 1997.

[165] A. Rahman and M. Fairhurst. Generalised approach to the recognition of structurally similar handwritten characters using multiple expert classifiers. *IEE Proceedings on Vision, Image and Signal Processing*, 144(1):15–22, 1997.

[166] M. Revow, C. K. Williams, and G. E. Hinton. Using generative models for handwritten digit recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(6):592–606, 1995.

[167] G. Rigoll, A. Kosmala, J. Rottland, and C. Neukirchen. A comparison between continuous and discrete density hidden Markov models for cursive handwriting recognition. In *Proceedings of International Conference on Pattern Recognition*, volume 2, pages 205–209, Banff, Canada, August 25–29 1996.

[168] F. Roli and G. Giacinto. *Hybrid Methods in Pattern Recognition*, chapter Design of Multiple Classifier Systems. World Scientific Press, 2002.

[169] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

[170] D. Ruta and B. Gabrys. New measure of classifier dependancy in multiple classifier systems. In *Proceedings of the Third International Workshop on Multiple Classifier Systems*, pages 127–136, Cagliari, Italy, June 24–26 2002.

[171] L. Råde and B. Westergren. *BETA Mathematics Handbook for Science and Engineering, 2nd edition*. Chartwell-Bratt Ltd, 1990.

[172] M. Sabourin, A. Mitiche, and D. Thomas. Classifier combination for hand-printed digit recognition. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 163–166, Tsukuba City, Japan, October 20–22 1993.

[173] Samsung. Samsung web site. http://www.samsung.com, April 2006.

[174] D. Sankoff and J. B. Kruskal. *Time warps, string edits, and macromolecules: the theory and practice of sequence comparison*. Addison-Wesley, 1983.

[175] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[176] R. E. Schapire. The boosting approach to machine learning: An overview. In *Proceedings of the MSRI Workshop on Nonlinear Estimation and Classification*, pages 149–172, Berkeley, California, USA, March 19–29 2001.

[177] R. E. Schapire and Y. Singer. Improved boosting using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[178] L. Schomaker. From handwriting analysis to pen-computer applications. *Electrics & Communication Engineering Journal*, pages 93–101, June 1998.

[179] A. Senior and K. Nathan. Writer adaptation of a HMM handwriting recognition system. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 1447–1450, Munich, Germany, April 21–24 1997.

[180] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[181] A. Sharkey and N. Sharkey. Combining diverse neural nets. *The Knowledge Engineering Review*, 12(3):231–247, 1997.

[182] N. Sherkat, R. Whitrow, and R. Evans. Wholistic recognition of handwriting using structural features. In *IEE Colloquium on Document Image Processing and Multimedia*, pages 12/1–12/4, London, United Kingdom, March 25 1999.

[183] P. Simard, Y. LeCun, J. Denker, and B. Victorri. Transformation invariance in pattern recognition, tangent distance and tangent propagation. In G. Orr and M. K., editors, *Neural Networks: Tricks of the trade*, 1998.

[184] D. Skalak. The sources of increased accuracy for two proposed boosting algorithms. In *In Proceedings of the Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms*, Portland, Oregon, USA, August 4–5 1996.

[185] D. B. Skalak. *Prototype selection for composite nearest neighbor classifier*. PhD thesis, University of Massachusetts Amherst, May 1997.

[186] S. J. Smith, M. O. Bourgoin, K. Sims, and H. L. Voorhees. Handwritten character classification using nearest neighbor in large database. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):915–919, 1994.

[187] S. Srihari. *Encyclopedia of Artificial Intelligence (Second Edition)*, chapter Character Recognition, pages 138–150. John Wiley, 1992.

[188] J. A. Starzyk and N. Ansari. Feedforward neural network for handwritten character recognition. In *Proceedings of the International Symposium on Circuits and Systems*, volume 6, pages 2884–2887, San Diego, California, USA, May 3–6 1992.

[189] C. D. Stefano, C. Sansone, and M. Vento. To reject or not to reject: that is the question-an answer in case of neural classifiers. *IEEE Transactions on Systems, Man and Cybernetics*, 30(1):84–94, 2000.

[190] J. Sternby and A. Ericsson. Core points - a framework for structural parameterization. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 1, pages 217–221, Seoul, Korea, August 29 – September 1 2005.

[191] N. Strathy and C. Suen. A new system for reading handwritten zip codes. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 1, pages 74–77, Montreal, Canada, August 11–15 1995.

[192] C. Y. Suen, C. Nadal, R. Legault, T. A. Mai, and L. Lam. Computer recognition of unconstrained handwritten numerals. *Proceedings of the IEEE*, 7(80):1162–1180, 1992.

[193] A.-H. Tan and L.-N. Teow. Learning by supervised clustering and matching. In *Proceedings of International Conference on Neural Networks*, volume 1, pages 242–246, Perth, Australia, November 27 – December 1 1995.

[194] H. Tanaka, K. Nakajima, K. Ishigaki, K. Akiyama, and M. Nakagawa. Hybrid pen-input character recognition system based on integration of online-offline recognition. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 209–212, Bangalore, India, September 20–22 1999.

[195] H. Tanaka, N. Iwayama, and K. Akiyama. Online handwriting recognition technology and its applications. *Fujitsu Scientific and Technical Journal*, 40(1):170–178, 2003.

[196] C. C. Tappert. Adaptive on-line handwriting recognition. In *Proceedings of International Conference on Pattern Recognition*, pages 1004–1007, Montreal, Canada, July 30 – August 2 1984.

[197] C. Tappert, C. Y. Suen, and T. Wakahara. The state of the art in on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(8):787–808, 1990.

[198] L.-N. Teow and A.-H. Tan. Adaptive integration of multiple experts. In *Proceedings of International Conference on Neural Networks*, volume 3, pages 1215–1220, Perth, Australia, November 27 – December 1 1995.

[199] K. Torkkola and T. Kohonen. Correction of quasiphoneme strings by the dynamically expanding context. In *International Conference on Pattern Recognition*, volume 1, pages 487–489, Rome, Italy, November 14–17 1988.

[200] S. Uchida and H. Sakoe. Handwritten character recognition using monotonic and continuous two-dimensional warping. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 499–502, Bangalore, India, September 20–22 1999.

[201] S. Uchida and H. Sakoe. Handwritten character recognition using elastic matching. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 163–167, Edinburgh, Scotland, August 3–6 2003.

[202] R. K. Ursem. Diversity-guided evolutionary algorithms. In *Proceedings of Parallel Problem Solving from Nature*, pages 462–471, Granada, Spain, September 7–11 2002.

[203] M. vanErp and L. Schomaker. Variants of the Borda count method for combining ranked classifier hypotheses. In *Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition*, pages 443–452, Amsterdam, Netherlands, September 11–13 2000.

[204] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[205] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[206] S. Veeramachaneni and G. Nagy. Adaptive classifiers for multisource OCR. *International Journal of Document Analysis and Recognition*, 6(3):154–166, 2003.

[207] V. Vuori. Adaptation in on-line recognition of handwriting. Master's thesis, Helsinki University of Technology, 1999.

[208] V. Vuori. *Adaptive methods for on-line recognition of isolated handwritten characters*. PhD thesis, Helsinki University of Technology, 2002.

[209] V. Vuori. Clustering writing styles with a self-organizing map. In *Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition*, pages 345–350, Niagara-on-the-Lake, Canada, August 6–8 2002.

[210] V. Vuori, M. Aksela, J. Laaksonen, E. Oja, and J. Kangas. Adaptive character recognizer for a hand-held device: Implementation and evaluation setup. In *Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition*, pages 13–22, Amsterdam, Netherlands, September 11–13 2000.

[211] V. Vuori, J. Laaksonen, and J. Kangas. Influence of erroneous learning samples on adaptation in on-line handwriting recognition. *Pattern Recognition*, 35(4):915–926, 2002.

[212] V. Vuori, J. Laaksonen, and E. Oja. A comparison of techniques for automatic clustering of handwritten characters. In *Proceedings of the 16th International Conference on Pattern Recognition*, volume 3, pages 168–171, Quebec, Canada, August 11–15 2002.

[213] V. Vuori, J. Laaksonen, E. Oja, and J. Kangas. On-line adaptation in recognition of handwritten alphanumeric characters. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 792–795, Bangalore, India, September 20–22 1999.

[214] V. Vuori, J. Laaksonen, E. Oja, and J. Kangas. Controlling on-line adaptation of a prototype-based classifier for handwritten characters. In *Proceedings of the 15th International Conference on Pattern Recognition*, volume 2, pages 331–334, Barcelona, Spain, September 3–7 2000.

[215] V. Vuori, J. Laaksonen, E. Oja, and J. Kangas. Experiments with adaptation strategies for a prototype-based recognition system of isolated handwritten characters. *International Journal of Document Analysis and Recognition*, 3(2):150–159, 2001.

[216] V. Vuori, J. Laaksonen, E. Oja, and J. Kangas. Speeding up on-line recognition of handwritten characters by pruning the prototype set. In *Proceedings of 6th International Conference on Document Analysis and Recognition*, pages 501–505, Seattle, Washington, USA, September 10–13 2001.

[217] V. Vuori and E. Oja. Analysis of different writing styles with the self-organizing map. In *Proceedings of the 7th International Conference on Neural Information Processing*, volume 2, pages 1243–1247, Taejon, Korea, November 14–18 2000.

[218] P. S.-P. Wang and A. Gupta. An improved structural approach for automated recognition of handprinted character. *International Journal of Pattern Recognition and Artificial Intelligence*, 5(1&2):97–121, 1991.

[219] J. R. Ward and T. Kuklinski. A model for variability effects in handprinting with implications for design of handwriting character recognition systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(3):438–451, May/June 1988.

[220] J. R. Ward. History of pen computing: Annotated bibliography in on-line character recognition and pen computing. http://rwservices.no-ip.info:81/pens/biblio70.html, November 2006.

[221] S. M. Watt and X. Xie. Prototype pruning by feature extraction for hand-written mathematical symbol recognition. In *Proceedings of Maple Conference*, pages 423–437, Waterloo, Canada, July 17–21 2005.

[222] S. M. Watt and X. Xie. Recognition for large sets of handwritten mathematical symbols. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 2, pages 740–744, Seoul, Korea, August 29 – September 1 2005.

[223] D. Windridge and J. Kittler. A morphologically optimal strategy for classifier combination: Multiple expert fusion as a tomographic process. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(3):343–353, 2003.

[224] D. Windridge and J. Kittler. Performance measures of the tomographic classifier fusion methodology. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(6):731–753, 2005.

[225] B. H. Xiao, C. H. Wang, and R. W. Dai. Adaptive combination of classifiers and its application to handwritten chinese character recognition. In *Proceedings of the 15th International Conference on Pattern Recognition*, volume 2, pages 327–330, Barcelona, Spain, September 3–7 2000.

[226] L. Xu, A. Krzyzak, and C. Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 22(3):418–435, 1992.

[227] L. Xu, M. I. Jordan, and G. E. Hinton. An alternative model for mixtures of experts. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 633–640, Cambridge, MA, 1995.

[228] A. Zell, N. Mache, and G. M. e. al. Snns : Stuttgart neural network simulator. http://www-ra.informatik.uni-tuebingen.de/SNNS/, September 2002.

[229] B. Zhang, M. Fu, H. Yan, and M. Jabri. Handwritten digit recognition by adaptive-subspace self-organizing map (ASSOM). *IEEE Transactions on Neural Networks*, 10(4):939–945, 1999.

[230] E. Zitzler. *Evolutionary Methods for Design, Optimisation, and Control*, chapter Evolutionary Algorithms for Multiobjective Optimization, pages 19–26. CIMNE, 2002.

[231] H. Zouari, L. Heutte, and Y. Lecourtier. Controlling the diversity in clas-
      sifier ensembles through a measure of agreement. *Pattern Recognition*,
      38(11):2195–2199, 2005.