ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MANUSCRIPT-BASED THESIS PRESENTED TO
ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
Ph.D.

BY
Paulo Rodrigo CAVALIN

# ADAPTIVE SYSTEMS FOR HIDDEN MARKOV MODEL-BASED PATTERN RECOGNITION SYSTEMS

MONTREAL, DECEMBER 20 2011

**BOARD OF EXAMINERS**

THIS THESIS WAS EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS:

Mr. Robert Sabourin, Thesis director
Département de génie de la production automatisée à l'École de technologie supérieure

Mr. Ching Y. Suen, Thesis co-director
Concordia University

Mr. Pierre Dumouchel, Committee president
Département de génie logiciel et des technologies de l'information à l'École de technologie supérieure

Mr. Marc Parizeau, External examiner
Université Laval

Mr. Christian Desrosiers, Examiner
Département de génie logiciel et des technologies de l'information à l'École de technologie supérieure

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND PUBLIC

ON DECEMBER 14 2011

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

## ACKNOWLEDGEMENTS

First of all, I would like to express my immense gratitude to my supervisors, Robert Sabourin and Ching Y. Suen. Your support over these years is unforgettable. I cannot imagine how I would conduct this work without your help.

I thank also the members of the board of examiners, Pierre Dumouchel, Marc Parizeau, and Christian Desrosiers. Your feedback has been very important to improve the content of the thesis, and to define interesting future work. In addition, I am grateful to Alceu de Souza Britto Jr, who is the person that encouraged me a lot for doing my PhD studies in Canada, and which has always been kind and helpful whenever I needed.

This thesis would not exist without the support of my family. I dedicate this thesis to my beloved wife, Michelle, who has left our warm country to comfort me during these years, and to my daughter, Julia, who is the bright shining light that guides me through this world. I would like also to thank my father, Luiz Alberto, and my mother, Maria Cristina, who have always been there for me, and my siblings, Junior, Ana Paula, and Ana Luísa, who have always encouraged me.

I cannot forget mentioning my friends from LIVIA: Albert, Bassem, Carlos, Clément, Christophe, David, Dominique, Eduardo, Eulanda, Éric, Francis, George, Idrissa, Jean-François, Jonathan Milgram, Jonathan Bouchard, Luana, Luis da Costa, Marcelo, Miguel, Phillipe, and Vincent. I cannot forget either my friends from Brazil, in special Fausto, Islenho, and Luiz Oliveira, with whom I have had several good conversations.

Finally, I would like to thank the financial support provided by both CAPES and the research grants allowed by my supervisors. Their grants enabled me to dedicate to this work at full time. Also, I appreciate a lot the help provided by the UFT, which allowed me to dedicate the necessary time to complete the thesis.

# ADAPTIVE SYSTEMS FOR HIDDEN MARKOV MODEL-BASED PATTERN RECOGNITION SYSTEMS

Paulo Rodrigo CAVALIN

## ABSTRACT

This thesis focuses on the design of adaptive systems (AS) for dealing with complex pattern recognition problems. Pattern recognition systems usually rely on static knowledge to define a configuration to be used during their entire lifespan. However, some systems need to adapt to knowledge that may not have been available in the design phase. For this reason, AS are designed to tailor a baseline pattern recognition system as required, and in an automated fashion, in both the learning and generalization phases. These AS are defined here, using hidden Markov model (HMM)-based classifiers as a case study.

We first evaluate incremental learning algorithms for the estimation of HMM parameters. The main goal is to find incremental learning algorithms that perform as well as the traditional batch learning techniques, but incorporate the advantages of incremental learning for designing complex pattern recognition systems. Experiments on handwritten characters have shown that a proposed variant of the Ensemble Training algorithm, which employs ensembles of HMMs, can lead to very promising results. Furthermore, the use of a validation dataset demonstrates that it is possible to achieve better performances than those of batch learning.

We then propose a new approach for the dynamic selection of ensembles of classifiers. Based on the concept called "multistage organizations", the main objective of which is to define a multi-layer fusion function that adapts to individual recognition problems, we propose dynamic multistage organization (DMO), which defines the best multistage structure for each test sample. By extending Dos Santos et al's approach, we propose two implementations for DMO, namely DSA$^m$ and DSA$^c$. DSA$^m$ considers a set of dynamic selection functions to generalize a DMO structure, and DSA$^c$ uses contextual information, represented by the output profiles computed from the validation dataset. The experimental evaluation, considering both small and large datasets, demonstrates that DSA$^c$ outperforms DSA$^m$ on most problems. This shows that the use of contextual information can result in better performance than other methods. The performance of DSA$^c$ can also be enhanced in incremental learning. However, the most important observation, supported by additional experiments, is that dynamic selection is generally preferred over static approaches when the recognition problem presents a high level of uncertainty.

Finally, we propose the LoGID (Local and Global Incremental Learning for Dynamic Selection) framework, the main goal of which is to adapt hidden Markov model-based pattern recognition systems in both the learning and generalization phases. Given that the baseline system is composed of a pool of base classifiers, adaptation during generalization is conducted by dynamically selecting the best members of this pool to recognize each test sample. Dynamic

selection is performed by the proposed K-nearest output profiles algorithm, while adaptation during learning consists of gradually updating the knowledge embedded in the base classifiers by processing previously unobserved data. This phase employs two types of incremental learning: local and global. Local incremental learning involves updating the pool of base classifiers by adding new members to this set. These new members are created with the Learn++ algorithm. In contrast, global incremental learning consists of updating the set of output profiles used during generalization. The proposed framework has been evaluated on a diversified set of databases. The results indicate that LoGID is promising. In most databases, the recognition rates achieved by the proposed method are higher than those achieved by other state-of-the-art approaches, such as batch learning. Furthermore, the simulated incremental learning setting demonstrates that LoGID can effectively improve the performance of systems created with small training sets as more data are observed over time.

**Keywords:** Pattern Recognition, Adaptive Systems, Ensembles of Classifiers, Incremental Learning, Dynamic Selection, Hidden Markov Models

# UN SYSTÈME ADAPTATIF BASÉ SUR LES HMM POUR LA RECONNAISSANCE DE FORMES

Paulo Rodrigo CAVALIN

## RÉSUMÉ

Cette thèse porte sur l'étude des systèmes adaptatifs pour la reconnaissance de formes. Habituellement les systèmes de reconnaissance reposent sur une connaissance statique du problème à résoudre et cela pour la durée de vie du système. Cependant il y a des circonstances où la connaissance du problème est partielle lors de l'apprentissage initial à l'étape de la conception. Pour cette raison, les systèmes de classification adaptatifs de nouvelle génération permettent au système de base de s'adapter à la fois en apprenant sur les nouvelles données et sont également capables de s'adapter à l'environnement lors de la généralisation. Cette thèse propose une nouvelle définition d'un système de reconnaissance adaptatif où les MMCs (Modèles de Markov Cachés) sont considérés comme étude de cas.

La première partie de la thèse présente une évaluation des principaux algorithmes d'apprentissage incrémental utilisés pour l'estimation des paramètres des MMCs. L'objectif de cette étude est de dégager les stratégies d'apprentissage incrémental dont la performance en généralisation se rapproche de cette obtenue avec un apprentissage hors-ligne (batch). Les résultats obtenus sur le problème de la reconnaissance de chiffres et de lettres manuscrits montrent la supériorité des approches basées sur les ensembles de modèles. De plus, nous avons montré l'importance de conserver dans une mémoire à court terme des exemples utilisés en validation, ce qui permet d'obtenir un niveau de performance qui peut même dépasser celui obtenu en mode batch.

La deuxième partie de cette thèse est consacrée à la formulation d'une nouvelle approche pour la sélection dynamique des ensembles de classifieurs. Inspiré du concept de fusion appelé « organisation multi-niveau » (multistage organizations), nous avons formulé une variante de ce concept appelé DMO (dynamic multistage organization - DMO) qui permet d'adapter la fonction de fusion dynamiquement pour chaque exemple de test à classer. De plus, le concept DMO a été intégré à la méthode DSA proposée par Dos Santos et al pour la sélection dynamique d'ensembles de classifieurs. Ainsi, deux nouvelles variantes, $\text{DSA}^m$ et $\text{DSA}^c$, ont été proposées et évaluées. Dans le premier cas ($\text{DSA}^m$), plusieurs fonctions de sélection permettent une généralisation de la structure DMO. Pour ce qui est de la variante $\text{DSA}^c$, nous utilisons l'information contextuelle (représentée par les profils de décisions des classifieurs de base) acquise par le système et qui est associée à la base de validation conservée dans une mémoire à court terme. L'évaluation des deux approches sur des bases de données de petite et de grande échelle ont montré que la méthode $\text{DSA}^c$ domine $\text{DSA}^m$ sur la plupart des cas étudiés. Ce résultat montre que l'utilisation d'informations contextuelles permet une meilleure performance en généralisation comparées aux méthodes non informées. Une propriété importante de l'approche $\text{DSA}^c$ est qu'elle peut également servir pour apprendre de nouvelles données dans le temps, une propriété très importante pour la conception de systèmes de reconnaissance adap-

tatifs dans les environnements dynamiques caractérisés par un niveau important d'incertitude sur le problème à résoudre.

Finalement, un nouveau framework appelé LoGID (Local and Global Incremental Learning for Dynamic Selection) est proposé pour la conception d'un système de reconnaissance adaptatif basé sur les MMC, et capable de s'adapter dans le temps durant les phases d'apprentissage de généralisation. Le système est composé d'un pool de classifieurs de base et l'adaptation durant la phase de généralisation est effectuée par la sélection dynamique des membres du pool les plus compétents pour classer chaque exemple de test. Le mécanisme de sélection dynamique est basé sur l'algorithme des K plus proches vecteurs de décision, tandis que l'adaptation durant la phase d'apprentissage consiste à la mise à jour et à l'ajout de classifieurs de base dans le système. Durant la phase d'apprentissage, deux stratégies sont proposées pour apprendre incrémentalement sur des nouvelles données: l'apprentissage local et l'apprentissage global. L'apprentissage incrémentale local implique la mise à jour du pool de classifieurs de base en ajoutant des nouveaux membres à cet ensemble. Les nouveaux membres sont générés avec l'algorithme Learn++. L'apprentissage incrémental global consiste à la mise à jour de la base de connaissances composée des vecteurs de décisions qui seront utilisés en généralisation pour la sélection dynamique des membres les plus compétents.

Le système LoGID a été validé sur plusieurs bases de données et les résultats comparés à ceux publiés dans la littérature. En général, la méthode proposée domine les autres méthodes incluant les méthodes d'apprentissage hors-ligne. Enfin, le système LoGID évalué en mode adaptatif montre qu'il est en mesure d'apprendre de nouvelles connaissances dans le temps au moment où les nouvelles données sont disponibles. Cette faculté d'adaptation est très importante également lorsque les données disponibles pour l'apprentissage sont peu nombreuses.

**Mot-clés:** Reconnaissance de formes, Systèmes adaptatifs, Ensemble de classifieurs, Apprentissage incrémental, Sélection dynamique de classifieurs, Modèles Markoviens

## TABLE OF CONTENTS

## LIST OF TABLES

Page

## LIST OF FIGURES

Page

## LIST OF ABBREVIATIONS

| | |
|---|---|
| 1NN | 1-nearest neighbor |
| ADS | Ambiguity-guided dynamic selection |
| AS | Adaptive system |
| BBL | Block by block learning |
| BL | Batch learning |
| CB | Codebook size |
| CS | Classifier selection |
| CSDS | Class-strength dynamic selection |
| DS | Dynamic Selection |
| DSA | Dos Santos et al's approach |
| DSA$^m$ | Proposed approach implementing DMO with multiple dynamic selection functions |
| DSA$^c$ | Proposed approach implementing DMO with contextual information |
| DSC | Dynamic selection of classifiers |
| DSEoC | Dynamic selection of EoCs |
| DMO | Dynamic multistage organizations |
| DT | Decision template |
| DTree | Decision tree |
| ED | Eucledian distance |

| | |
|---|---|
| EoC | Ensemble of classifiers |
| EoHMM | Ensemble of HMMs |
| ET | Ensemble training |
| EN | Ensemble training using ensembles of HMMs |
| EN_stop | EN with a stop criterion |
| E-step | Expectation step |
| GA | Genetic algorithm |
| GSDS | Global-strength dynamic selection |
| HMM | Hidden Markov model |
| IBW | Incremental Baum-Welch |
| IIL | Instance by instance learning |
| IL | Incremental learning |
| IML | Incremental maximum-likelihood |
| KNN | K-nearest neighbor |
| KNOP | K-nearest output profiles |
| KNORA | K-nearest oracles |
| LoGID | Local and global incremental learning for dynamic selection |
| MDL | Minimum description length |
| MDS | Margin-based dynamic selection |
| MLP | Multilayer perceptron |

| | |
|---|---|
| MO | Multistage organization |
| MV | Majority voting |
| M-step | Maximization step |
| NA | Not available |
| NIST | National Institute of Standards and Technology |
| OP | Output profile |
| OTM | Oracle-based template matching |
| PVDS | Pair of votes dynamic selection |
| RSS | Random subspaces |
| SM | Similarity measure |
| SVM | Support vector machine |
| TL | Topology learning |
| TM | Template matching |
| VQ | Vector quantization |

## LIST OF SYMBOLS

| | |
|---|---|
| $A$ | The state probability distribution of an HMM |
| $a_{ij}$ | The HMM probability of being at state $S_i$ at time $\bar{t}-1$, and at state $j$ at time $\bar{t}$ |
| $\alpha_{i,j,k}$ | Equality, or otherwise, of $\tilde{x}_{i,test,k}$ and $\tilde{x}_{j,val,k}$ |
| $B$ | The observation symbol probability distribution of an HMM |
| $b_j(k)$ | The HMM probability of observing symbol $v_k$ at state $j$ |
| $C$ | Chapter 1: the number of classes<br>Chapters 2 and 3, and Appendix I: the pool of classifiers |
| $C_t$ | The pool of classifiers at time $t$ |
| $c_k$ | Chapter 1: a class<br>Chapters 2 and 3, and Appendix I: a classifier |
| $C'_i$ | A subset of classifiers extracted from $C$ |
| $C^*$ | The set the possible EoCs that can be formed with the classifiers in $C$ |
| $C^{*\prime}$ | A pool of ensembles of classifiers extracted from $C^*$ |
| $C'_k$ | An ensemble of classifiers |
| $C^{*\prime\prime}_i$ | A set of EoC dynamically selected to recognize $x_{i,test}$ |
| $C''_{i,j}$ | A dynamically selected EoC |
| $cl_i$ | The level of confidence of the voting provided by the ensemble $C^{*\prime\prime}_i$ for $x_{i,test}$ |
| $D_t$ | A block of training data available at a given time $t$ |
| $d_i$ | Final recognition decision for $x_{i,test}$ |

| | |
|---|---|
| $DIST_t$ | Distribution used by Learn++ at time $t$ |
| $DSel$ | Dynamic selection set |
| $DSel'$ | Output profiles computed from $DSel$ |
| $DSel'_t$ | Output profiles computed from $D_t$ |
| $\delta_{i,j}$ | Level of similarity between $\tilde{x}_{i,test}$ and $\tilde{x}_{i,val}$ |
| $\varepsilon$ | A small constant added to each parameter in $\lambda_{t-1}$ |
| $\eta$ | The learning rate applied on $D_t$ |
| $\eta_{j,i}$ | Pair of votes given $C'_j$ and $x_{i,test}$ |
| $F_{rec}$ | The complexity factor computed for the recognition phase |
| $F_{tr}$ | The complexity factor computed for the training phase |
| $\gamma_{j,i}$ | The value of ambiguity given $C'_j$ and $x_{i,test}$ |
| $\iota_{j,i}$ | The value for global-strength given $C'_j$ and $x_{i,test}$ |
| $K$ | Chapter 1: total of observation sequences processed by ensemble training<br>Chapter 2: the number of most similar validation output profiles computed by DSA$^c$<br>Chapter 3: the number of most similar validation output profiles computed by KNOP |
| $k$ | An iteration step performed by Learn++ |
| $\lambda$ | Chapter 1: an HMM<br>Chapters 2 and 3, and Appendix I: a dynamic selection function |
| $\lambda_t$ | Chapter 1: an HMM trained at time $t$ |
| $\lambda'_{t-1}$ | A mathematical transformation involving $\lambda_{t-1}$ and $\phi_t$ |

| | |
|---|---|
| $\lambda_c$ | An HMM associated with features extracted from the columns of a character image |
| $\lambda_r$ | An HMM associated with features extracted from the rows of a character image |
| $\lambda_i$ | Chapters 2 and 3, and Appendix I: a dynamic selection function |
| $\bar{\lambda}_{t-1}$ | A partial HMM computed from $\phi_t$ only |
| $\Lambda$ | A set of dynamic selection functions |
| $\Lambda_i$ | An HMM-based classifier |
| $L^*_{i,j}$ | The set of likelihoods produced by $\Lambda_j$, for all classes |
| $L$ | The value of likelihood yielded by an HMM |
| $m_i$ | The value of a margin given $C^{*\prime\prime}_i$ and $x_{i,test}$ |
| $mv_{j,i}$ | The class with the highest number of votes, given $C'_j$ and $x_{i,test}$ |
| $\mu_{j,i}$ | The value of a margin, given $C'_j$ and $x_{i,test}$ |
| $M$ | The number of classes |
| $N$ | Chapter 1: the number of states of an HMM<br>Chapters 2 and 3, and Appendix I: total number of base classifiers |
| $N_{max}$ | Maximum size defined for the pool of classifiers |
| $NB$ | number of blocks of data |
| $O$ | Chapter 1: an observation sequence |
| $O_t$ | Chapter 1: one symbol from the set $V$ |
| $O_i$ | Chapters 2 and 3, and Appendix I: the outputs yielded by base classifiers given $x_{i,test}$ |

| | |
|---|---|
| $o_{i,k}$ | The output yielded by classifier $c_k$ given $x_{i,test}$ |
| $O_i''$ | The set of outputs yielded by the ensemble $C_i^{*\prime\prime}$ given $x_{i,test}$ |
| $o_{i,k}$ | The output yielded by the EoC $C_{i,k}''$ given $x_{i,test}$ |
| $Opt$ | Optimization set |
| $\Omega$ | The set of classes |
| $\omega_i$ | A class in the set $\Omega$ |
| $p_j$ | Global performance of $C_j'$ |
| $p_{j,k}$ | Performance of $C_j'$ for class $\omega_k$ |
| $\pi$ | Initial state distribution of an HMM |
| $\pi_i$ | Initial probability of an HMM for state $i$ |
| $\phi_t$ | HMM sufficient statistics computed from the observation of both $D_t$ and $\lambda_{t-1}$ |
| $\Psi_i$ | A temporary set containing the $K$ most similar output profiles |
| $S$ | The set of HMM states |
| $S_i$ | An HMM state |
| $S_{i,j}^*$ | The set of scores produced by $\Lambda_j$ |
| $t$ | A time step in an incremental learning setting |
| $\bar{t}$ | A time step during the processing of an observation sequence by an HMM |
| $T$ | The length of an observation sequence |
| $T_k$ | The maximum number of iterations defined for Learn++ |
| $Test$ | Test set |

| | |
|---|---|
| $Train$ | Training set |
| $TE_k$ | Test set defined by Learn++ at iteration $k$ |
| $TR_k$ | Training set defined by Learn++ at iteration $k$ |
| $TS$ | Total number of training samples |
| $\theta$ | A predefined threshold used by the switch mechanisms of DSA$^m$, DSA$^c$, and KNOP |
| $\Theta_{j,i}$ | The value of class-strength given $C'_j$ and $x_{i,test}$ |
| $U$ | Total number of EoCs in $C^{*\prime\prime}_i$ |
| $V$ | The set of symbols that can be observed by an HMM |
| $v_i$ | A symbol that can be observed by an HMM |
| $v1_i$ | The most voted class, given $x_{i,test}$ |
| $v2_i$ | The second most voted class, given $x_{i,test}$ |
| $Val$ | Validation set |
| $Val'$ | Output profiles from the validation set |
| $\vartheta$ | Threshold used by the filtering mechanism of DSA$^c$ |
| $\vartheta_{min}$ | Threshold used by the filtering mechanism of LoGID |
| $\vartheta_{max}$ | Threshold used by the filtering mechanism of LoGID |
| $v_{k,j,i}$ | Number of votes for class $\omega_k$ |
| $x_i$ | An input feature vector |
| $\tilde{x}_i$ | Output profile of $x_i$ |
| $x_{j,dsel}$ | A sample in the dynamic selection set $DSel$ |

| | |
|---|---|
| $\tilde{x}_{j,dsel}$ | Output profile of $x_{j,dsel}$ |
| $x_{i,test}$ | The $i$-th test sample |
| $\tilde{x}_{i,test}$ | Output profile of $x_{i,test}$ |
| $\tilde{x}_{i,test,k}$ | The output produced by classifiers $c_k$ given $x_{i,test}$ |
| $x_{j,val}$ | The $j$-th validation sample |
| $\tilde{x}_{j,val}$ | Output profile of $x_{j,val}$ |
| $\tilde{x}_{j,val,k}$ | The output produced by classifiers $c_k$ given $x_{j,val}$ |
| $x_{k,train}$ | The $k$-th training sample |
| $W_k$ | The weighting factor for an observation sequence processed by ET |
| $\zeta$ | Chapter 1: A set of HMMs<br>Chapter 2: The fusion function used by DSA$^c$ |

# INTRODUCTION

Over the past few decades, the development of pattern recognition (PR) systems has been a subject of great interest at several research centers. Significant advances can be observed in many application fields, such as speech recognition (O'Shaughnessy, 2008; Lu et al., 2010), handwriting recognition (Cheriet et al., 2009; Su et al., 2009), face recognition (Zhao et al., 2003; Tan et al., 2006; Zhang and Gao, 2009), signature verification (Bertolini et al., 2010; Batista et al., 2011), and intrusion detection (Khreich et al., 2012), etc, for which various systems have been proposed.

PR systems are designed to classify an input pattern in one of the classes of patterns of an underlying problem (Duda et al., 2000). Input patterns generally present different types of variability, putting major pressure on classification schemes to achieve the lowest possible recognition error rates. For sample recognition, these systems generally rely on the knowledge gained from processing a set of training samples and on the way such knowledge is used during operation.

Normally, the design of PR systems relies mainly on batch learning (BL) settings. In other words, training samples are processed off-line, and some hard-decisions are made based on the results. First, a finite training dataset is acquired, assuming that the training data are adequate, representative, and available in sufficient quantity. Then, these data are processed through several iterations, until an "optimum" set of parameters for the system has been found. By considering that set of parameters, the system is put into operation to carry out the recognition of the test samples.

BL settings, though, might be suboptimal for many problems, especially more complex PR problems. Owing to the lack of appropriate budget, staff, or schedule, it may be very difficult to acquire a training set off-line that is adequate, representative, and contains sufficient samples. Furthermore, there is no guarantee that all the necessary samples have been gathered to train the system, even when abundant resources are available to acquire them. It is also possible that unexpected test samples will be presented to the system during the recognition phase. As a

result, there is likely to be be a great deal of uncertainty as to whether or not the system can recognize samples correctly. BL may not provide the most suitable setting in this case, since the system would have to be retrained to accommodate new knowledge according to changing needs arising during operation. This would require all previous data to be stored somewhere to be reprocessed during retraining, a costly process, given that data storage is memory consuming and reprocessing is time consuming. In addition, retraining a system requires the supervision of a pattern recognition expert, which is a costly maintenance activity.

**Problem statement**

Various solutions have been investigated to overcome the limitations of BL settings in dealing with complex pattern recognition problems, prompted mainly by the possibility of new data becoming available during operation and containing very important information with respect to enhancing existing PR systems. Many adaptive methods have been proposed for this purpose, with the expectation that they will deal better with complex recognition cases. The main goal is that they be flexible enough to avoid reliance on hard-decisions made during the design phase, and to adapt to important decisions made based on data that are observed over time.

In the literature, two types of adaptive approaches can be identified. The first comprises approaches designed to adapt classifiers at the learning level. The second involves approaches that adapt classifiers at the generalization level. These methods are usually referred to as incremental learning (IL) approaches (Polikar et al., 2001; Florez-Larrahondo, 2005; Mongillo and Deneve, 2008) and dynamic selection (DS) approaches (Woods et al., 1997; Giacinto and Roli, 2001; Zhu et al., 2004; Soares et al., 2006), respectively.

The fundamental objective of IL algorithms is to accommodate new knowledge in an existing classification scheme. Such knowledge is intrinsically present in data that become available over time, e.g. after the system is put into operation. IL algorithms focus on processing these data and updating existing classifiers without reliance on the data that have already been processed, in order to avoid redundant and costly computations. Various methods have been proposed for this purpose, many of which update the parameters of the classifiers directly

(Mizuno et al., 2000; Florez-Larrahondo, 2005; Mongillo and Deneve, 2008). This method might be negatively affected by problems such as *catastrophic forgetting* (Polikar et al., 2001), since important information may be lost if existing parameters are changed. Recently, it has been demonstrated that ensembles of classifiers (EoCs) may provide a viable solution to this problem (Polikar et al., 2001; Yu-Shu and Yi-Ming, 2009; Ulas et al., 2009; Kapp et al., 2010). In this case, new classifiers are appended to an existing pool of classifiers without the need to change existing parameters.

The methods proposed for DS are aimed at defining the best classification scheme for a given sample "on the fly", e.g. during the operational phase. In this case, a pool of diverse classifiers is usually generated off-line in the design phase. For the generalization, the best members from that pool are selected to form a new classifier scheme for the recognition of each test sample. This approach assumes that by generating a pool of classifiers that is diverse enough, a DS algorithm can select the best classifiers for recognizing each test sample. This approach may deal better with the variability of the test samples, owing to its ability to adapt the pool of classifiers to each test case.

IL and DS approaches are used in distinct frameworks, that is, researchers generally aim to define an adaptive method for either the learning or the generalization phase. Static selection methods are often used to combine EoCs generated with IL settings, and EoCs created with BL are usually considered for the DS methods. We believe that neither option is optimal, and that EoCs generated with IL could be better exploited if they are combined dynamically during the generalization phase, and DS algorithms could define better classification schemes if the pool of classifiers also evolves over time to accommodate new knowledge. For this reason, the main topic of this thesis is the integration of IL with DS algorithms into a single framework, so that the classification scheme can be fully adapted to new data in both the learning and generalization phases. Some researchers have been conducting related investigations (Gangardiwala and Polikar, 2005; Muhlbaier et al., 2009), but to the best of our knowledge, no such integration framework has been formalized and no extensive study of its behavior has ever been published in the literature.

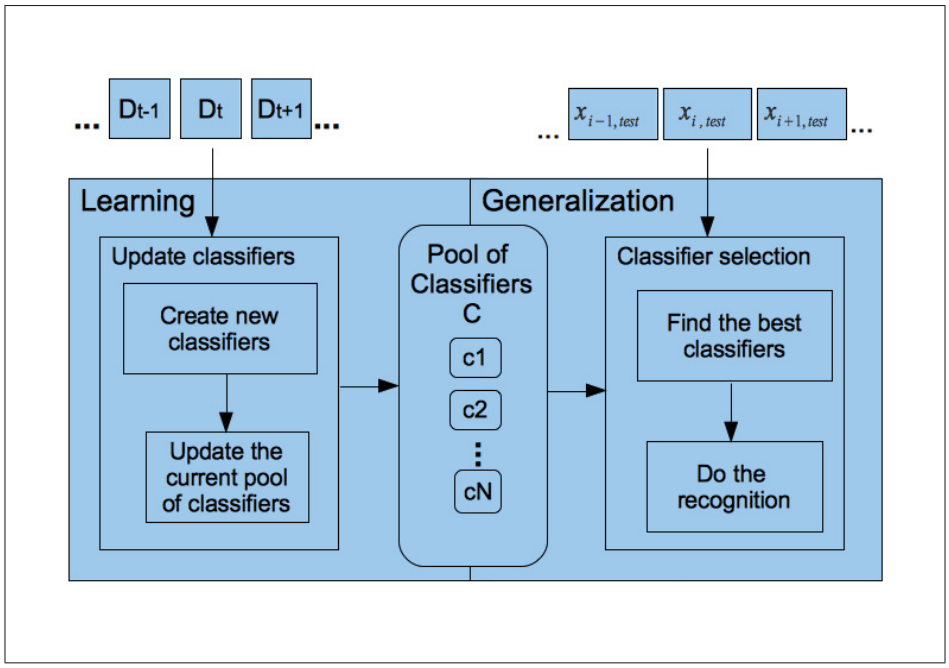



Figure 0.1 General architecture of an adaptive system

The framework mentioned above can be formalized as an adaptive system (AS). A general architecture for this concept is depicted in Figure 0.1. In this case, the baseline recognition system is represented by the pool of classifiers $C$, composed of at least one member. The learning and generalization levels of the AS are designed to adapt the classifiers in $C$ to the new data that are observed over time. At the learning level, $C$ is updated to accommodate new classifiers trained with the block of data $D_t$, presented at a given time $t$. At the generalization level, the main idea is to select the best members of $C$ for recognizing the $i$-th test sample denoted as $x_{i,test}$.

**Objective and contributions**

In this thesis, we focus on the definition of an AS. Specifically, our main goal is to define a framework in which a pool of base classifiers can be adapted at both the learning and generalization levels, and to evaluate its behavior. To achieve this goal, this thesis investigates:

A. how to conduct adaptation at the learning level;

B. how to conduct adaptation at the generalization level;

C. how to integrate these adaptations into a single framework.

We first present investigations related to the use of IL at the learning level, and to the use of dynamic selection algorithms at the generalization level. Our aim is to evaluate which techniques are the most appropriate for use in the adaptive framework. The focus then moves to the integration of the best IL and DS algorithms for defining an AS, the proposal of a framework that implements this type of system, and the evaluation of this framework.

Note that in this thesis we present a case study on hidden Markov models (HMMs). HMM-based classifiers constitute a special family of classifiers that deals with observation sequences. This type of classifier is promising for a very large number of applications, such as handwriting recognition (Gunter and Bunke, 2004), speech recognition (Najkar et al., 2010), and face recognition (Kim et al., 2003).

In terms of our contributions, we present a study on IL algorithms for HMMs for the learning level, taking into account the recognition of alphanumeric characters. This study considers the HMM-based isolated character recognizer proposed in (Britto et al., 2003), for evaluating four different IL algorithms on two isolated character recognition problems, i.e. isolated digits and uppercase letters. This work allowed us to compare single classifier-based IL algorithms against EoC-based algorithms. The latter have been shown to be promising, given that they obtain results that are similar to those achieved by BL methods. Furthermore, we demonstrate that storing samples in a validation set, i.e. a short-term memory, might be beneficial for both improving performance and reducing complexity, since an algorithm for controlling the inclusion of new members into the ensemble can be used.

The first contribution we propose for the generalization level is the concept called dynamic multistage organization (DMO), which is an extension of the multistage organization concept, inspired by dynamic selection of classifiers. In this case, a multistage structure is created for each test sample, defining a fusion function that best exploits the diversity presented by the

pool of base classifiers. We then propose two implementations for DMO: DSA$^m$ and DSA$^c$. The former uses a set of diverse DS functions to select a set of EoCs, which are then combined to provide the final decision. The latter defines a set of EoCs by comparing the output profile of the test sample with the output profiles of the samples stored in the validation set. The most similar samples are used to indicate which EoCs are the best ones for recognizing the test sample. Another important contribution of this work is an evaluation of DS methods against static ones. Under a varied set of conditions, the performance of the DSA$^c$ approach has been compared with those of Multilayer Perceptron (MLP) Neural Networks and Support Vector Machines (SVMs). The results indicate that DS might be preferred over static selection when the level of uncertainty is high, that is, when only small-size training sets are available, and that the performance provided by DS may be worth the higher complexity presented by this type of approach.

Finally, we show how our contributions achieve the main goal of this thesis, which is the definition of an AS represented by the LoGID (**Lo**cal and **G**lobal **I**ncremental Learning for **D**ynamic Selection) framework. This framework considers the K-nearest output profiles (KNOP) algorithm for conducting DS during generalization. This selection is performed by evaluating the output profiles defined with the outputs of the base classifiers. For the learning level, a method inspired by the Learn++ algorithm is used to update the pool of base classifiers. Two pruning algorithms are also proposed to avoid excessive growth of the pool of base classifiers and the set of output profiles respectively. Moreover, in this framework we propose the use of IL at two different levels: local and global. Local IL consists of updating the pool of base classifiers, while global IL consists of adapting the set of output profiles used by the DS algorithm. We demonstrate that these types of IL can be complementary, providing better results when they are combined.

**Organization of this Thesis**

This manuscript-based thesis is organized intro three chapters and one appendix.

In Chapter 1, we present an evaluation of IL algorithms for the estimation of HMM parameters. The main goal is to determine which IL algorithms can perform as well as the BL techniques, but incorporate the advantages of IL in terms of designing complex pattern recognition systems. These algorithms are evaluated based on the recognition of alphanumeric characters. The content of this chapter was published at the 11th International Conference on Frontiers in Handwriting Recognition (Cavalin et al., 2008) and in Pattern Recognition (Cavalin et al., 2009).

In Chapter 2, a new approach for dynamic selection of EoCs is proposed. For this concept, called dynamic multistage organizations (DMO), we propose two implementations that extend Dos Santos et al's approach: DSA$^m$ and DSA$^c$. Experimental evaluations on a varied set of databases have demonstrated the effectiveness of these approaches. In addition, we have demonstrated that the performance of DSA$^c$ can be enhanced in IL settings, and that DS is generally preferred over static approaches when the recognition problem presents a high level of uncertainty, e.g. when only small-size databases are available. The content of this chapter was published at the 9th International Workshop on Multiple Classifier Systems (Cavalin et al., 2010) and in the journal Neural Computing and Applications (Cavalin et al., 2011a). The evaluation of complementary similarity measures that can be used with DSA$^c$ is presented in Appendix I.

Chapter 3 presents the design of an AS represented by the LoGID framework for adapting hidden Markov model-based pattern recognition systems during both the learning and generalization phases. Adaptation during learning is achieved by considering two types of IL: local and global. The former involves updating the pool of base classifiers by adding new members to this set, created with the Learn++ algorithm. The latter consists of updating the set of output profiles used by the proposed KNOP algorithm. Adaptation during generalization is conducted by dynamically selecting the best members of this pool to recognize each test sample, using KNOP. LoGID has been evaluated on a diversified set of databases. The results indicate that the framework is promising, since in most databases the recognition rates achieved by the proposed method are higher than those achieved by other state-of-the-art approaches, including

BL methods. Furthermore, the simulated IL setting demonstrates that LoGID can effectively improve the performance of systems created with small training sets as more data are observed over time. This chapter has been submitted to Pattern Recognition (Cavalin et al., 2011b).

Finally, we present our main conclusions and discuss some possibilities for future work related to this thesis.

# CHAPTER 1

# EVALUATION OF INCREMENTAL LEARNING ALGORITHMS FOR HMM IN THE RECOGNITION OF ALPHANUMERIC CHARACTERS

In this chapter, we present an evaluation of incremental learning (IL) algorithms for the estimation of HMM parameters. Our main goal is to determine which IL algorithms perform as well as the traditional BL techniques, while incorporating the advantages of IL for designing complex pattern recognition systems. Experiments on handwritten characters have shown that a proposed variant of the Ensemble Training algorithm, employing ensembles of HMMs, can lead to very promising performance results. Furthermore, the use of a validation dataset demonstrates that it is possible to achieve better performances than those of BL.

## 1.1 Introduction

The design of complex pattern recognition systems for recognizing handwriting or speech involves the search for classifiers that produce high generalization performances (Duda et al., 2000). The performance of a classifier comes from the accurate estimation of its parameters, which can be generally adjusted by means of a training database and a learning algorithm (Florez-Larrahondo, 2005; Gotoh et al., 1998; Mizuno et al., 2000; Polikar et al., 2001). In spite of being possible to find in the literature different approaches for such an estimate, traditionally a batch learning (BL) setting is used, which is somehow a standard procedure. Moreover, BL is known to be very robust (Oliveira et al., 2002; Dong et al., 2005).

Basically, a BL approach consists of learning the parameters of a classifier from a completely-available training dataset, and the learning algorithm is able to execute as many iterations on the training set as necessary for tuning such parameters. After that process, the parameters never change.

Despite its robustness, BL presents some drawbacks. First, the training database may not be a good representation of the general problem to which the system is related, and the classifiers

will perform poorly in generalization no matter how good the learning algorithm is. This problem could be solved by incorporating new information that is available through the execution of the system to which the classifiers are associated, but there is no known way to do this unless we train a new classifier using both the old and the new data, which is a process that requires lots of time and memory when considering a BL setting.

An incremental learning (IL) setting, however, is promising for overcoming the shortcomings found in BL approaches. IL consists of techniques that have originally been proposed to enable classifiers to gather more information from unseen data, without having to access previously-learned data. Although the term "incremental learning" has been used to refer to different concepts in the literature, IL, in this paper, represents an algorithm with the following characteristics: a) it is able to extract additional information from new data; b) it does not require access to the data used to train the existing classifiers; c) it preserves previously acquired knowledge; and d) it is able to accommodate new classes that may be introduced by new data (Polikar et al., 2001).

Although IL is meant to be as robust as BL in estimating parameters of classifiers, recent research has suggested that generally IL performs worse than BL (Florez-Larrahondo, 2005; Gotoh et al., 1998; Mizuno et al., 2000; Polikar et al., 2001). Since the performance of a learning algorithm, as stated by the no free-lunch theorem (Duda et al., 2000), is strictly dependent on the problem to which it is applied, the main goal of this paper is to provide an evaluation of IL algorithms in the recognition of isolated handwritten characters, by considering a state-of-the-art HMM-based handwriting recognition system (Britto, 2001; Britto et al., 2003; Cavalin et al., 2006). An HMM-based framework has been selected due to the potential of HMMs for the handwriting recognition problem in general, and because the application field of IL algorithms for this kind of system is vast. For instance, IL algorithms can be used to adapt previously-learned HMMs to different shapes of characters. Yet, IL can be an important tool to improve systems that use HMM-based frameworks to perform segmentation and recognition at the same time, such as the first stage of the system presented in (Britto et al., 2003). In that case, a two-step learning process can be performed to acquire knowledge for recognition

aspects first, and afterward for segmentation. But, we focus only on isolated character recognition in order to reduce the scope of this research. Furthermore, the use of a large off-line database, such as the NIST SD19 digits database, allows us to perform various simulations of IL settings.

The remainder of this chapter is organized as follows. In Section 1.2 we present some background theory, such as a brief introduction to HMMs and HMM-based classifiers, and an overview of IL techniques focused on HMMs. In Section 1.3, we describe the methodology employed in this work, which includes the baseline system and the IL algorithms evaluated experimentally. Next, in Section 1.4, we report the experimental evaluation and analyze the corresponding results. Conclusions drawn from this work are described in Section 1.5.

## 1.2 General Theory

In this section we present all the theory and notation needed for a proper understanding of this paper, including an introduction to HMMs, a brief explanation of HMM-based classifiers, and an overview of IL techniques focused on HMMs.

### 1.2.1 Hidden Markov Models

HMMs are a modeling technique derived from Markov Models, which are stochastic processes whose output is a sequence of states corresponding to some physical event. HMMs have the observation as a probabilistic function of the states, i.e. the resulting model is a doubly embedded stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observations (Rabiner, 1989).

Mathematically, an HMM is characterized by the following components:

A. A set of states defined as $S = \{S_1, S_2, \ldots, S_N\}$, where $N$ denotes the number of states of the model, and the state at time $\bar{t}$ is defined as $q_{\bar{t}}$

B. A set of observable symbols defined by $V = \{v_1, v_2, \ldots, v_M\}$, where $M$ denotes the number of distinct observable symbols per state.

C. The state probability distribution $A = \{a_{ij}\}$, where

$$a_{ij} = P[q_{\bar{t}+1} = S_j | q_{\bar{t}} = S_i], \qquad 1 \leq i, j \leq N \tag{1.1}$$

D. The observation symbol probability distribution in state $j$, B $= \{b_j(k)\}$, where

$$b_j(k) = P[\text{observing } v_k \text{ at time } \bar{t} | q_{\bar{t}} = S_j], \qquad 1 \leq j \leq N, \quad 1 \leq k \leq M \tag{1.2}$$

E. The initial state distribution $\pi = \{\pi_i\}$,

$$\pi_i = P[q_1 = S_i], \qquad 1 \leq i \leq N \tag{1.3}$$

For convenience, the compact notation in the next equation represents an HMM.

$$\lambda = (A, B, \pi) \tag{1.4}$$

Given an observation sequence $O = O_1\ O_2\ \ldots\ O_T$, where $O_t$ is one symbol from $V$, and $T$ is the observation sequence length, $\lambda$ is used to process the observation sequence $O$.

### 1.2.2 HMM-based Classifiers

HMMs are able to perform recognition tasks in pattern recognition systems. The most popular approach for such tasks consists of creating a set of HMMs so that each class is represented by an independent HMM. The classification of an unknown observation sequence $O$, into a class $c$, can be carried out by computing which HMM outputs the highest likelihood related to $O$.

In detail, consider a $C$-class problem in which each class is represented by a single HMM $\lambda_i$, where $1 \leq i \leq C$. Suppose that $L(O|\lambda_i)$ is the likelihood of $O$ given $\lambda_i$ (the likelihood can be

easily computed by the Forward-Backward procedure or by the Viterbi algorithm (Rabiner, 1989)). In order to find $c$, the following equation must be used:

$$c = \arg\max L(O|\lambda_i),\ 1 \leq i \leq C \tag{1.5}$$

### 1.2.3 Incremental Learning Algorithms for HMMs

Incremental Learning is a topic with increasing interest in research involving HMMs and pattern recognition systems. In the latter, HMMs are used to compose HMM-based classifiers, in which each class is represented by one or more HMMs. IL of HMMs basically consists of updating the HMM-based classifier when unseen data are available. Unseen data may be represented by either a single observation sequence or a block of observation sequences.

The objective of the application of IL algorithms with HMMs is varied. IL can be used to improve the parameters of an existing classifier by accommodating new chunks of data that are available over time. It can also be used to adapt a classifier to new conditions, where the partial preservation of old information is helpful, but the new knowledge is more important than the old one. Also, it can be a way to deal with limited resources, where a large amount of data cannot be either stored or processed at once.

The main idea of an IL for HMMs is the following. Suppose the learning method is receiving a block of data $D_t$, at a given time $t \geq 1$, given the current HMM $\lambda_{t-1}$. Incremental learning, in this case, consists of computing the parameters of the new HMM $\lambda_t$, where:

$$\lambda_t = \lambda'_{t-1} \tag{1.6}$$

In this case, $\lambda'_{t-1}$ corresponds to a mathematical transformation involving both $\lambda_{t-1}$ and $\phi_t$, where the latter represents the sufficient statistics computed from the observation of $D_t$ and $\lambda_{t-1}$.

The IL algorithms proposed in the literature essentially differ in three aspects: 1) the amount of data accumulated in $D_t$; 2) the weight of the data presented in $D_t$; and 3) how the combination of $\lambda_{t-1}$ and $\phi_t$ is performed. Each aspect is discussed in the remainder of this section.

#### 1.2.3.1 The amount of new data

In considering that $D_t$ can be composed of a single observation sequence, as in (Florez-Larrahondo, 2005; Mizuno et al., 2000), or can be composed of a block of $S_t$ samples, as in (Gotoh et al., 1998), a given IL algorithm may present two different types of behavior.

On one hand, with a smaller number of samples in $D_t$ the learning algorithm performs more updates in $\lambda_{t-1}$, and consequently may converge very quickly. One drawback, however, is that after learning a sufficient number of samples, this algorithm may be biased to its current parameters. In other words, if a new sample is too different from those previously learned, the information presented by the new sample may not be appropriately learned since low-probability values are computed from this sample.

On the other hand, saving more observation sequences in $D_t$ makes the algorithm less sensitive to noise and variations in the data stream. But a block of data needs more memory and time to be stored and processed than a single sample, because this approach is more complex than the first one.

#### 1.2.3.2 The weight of the new data

Another important aspect of IL algorithms is the weight of the unseen data presented in $D_t$. In some cases, this data is used to add more knowledge to a given HMM, assuming that the knowledge in $D_t$ is complementary to the knowledge presented in all $D_{t'}$, where $t' < t$, thus the weight of the new data must be the same as the previous data. In other cases, $D_t$ presents some data that is useful to transform the parameters of a given HMM from more generalized ones to more specialized parameters, and the weight of the new data is greater than the weight of the old data. The latter is generally referred to as adaptation (Digalakis, 1999; Chien et al., 1997).

Generally, the use of a learning rate $\eta$ on $D_t$ (Mizuno et al., 2000; Stenger et al., 2001) allows for explicitly changing the importance of the unseen data. The learning rate defines the behavior of the algorithm in terms of conservatism and adaptation. The higher the value set for $\eta$, the more adaptive the algorithm to new data. Consequently, old data is forgotten very quickly. Lower learning rates define an algorithm that gives as much importance to newer data as it does to older ones, conserving the acquired knowledge as long as possible.

Some algorithms, though, employ an implicit learning rate scheme, where the weight of the new data is defined by the IL algorithm itself. For example, in (Florez-Larrahondo, 2005) all the samples have the same weight since $\phi_t$ stores sufficient statistics of all the samples processed before time $t$. Furthermore, some weight for the unseen data might be computed by taking into account some measure of performance, where samples with a higher performance value will have higher weights.

#### 1.2.3.3 Combining old and new information

The third issue involving IL algorithms for HMMs lies in the way $\lambda_{t-1}$ and $\phi_t$ are combined to generate $\lambda'_{t-1}$. Some solutions have been proposed for this objective, from which we identify two distinct groups.

The first group consists of methods that compute $\lambda'_{t-1}$ directly from $\lambda_{t-1}$. For instance, some methods compute $\lambda'_{t-1}$ by performing a partial expectation step (E-step) of the Baum-Welch algorithm on $\lambda_{t-1}$ and $\phi_t$ (Florez-Larrahondo, 2005; Gotoh et al., 1998; Mizuno et al., 2000; Digalakis, 1999; Neal and Hinton, 1993; Singer and Warmuth, 1997), and a maximization step (M-step) after each time $t$. One shortcoming of this approach is that once one parameter in $\lambda_{t-1}$ is set to zero, there is no way to re-estimate this parameter again. To work around that problem, a small constant $\varepsilon$ is added to each parameter in $\lambda_{t-1}$ (Florez-Larrahondo, 2005), but this solution results in an imprecise estimate of parameters and additional evaluations are required for finding the best value of $\varepsilon$. In (Baldi and Chauvin, 1994), a normalized-exponential representation of HMM parameters was proposed, which avoids zero probabilities. However, the latter never enables zero probabilities, which can also become an issue on the other hand.

The second group consists of methods that estimate a partial HMM $\bar{\lambda}_{t-1}$, from $\phi_t$, without taking into account the knowledge stored in $\lambda_{t-1}$. Then, $\bar{\lambda}_{t-1}$ and $\lambda_{t-1}$ are combined to generate $\lambda'_{t-1}$. Such a combination can be done by merging both $\bar{\lambda}_{t-1}$ and $\lambda_{t-1}$ (Mackay, 1997; Davis and Lovell, 2003), or by creating an Ensemble of HMMs (EoHMMs). Although we cannot find in the literature a method based on the latter, Polikar's Learn++ algorithm is suitable for this objective and can be used with HMM-based classifiers as well (Polikar et al., 2001). This kind of approach avoids the aforementioned zero-probability problem by processing unseen data independently of previous knowledge, which allows for performing batch learning on each block for creating $\bar{\lambda}_{t-1}$ from a randomly-defined initial HMM. This approach, though, may be relatively time-consuming for creating a good estimate for $\bar{\lambda}_{t-1}$.

## 1.3 Methodology

Here we present the entire methodology employed in this work. Section 1.3.1 describes the baseline isolated characters recognizer. In Section 1.3.2, we present the IL algorithms evaluated in this work. And the methodologies proposed for complexity analysis are presented in Section 1.3.3.

### 1.3.1 The Baseline System

The baseline system is the isolated character recognizer presented in (Britto, 2001; Britto et al., 2003; Cavalin et al., 2006). This recognizer is divided into three modules: Pre-processing, Feature Extraction, and Recognition (see Figure 1.1 for an overview of this system).

The Pre-processing module performs corrections of slant inclination in isolated characters.

The Feature Extraction module extracts two observation sequences based on a sliding-window approach. One observation sequence is extracted in the horizontal direction, representing column observations, and the other one is extracted in the vertical direction, representing row observations. Each discrete observation represents a 47-dimensional feature vector, which is mapped by means of Vector Quantization (VQ).

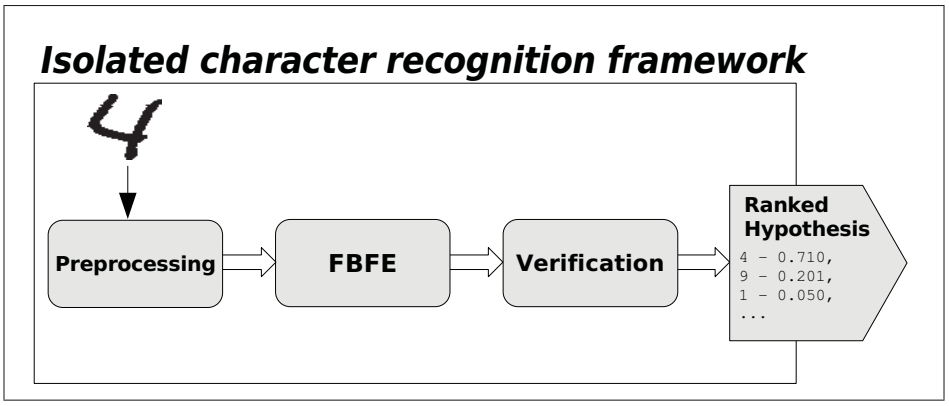



Figure 1.1 An overview of the isolated character recognition framework

The 47-dimensional feature vector combines both foreground and background information, being represented by 34 and 13 features, respectively. From the 34 foreground features, 32 represent local information about the writing, observed from background-foreground transitions. The other 2 features represent a global point of view about the writing in the frame from which they are extracted. The 13 background features are based on a 13-configuration chain code, representing concavity information.

The Recognition module combines both column and row likelihoods to classify the corresponding image as one of the 10 classes of digits, or as one the 26 classes of uppercase letters. Note that each class is represented by two HMMs $\lambda_c$ and $\lambda_r$, being $\lambda_c$ an HMM trained from column observation sequences, and $\lambda_r$ another HMM trained from row observation sequences. In this case, $\log L(O|\lambda)$ is represented by the following equation:

$$\log L(O|\lambda) = \log L(O|\lambda_c) + \log L(O|\lambda_r) \tag{1.7}$$

The system is described in (Britto, 2001; Britto et al., 2003; Cavalin et al., 2006) in greater detail.

Algorithm 1.1 The Incremental Baum-Welch algorithm

1: $t = 0$
2: Initialize sufficient statistics $\phi_t$ to zero.
3: **for** each new observation sequence **do**
4: $t = t + 1$
5: Compute $\phi_t$ from $D_t$ and $\lambda_{t-1}$
6: $\phi_t = \phi_t + \phi_{t-1}$
7: Compute $\lambda'_{t-1}$ by taking into account $\lambda_{t-1}$ and $\phi_t$, using the re-estimation procedure used by the traditional Baum-Welch algorithm.
8: $\lambda_t = \lambda'_{t-1}$
9: **end for**

### 1.3.2 Incremental Learning algorithms

In this section, a brief description of four IL algorithms for HMMs is provided. Advantages and disadvantages of each one are pointed out.

#### 1.3.2.1 The Incremental Baum-Welch algorithm (IBW)

The Incremental Baum-Welch (IBW) algorithm is a straight-forward adaptation of the original BL Baum-Welch algorithm to IL. First proposed in (Stenger et al., 2001) for continuous HMMs, it was later adapted to discrete models in (Florez-Larrahondo et al., 2005).

The IBW algorithm consists of performing a partial E-step using just a single observation sequence, and an M-step for each time step $t$. In other words, the values of $a_{ij}$ and $b_j(k)$, respectively corresponding to the matrices $A$ and $B$ of an HMM, are updated at each time step $t$, given $\lambda_{t-1}$ and $D_t$, where $D_t$ is composed of a single observation sequence. See Algorithm 1.1 for detail.

This algorithm presents some worth noting aspects. In spite of only considering one observation sequence to update $\lambda_{t-1}$, sufficient statistics used in this algorithm represent information computed from all the observed data, meaning that each sample has the same weight. More-

Algorithm 1.2 The Incremental Maximum-Likelihood algorithm

1: Initialize sufficient statistics $\phi_t \forall t$ to zero.
2: **for** $t = 1, 2, \ldots, T$ or until convergence **do**
3: Compute $\phi_t$ from $D_t$ and $\lambda_{t-1}$
4: $\phi_t = \phi_t + \phi_{t-1}$
5: Compute $\lambda'_{t-1}$
6: $\lambda_t = \lambda'_{t-1}$
7: **end for**

over, the addition of a constant $\varepsilon$ in the matrices $A$ and $B$, after doing step 8, was proposed to avoid that some parameters receive a null value.

#### 1.3.2.2 The Incremental Maximum-Likelihood algorithm (IML)

In (Gotoh et al., 1998), the Incremental Maximum-Likelihood (IML) algorithm, which updates an existing HMM by considering a block of data, has been evaluated in an IL setting.

Since the main objective in the work was to speed up the learning process, the proposed IML algorithm works by dividing an off-line training database into smaller blocks. Each iteration of the algorithm processes a different block of data. Thus, given an initial HMM $\lambda_0$, and the blocks of data $D_t, 1 \leq t \leq T$ drawn from the training set, this algorithm works according to Algorithm 1.2.

For an IL setting, IML can be easily adapted, although $T$ is not known a priori and the blocks of data $D_t$ are acquired over time. Such an adaptation results in an algorithm very similar to Algorithm 1.1, where the main difference lies is their *for loop*, which is performed for new blocks of observation sequences instead of individual sequences.

Despite not being explicitly mentioned by the authors, this algorithm also requires the addition of a small constant $\varepsilon$ to the matrices $A$ and $B$ after $\lambda_t$ is computed. Otherwise, as stated before, the information cannot be completely learned by this algorithm if never-seen observations are present within the new observation sequences.

#### 1.3.2.3 Ensemble Training (ET)

Another interesting approach for IL, namely Ensemble Training (ET), was presented in (Mackay, 1997; Davis and Lovell, 2003). Although this algorithm has never been employed within an IL setting (to our knowledge), this algorithm can be easily adapted to this kind of setting since the parameters of the final HMM (i.e. the model used for the recognition) are independently computed for each observation sequence. And despite being originally proposed to deal with single observation sequences, this algorithm can also be easily extended to work with blocks of observation sequences.

ET consists of independently doing the learning of each of the observation sequences from the training set so that each sequence generates an HMM. After all the sequences are learned, the corresponding models are merged to generate a single model representing the whole data. Despite its original name, this method is not characterized as an EoHMMs method because only a single HMM results from the application of this method.

In greater detail, ET works as follows. Suppose that $K$ observation sequences are available for training, and for each of these $K$ observation sequences, one model $\lambda_k = (A_k, B_k, \pi_k)$ is estimated by ET, resulting in the formation of $K$ independent models estimated from the training set. From these $K$ models, the matrices $A$, $B$, and $\pi$, for the final HMM, are computed in the following way:

$$\overline{a}_{ij} = \frac{\sum_k W_k a_{ij}^k}{\sum_k W_k} \tag{1.8}$$

$$\overline{b}_{ij} = \frac{\sum_k W_k b_{ij}^k}{\sum_k W_k} \tag{1.9}$$

$$\overline{\pi}_i = \frac{\sum_k W_k \pi_i^k}{\sum_k W_k} \tag{1.10}$$

where $W_k$ is the weighting factor for each sequence. In this work we empirically defined that $W_k = 1/K$, which indicates that each training sequence has the same weight.

A straight-forward way to adapt ET to work in an IL setting is by conserving a current HMM $\lambda_{t-1}$, which corresponds to all data up to the time step $t-1$. The re-estimation of $A$, $B$, and $\pi$ (the new current model $\lambda_t$), when new data is available, considers only $\lambda_{t-1}$ and the model generated at time $t$ ($\lambda'_t$). One important aspect to assure that the older information is kept in $\lambda_t$ is to consider the weights of the previously-seen data, by accumulating both $W_{t-1}$ and $W'_t$ into $W_t$. Suppose we are updating the model $\lambda_{t-1} = (A_{t-1}, B_{t-1}, \pi_{t-1})$ after observing the data $D_t$, thus we compute $\lambda_t = (A_t, B_t, \pi_t)$, given $\lambda'_t$, by using the following equations:

$$\overline{a}^t_{ij} = \frac{W_{t-1} a^{t-1}_{ij} + W'_t a'^t_{ij}}{W_{t-1} + W'_t} \tag{1.11}$$

$$\overline{b}^t_{ij} = \frac{W_{t-1} b^{t-1}_{ij} + W'_t b'^t_{ij}}{W_{t-1} + W'_t} \tag{1.12}$$

$$\overline{\pi}^t_i = \frac{W_{t-1} \pi^{t-1}_i + W'_t \pi'^t_i}{W_{t-1} + W'_t} \tag{1.13}$$

$$W_t = W_{t-1} + W'_t \tag{1.14}$$

The ET algorithm is very flexible because any learning algorithm can be used to generate the HMM corresponding to the new data, including the original Baum-Welch algorithm.

#### 1.3.2.4 Ensemble Training Using Ensembles of HMMs (EN)

In this section we propose an adaptation of the ET algorithm using EoHMMs, to which we refer as EN for the sake of simplicity. Note that this approach is very similar to the algorithm

presented in (Polikar et al., 2001), namely Learn++, but we consider this adaptation much simpler.

Even though the merging of HMMs used by ET can reliably learn HMMs incrementally, the main advantage of using EoHMMs is that the available set of parameters for acquiring knowledge always increase when new data is available. This is very suitable to incorporate new knowledge. Also, previously-acquired knowledge is never discarded in this case since an HMM's parameters never change once learned.

In this case, instead of having a current HMM $\lambda_{t-1}$, at a learning time $t$, this version of ET has a current set of HMMs $\zeta = (\lambda_1, \lambda_2, \ldots, \lambda_{t-1})$. After computing the partial HMM $\lambda'_t$, this is included in $\zeta$, so that $\zeta = (\lambda_1, \lambda_2, \ldots, \lambda_{t-1}, \lambda'_t)$, which is later converted to $\zeta = (\lambda_1, \lambda_2, \ldots, \lambda_{t-1}, \lambda_t)$.

The likelihood of $\zeta$ can be easily computed by the sum rule, by considering the logarithmic likelihood of each HMM contained in $\zeta$. For example, after learning $t$ blocks of data, the likelihood of $\zeta$, given an observation sequence $O$, can be computed by the following equation:

$$L(O|\zeta) = \sum_{i=1}^{t} logL(O|\lambda_i) \tag{1.15}$$

### 1.3.3 Complexity Analysis

In terms of complexity analysis, two independent factors might be important when considering HMM-based classifiers: a) training complexity, and b) recognition complexity. The methodologies proposed to evaluate each factor are described in the remainder of this section.

#### 1.3.3.1 Evaluation of training complexity

For measuring training complexity, we propose a methodology that is able to compare different learning methodologies employed on the same training data, using the same classifier topology.

Such a methodology is based on the number of samples in each block of data, and the total number of iterations until convergence on each block.

Suppose $NB$ is the number of blocks of data, $S_i$ corresponds to the number of samples in block $i$, where $1 \leq i \leq NB$, and $I_i$ corresponds to the number of iterations to converge on block $i$. The complexity factor of training $F_{tr}$ for learning all data is defined by:

$$F_{tr} = \sum_{i}^{NB} (S_i \times I_i) \tag{1.16}$$

The complexity for estimating the parameters of a single sample is not taken into account by this method. This information tends to be similar, for all algorithms, when the core of the learning algorithms do not differ significantly. That is the case of the algorithms involved in this work, which share the same re-estimation procedure (i.e., they are all based on the forward-backward procedure). Furthermore, the average length of the training observation sequences, and the total number of states of the classifiers, are not taken into account since the same training set and the same classifier topology are used by all the learning algorithms evaluated in this work.

#### 1.3.3.2 Evaluation of recognition complexity

In order to compare the recognition complexity of different HMM-based classifiers, we propose a method based on the total number of states in each classifier. This method can be easily justified if we take into account that the time complexity of the Viterbi algorithm, for decoding an observation sequence $O$ of length $T$ by an HMM with $N$ states, is $O(N^2T)$. Notice that the Viterbi algorithm is a popular algorithm for computing likelihoods of observation sequences given an HMM.

In the case of an HMM-based classifier, composed of $C$ classes, we must compute the total recognition for processing $O$ by all the models that represent the classes. In considering the notation presented in Section 1.2.2, suppose $N_i$ is the number of states of $\lambda_i$, thus the complexity

of the Viterbi algorithm for decoding $O$ for class $i$ is $O({N_i}^2 T)$. To compute the recognition complexity factor $F_{rec}$, of an HMM-based classifier, we can use the following equation:

$$F_{rec} = \sum_{i=1}^{C} {N_i}^2 T \tag{1.17}$$

In generalizing the complexity of the Viterbi algorithm to $O({N_i}^2)$, since $T$ is unknown a priori and irrelevant to compare the recognition complexity of the same test set by different HMM-based classifiers, the equation for computing $F_{rec}$ can be reduced to:

$$F_{rec} = \sum_{i=1}^{C} {N_i}^2 \tag{1.18}$$

Equation 1.18 can be easily adapted to compute the complexity of EoHMMs, which is useful for the algorithm presented in Section 1.3.2.4. In considering that a class is represented by more than one HMM, the complexity of each class can be computed by summing the squared number of states of all HMMs related to the class.

## 1.4 Experimental Evaluation

The experimental evaluation consisted of evaluating the IL algorithms described in Section 1.3.2, for the recognition of handwritten isolated digits and uppercase letters. The isolated character recognition system presented in Section 1.3.1 was the baseline system. The algorithms are compared to a BL setting, using the traditional Baum-Welch algorithm (see (Rabiner, 1989) for details).

Each experiment used the same codebook of 256 symbols, whose codewords were computed from the whole training set. The HMM states were optimized by Wang's method (Wang, 1994), by setting the number of states to the minimum possible value found in the training set. These parameters were defined in previous research (Britto et al., 2003).

The isolated digits were organized in 195,000 samples for training (equally distributed into 19,500 samples for each of the ten classes), 28,000 for validation (both from hsf_{0,1,2,3}), and 60,089 samples for test (taken from hsf_7). See in Table 1.1 the number of states for each HMM representing a class of digit.

Table 1.1 The number of states of digit HMMs

| **Digit** | **# states** | | **Digit** | **# states** | | **Digit** | **# states** | | **Digit** | **# states** | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **col** | **row** | | **col** | **row** | | **col** | **row** | | **col** | **row** |
| **0** | 13 | 14 | **3** | 14 | 20 | **6** | 15 | 18 | **9** | 16 | 21 |
| **1** | 5 | 16 | **4** | 15 | 18 | **7** | 15 | 18 | | | |
| **2** | 14 | 16 | **5** | 13 | 19 | **8** | 14 | 20 | | | |

The uppercase letters were organized in 43,160 samples for training (equally distributed into 1,660 samples per class from the hsf_{0,1,2,3}), 11,941 images for validation from hsf_4, and 12,092 images for test from hsf_7 series. Table 1.2 shows the state configuration of the HMMs representing uppercase letters.

Table 1.2 The number of states of uppercase letter HMMs

| **Letter** | **# states** | | **Letter** | **# states** | | **Letter** | **# states** | | **Letter** | **# states** | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **col** | **row** | | **col** | **row** | | **col** | **row** | | **col** | **row** |
| **A** | 12 | 8 | **H** | 13 | 11 | **O** | 11 | 11 | **V** | 8 | 9 |
| **B** | 10 | 14 | **I** | 7 | 2 | **P** | 9 | 14 | **W** | 12 | 16 |
| **C** | 7 | 8 | **J** | 11 | 6 | **Q** | 15 | 20 | **X** | 17 | 12 |
| **D** | 11 | 9 | **K** | 18 | 15 | **R** | 15 | 13 | **Y** | 12 | 10 |
| **E** | 15 | 12 | **L** | 9 | 8 | **S** | 11 | 8 | **Z** | 16 | 11 |
| **F** | 11 | 11 | **M** | 12 | 14 | **T** | 11 | 9 | | | |
| **G** | 18 | 15 | **N** | 10 | 13 | **U** | 6 | 8 | | | |

We evaluated each algorithm in a simulated IL setting, in order to evaluate the evolution of each algorithm when chunks of data are presented one at a time. The IL setting was simulated by dividing the training databases into small blocks of data, with a homogeneous distribution of samples per class. For isolated digits, the training set was divided into 19 blocks of 10,000 samples (1,000 samples per class), and one block of 5,000 samples (500 samples per class). For uppercase letters, the training set was divided into 10 blocks of 4,316 samples (166 samples

per class). The classifiers' performances were evaluated on both the validation and test sets after learning each block of data.

Note that the validation dataset is not directly used by most of the algorithms for learning parameters, but this set was used to set some important configuration parameters for all of them. For instance, all the algorithms consider the same numbers of states and the same codebook for the HMMs, which were previously-defined by using a BL approach. Furthermore, IBW and IML have a constant $\varepsilon$, whose value was set after evaluating several values by training the system on the first data block and using the validation set for evaluation. And the same evaluation was done for ET and EN, to find the number of iterations for computing $\lambda'_t$ (worth noting that we also evaluated the use of the validation set to select the best models). Note that for both digits and letters, $\varepsilon$ was set equal to 0.00001 for IBW, and to 0.0001 for IML. For ET and EN, we set 10 fixed iterations for digits, and 15 for letters. A complete list of the use of the validation set by each algorithm can be found in Table 1.3 (note that the EN_stop is presented and justified further in this section).

Table 1.3 The use of the validation set by the algorithms

| **algorithm** | Hold-out validation | Number of states | Codebook | Configuration parameters | Stop criterion |
|---|---|---|---|---|---|
| **BL** | X | X | X | X | X |
| **IBW** | - | X | X | X | - |
| **IML** | - | X | X | X | - |
| **ET** | - | X | X | X | - |
| **EN** | - | X | X | X | - |
| **EN_stop** | - | X | X | X | X |

Also, we evaluated the performances of each algorithm to generate classifiers by learning the entire training set. The main objective was to check the differences in terms of performance among the algorithms when all the data is used for training.

Note that the experiments were repeated five times (except for BL due to computational complexity reasons), using different samples in each block. Consequently, the recognition rates and graphic curves are represented by the average of the five runs.

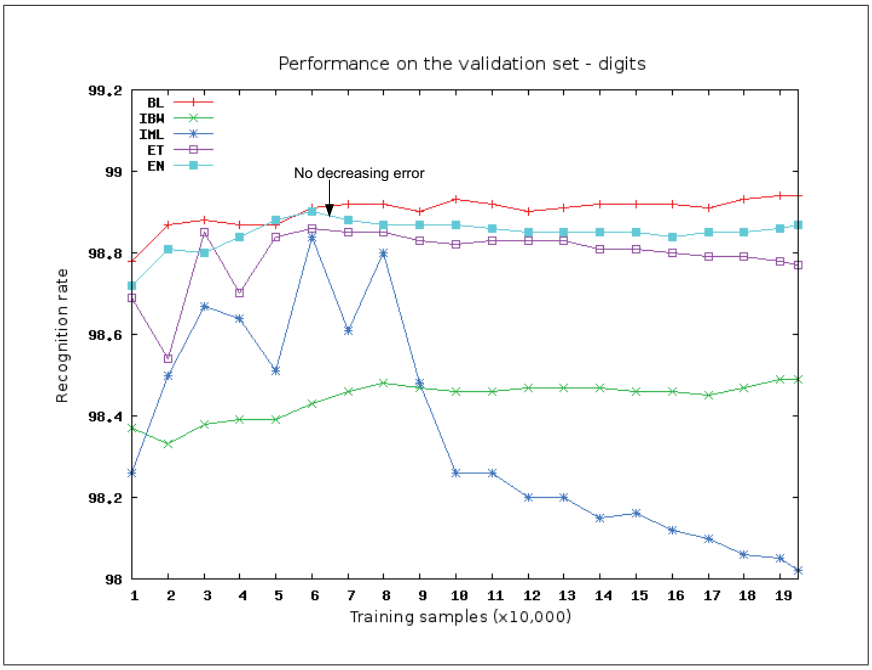



Figure 1.2 The recognition results of all the algorithms on the validation set, for digits

Figures 1.2 and 1.3 demonstrate the performance of each learning algorithm for isolated digits, on the validation and test sets respectively. These curves represent the performances of the classifiers taking into account a progressive growth of the training set, which consisted of presenting one block of data at a time to each learning algorithm. The blocks of data were created by the aforementioned simulation of IL. The results of the same experiments, reproduced for uppercase letters, are presented in Figures 1.6 and 1.7, on the validation and test sets respectively. In Figures 1.4 and 1.5 we present a complexity analysis of training and recognition for digits, and in Figures 1.8 and 1.9 we present the same analysis for uppercase letters.

For digits, BL presents the best overall performances, worth noting that EN has performances very near the BL. ET performed slightly inferior to both BL and EN, but it was significantly better than the other IL algorithms, e.g. IBW and IML. We see that the algorithms generally reach the best performances after learning six or seven blocks of data, which correspond to about 70,000 training samples. Despite a small decrease in performance after learning more blocks,

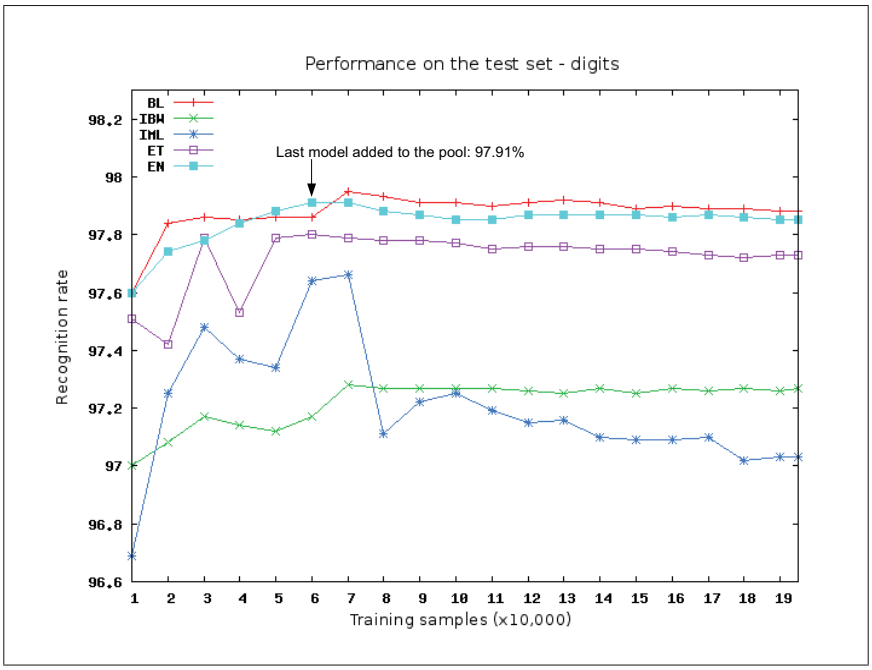



Figure 1.3 The recognition results of all the algorithms on the test set, for digits

all the algorithms (apart from IML) remain with stable performance after learning 70,000 samples. IML presents a significant decrease in performance after learning eight blocks, which suggests that a stop criterion must be employed with this algorithm to control its performance. Moreover, note that ET remains stable after learning 50,000 samples, which indicates that this algorithm reaches stable performances with less training samples than the other algorithms.

In terms of training complexity, BL is by far the most complex one, being around four times slower than both ET and EN. In considering the small difference in terms of performance among these algorithms, ET and EN are much more interesting algorithms when the resources for training are limited. Besides, both ET and EN can run on a parallel architecture, thus the training complexity can get an even more significant reduction.

In terms of recognition complexity, however, all the algorithms except EN present the same complexity. The recognition complexity of EN always increases after learning new data. However, we can control the recognition complexity of EN by evaluating the error on the validation

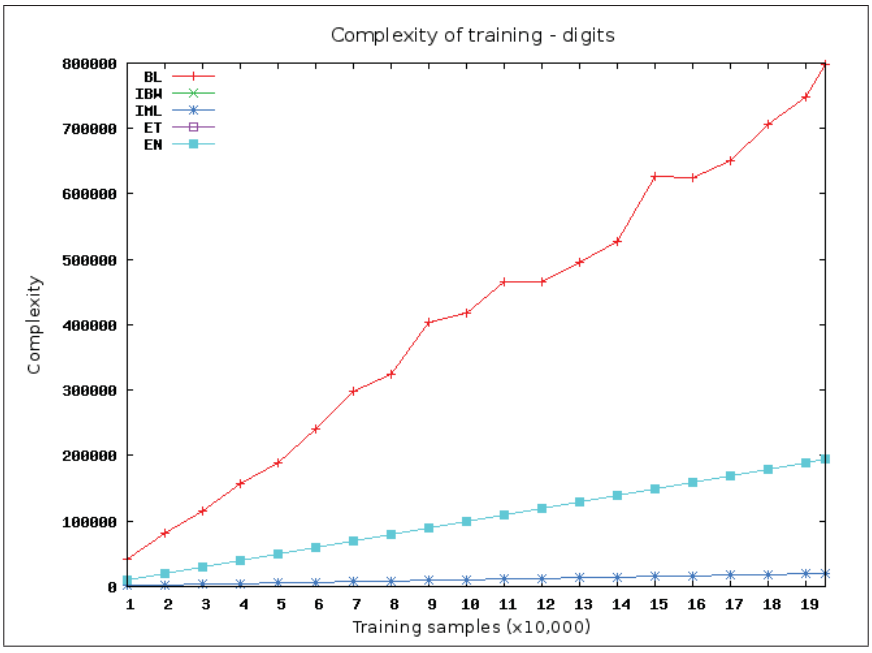



Figure 1.4 Training complexity for digits

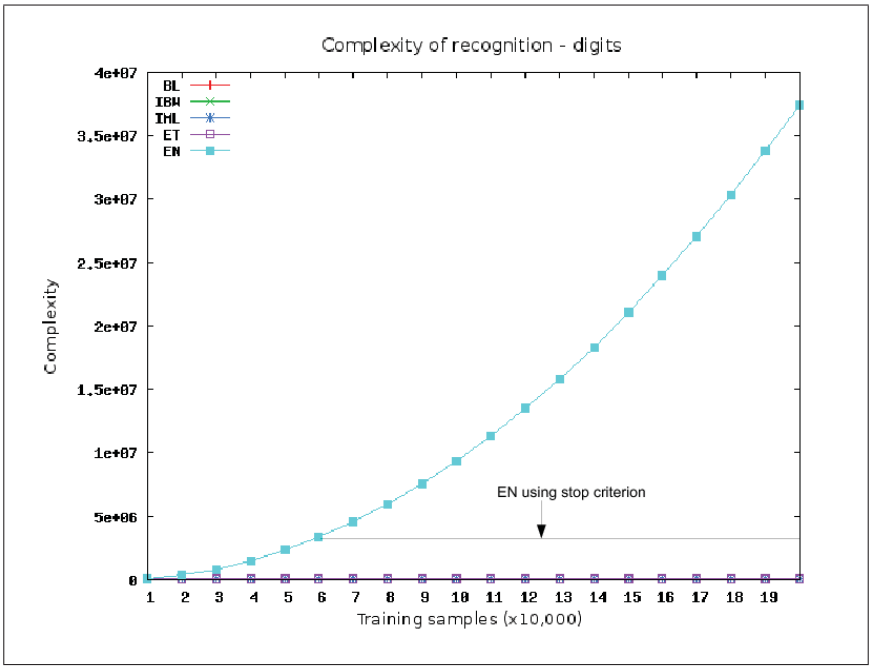



Figure 1.5 Recognition complexity for digits

set. For example, the use of a stop criterion, where new HMMs are added to $\zeta$ only when the error on the validation set decreases, is able to set a good performance-complexity trade-off for

EN. By observing Figure 1.2, we could stop including new HMMs after learning six blocks, and the performances remain at the same level of learning twenty blocks. But as we can see in Figure 1.5, the recognition complexity of learning only six blocks is significantly lower than the one of learning the whole training set.

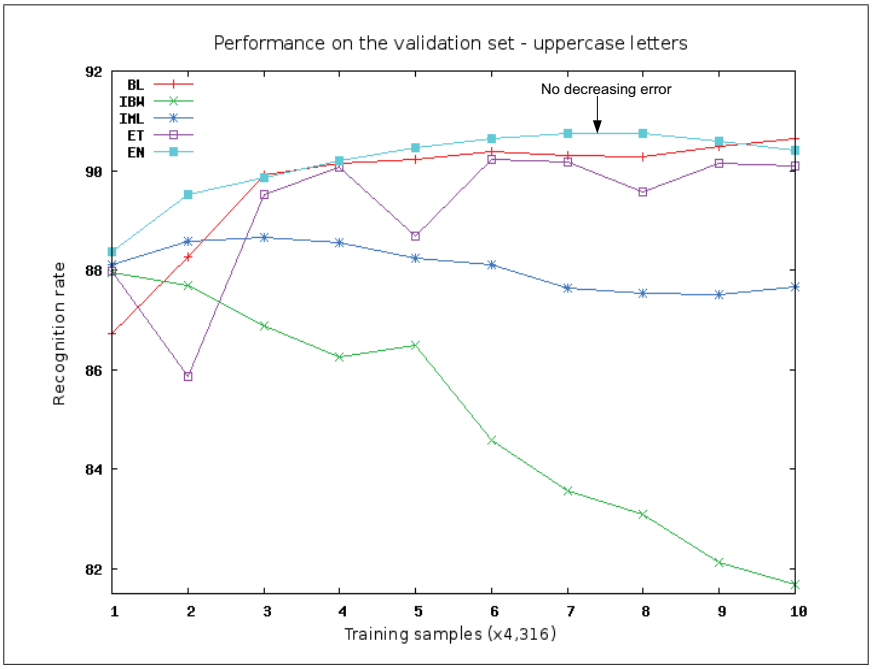



Figure 1.6 The recognition results of all the algorithms on the validation set, for uppercase letters

The experiments on uppercase letters showed a different scenario. The performances of EN surpassed the ones presented by BL. And ET presented performance almost as good as BL's. Again, both IBW and IML presented the worst performance. BL needed only four blocks of data to reach a stable learning point, and EN required about seven blocks. ET, in this case, did not present as stable performance as it did for digits. It is worth noting that the performance presented by EN starts to decrease after learning nine blocks, meanwhile BL always remains in a stable state.

Regarding complexity aspects, the same observed from digits is observed from letters. BL is the most complex algorithm for learning, and EN the most complex algorithm for recognition. We could, as mentioned before, stop including new HMMs to $\zeta$ when the error on the validation

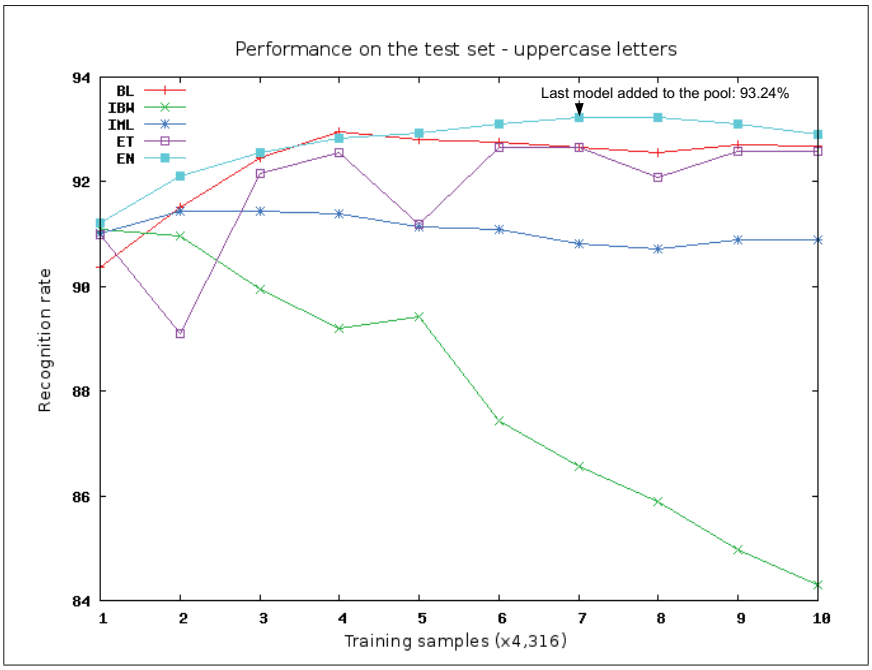



Figure 1.7 The recognition results of all the algorithms on the test set, for uppercase letters

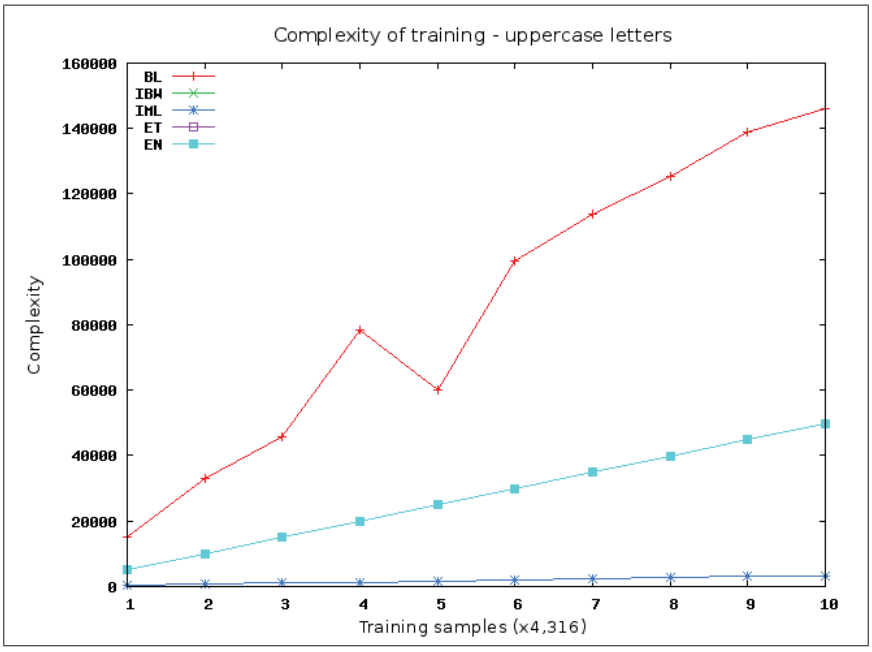



Figure 1.8 Training complexity for uppercase letters

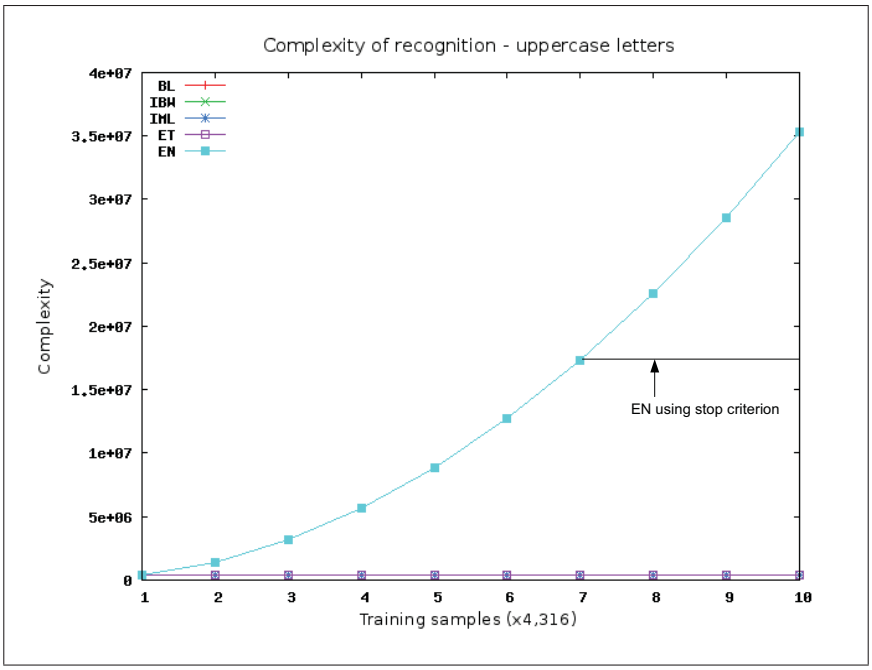



Figure 1.9 Recognition complexity for uppercase letters

set does not decrease. Thus, in this case we could stop it after learning seven blocks, which would result in a significant decrease of recognition complexity and in better recognition rates.

In Table 1.4 a summary of the results of the algorithms for learning the entire training set is presented. We also included the results of the EN algorithm with a stop criterion.

Table 1.4 A summary of the performances of the classifiers, in recognition rates, after learning the whole training set. Rejection rates correspond to the rate of samples that have been rejected. w. er.: error rate used to define the rejection thresholds on the validation set

| **Algorithm** | **Digits** | | | **Uppercase letters** | | |
|---|---|---|---|---|---|---|
| | **Val** | **Test no rejection** | **Reject w. er.:0.5%** | **Val** | **Test no rejection** | **Reject w. er.: 1.0%** |
| **BL** | **98.94** ±0.00 | 97.88 ±0.00 | 9.27 | 90.64 ±0.00 | 92.69 ±0.00 | 46.00 |
| **IBW** | 98.49 ±0.10 | 97.27 ±0.08 | - | 81.68 ±4.97 | 84.29 ±4.79 | |
| **IML** | 98.02 ±0.21 | 97.03 ±0.11 | - | 87.67 ±0.43 | 90.90 ±0.65 | - |
| **ET** | 98.77 ±0.02 | 97.73 ±0.01 | 9.63 | 90.10 ±0.05 | 92.57 ±0.12 | 42.09 |
| **EN** | 98.87 ±0.03 | 97.85 ±0.01 | **7.92** | 90.40 ±0.11 | 92.91 ±0.13 | 39.72 |
| **EN_stop** | 98.90 ±0.02 | **97.91** ±0.03 | 8.01 | **90.75** ±0.13 | **93.24** ±0.07 | **39.36** |

For isolated digits the original EN algorithm is overperformed by a BL setting, where the latter provided recognition rates on the validation set of 98.94%, and 97.88% on the test set. The recognition rates presented by EN were 98.87% on the validation set, and 97.85% on the test set. ET performed slightly worse than both EN and BL, presenting 98.77% on the validation set, and 97.73% on the test set. IBW and IML were the worst algorithms in these experiments, having the recognition rates of IBW 98.49% on the validation set and 97.27% on the test set, and the recognition rates of IML 98.02% on the validation set and 97.03% on the test set. The modified EN presented the best recognition rates, being 98.90% on the validation set, and 97.91% on the test set.

In the uppercase letter problem, the recognition rates presented by the classifiers designed in a BL setting were 90.64% and 92.69%, on the validation set and the test set respectively. EN performed better than BL, whose recognition rates were 90.40% on the validation set, and 92.91% on the test set. ET provided performances very close to BL, with 90.10% of recognition rates on the validation set, and 92.57% on the test set. Similar to the experiments with digits, IBW and IML were the worst algorithms, but in this case IBW performed worse than IML. The latter presented recognition rates of 87.67% on the validation set, and 90.90% on the test set, and the former presented only 81.68% on the validation set, and 84.29% on the test set. EN employing a stop criterion presented the best performance again, with 90.75% of recognition rates on the validation set, and 93.24% on the test set.

In order to provide a more complete evaluation of the algorithms, we also evaluated the rejection rates of the four best algorithms for each problem. For such an evaluation, we took the best classifiers generated from the numerous runs of each algorithm.

Figure 1.10 shows the error-reject evaluation for isolated digits. We can see that both BL and ET presented very similar reject rates, meaning that both algorithms presents similar reliability. Also, both EN and its modified version presented similar reject curves too, being the algorithms with the highest reliability in this problem.

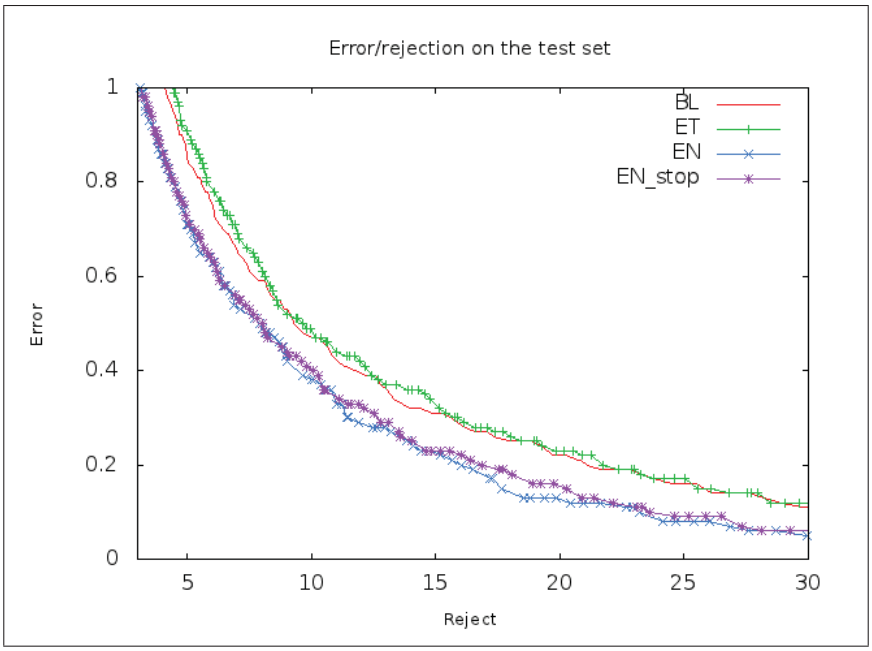



Figure 1.10 Error-reject analysis of batch learning (BL), ensemble training (ET), ensemble training using EoHMMs (EN), and EN with a stop criterion, for digits

Figure 1.11 shows the error-reject evaluation for uppercase letters. Note that despite providing slight lower recognition rates than BL at the zero-level reject experiments, the ET algorithm provided lower rejection rates for some error rates. The EN algorithm provided much lower rejection rates than both BL and ET, which shows that the former is really a robust learning approach. Furthermore, the EN algorithm with a stop criterion was able to present the same rejection rates of EN.

Considering other methods applied on the same isolated digit dataset, we can find results varying from 99.16% to 99.37% (Oliveira et al., 2002; Oliveira and Sabourin, 2004; Milgram et al., 2006; Radtke et al., 2006), using classifiers such as MLP and SVM, and ensembles of MLP as well, in a BL setting. In spite of HMMs performing worse than other classifiers in this task, some recent research with EoHMMs demonstrated that HMM-based classifiers, with improved codebooks, the recognition rates can be increased from 98.00% (Britto, 2001) to 98.86% (Ko et al., 2007). By considering the learning approaches presented in this work, we believe that

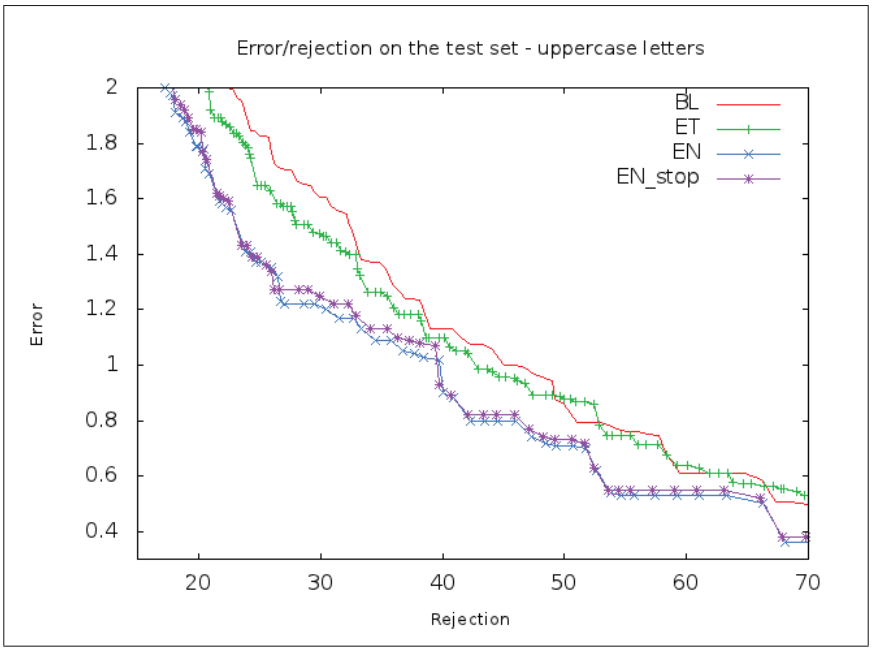



Figure 1.11 Error-reject analysis of batch learning (BL), ensemble training (ET), ensemble training using EoHMMs (EN), and EN with a stop criterion, for uppercase letters

the performances of the latter can be enhanced further by employing the EN algorithm, since the diversity of the HMMs in the ensemble will be increased significantly.

Considering the uppercase letter problem, our method also provides lower recognition rates than some methods in the literature. We find results varying from 94.16% using ensemble of KNN (Dos Santos et al., 2008), 95.00% and 95.98% using MLP and ensemble of MLP (Radtke, 2006), respectively, and 96.82% using SVM (Milgram et al., 2006). Notice that HMMs are generally known to be less accurate than MLP and SVM in problems such as the recognition of isolated digits and letters due to its greater sensitivity to noise. However, HMMs model continuous signals, which make them a very interesting approach to model problems than can be decomposed into relatively simpler problems, such as the recognition of numeral strings and words, which can be decomposed into digits and letters. HMMs can be adjusted to segment complex signals by means of training, instead of relying on heuristic approaches such as MLP and SVM. Furthermore, the use of optimization methods, such as the ones used in (Ko et al.,

2007), can lead to improve the accuracy of HMMs in isolated characters, which can also result in much better performance in the aforementioned more complex problems.

## 1.5 Conclusions and Future Work

In this work we presented the evaluation of four different IL algorithms for HMMs, and compared their performance with the traditional Baum-Welch algorithm in a BL setting. Two handwritten isolated character recognition problems were considered.

The experiments showed that BL performs slightly better that IL algorithms for isolated digits, but is outperformed by EN for uppercase letters, when a validation set is not used to control learning. However, when a validation set is considered by the EN algorithm, this algorithm can provide better performance than BL, and at the same time the recognition complexity of EN can be reduced. These results indicate that it is possible to employ IL algorithms to design complex pattern recognition systems, and reach higher reliability than BL algorithms. Furthermore, the use of external knowledge (e.g. validation) also seems to contribute to improve the generalization performances of the classifiers.

We can pursue this work in several directions. One starting point is the investigation of other methods for reducing the recognition complexity of EN, which can also result in increasing performances. One promising approach is to employ both EN and ET algorithms in a single IL framework. By doing that, we can set, for instance, a fixed number of HMMs for the EN algorithm, where each one is learned using the ET algorithm. In theory, such a framework is able to counter-balance the advantages and disadvantages of both EN and ET, and can provide better performance than BL, with lower training complexity.

Another aspect to be investigated is how to decrease the use of the external knowledge (e.g. the validation set) for ET and EN (see in Table 1.3 for which aspects these algorithms needed validation). For example, the use of a k-fold cross-validation would be useful to determine the number of iterations to train each block of data, for ET and EN, which is one of the configuration parameters of these algorithms. Furthermore, topology learning could be employed to

determine the best HMM topology from each block (Florez-Larrahondo, 2005). In addition, we can also investigate techniques to optimize the codebooks from each block.

## 1.6 Discussion

In this chapter we demonstrate that ensembles of classifiers (EoCs) constitute a viable solution to the definition of IL algorithms. Such algorithms can incorporate new knowledge by including new members in an existent pool of classifiers. In this case, the parameters of existing members remain static, keeping previously-learned knowledge intact. It is worth noting that similar conclusions have been drawn in papers related to this work (Khreich et al., 2012).

However, there are some pitfalls to the static combination of classifiers used during generalization. For instance, some classifiers might be more competent than others in recognizing a given test sample. Also, some classifiers may become obsolete over time, that is, they may not be as competent as they were in the past for many test samples. For this reason, a better solution might be to use dynamic selection (DS) during operation.

Even though a proof-of-concept using HMMs has been presented, EoC-based algorithms are general and can be applied to other types of classifiers. Most algorithms dynamically select classifiers by taking into account the input feature set. These approaches are restricted to feature-based classifiers, however. They are not suitable for HMMs, since classifiers based on these models rely on observation sequences instead of feature vectors of fixed size.

In the next chapter, we present investigations on DS algorithms that can select classifiers by evaluating only the outputs yielded by the members of the pool of classifiers. These investigations are aimed at defining an adaptive framework that is suitable for a broader range of classifiers.

# CHAPTER 2

# DYNAMIC SELECTION APPROACHES FOR MULTIPLE CLASSIFIER SYSTEMS

In this chapter, we propose a new approach for dynamic selection of ensembles of classifiers. Based on the so called multistage organizations concept, the main objective of which is to define a multilayer fusion function adapted to each recognition problem, we propose dynamic multistage organization (DMO), which defines the best multistage structure for each test sample. By extending Dos Santos et al's approach, we propose two implementations for DMO, namely DSA$^m$ and DSA$^c$. While the former considers a set of DS functions to generalize a DMO structure, the latter considers contextual information, as represented by the output profiles computed from the validation dataset, to perform this task. The experimental evaluation, considering both small and large datasets, demonstrates that DSA$^c$ outperforms DSA$^m$ on most problems, showing that the use of contextual information can achieve better performance than other methods. In addition, the performance of DSA$^c$ can also be enhanced in IL. However, the most important observation, supported by additional experiments, is that DS is generally preferred over static approaches when the recognition problem presents a high level of uncertainty.

## 2.1 Introduction

Over the past decades, Multiple Classifier Systems have emerged as a viable alternative to make pattern recognition systems achieve lower and lower error rates. This kind of system can be composed of either existing classifiers, aiming at enhancing their individual performances, or classifiers constructed by an automatic method, to which we refer as ensembles of classifiers (EoCs). In both cases, nonetheless, it is well-known that the set of classifiers must contain members that are complementary and diverse (Brown et al., 2005; Dos Santos et al., 2006), so that the combined classifiers outperform the best member of the set.

The task of finding the aforementioned complementary and diverse set of classifiers is not trivial. Actually, the performance of the fusion function, which carries out the combination of the

decisions provided by the base classifiers, may heavily depend on such a “good” set of classifiers (Shipp and Kuncheva, 2002). For example, it has been shown that the performance of the majority voting function, which is a widely used combination rule, significantly improves for the case of negatively correlated classifiers (Kuncheva et al., 2003; Ruta and Gabrys, 2002). However, to construct an EoC with negatively correlated classifiers remains a very unlikely situation in real-world classification problems, and their benefits remain out of reach. If existing classifiers, to which we have no access to change their parameters, are included in the pool, this task may become even less evident.

One way to enhance the use of multiple classifiers is to define a fusion scheme that takes greater advantage of the diversity presented by the base classifiers, even though such a diversity is not so apparent at first. In other words, we need to define a way to expand the limits of the combination method, to better use the existing diversity of the pool of classifiers. One interesting approach, named multistage organizations (MO), has been proposed in (Ruta and Gabrys, 2002, 2005) for such an objective.

The main advantage of using MO relies on the ability to construct a multistage structure, which represents the fusion function, that is adapted to each recognition problem. Such an adaptation is achieved by defining the relationships between consecutive layers based on evidences provided by the training data. Nevertheless, only a single structure is created, in an ad-hoc fashion, for all the test samples. Due to its static nature, the method might not be able to handle all the difficulties presented by complex recognition patterns, which supposedly has the same drawback of static approaches to select classifiers.

To deal with those issues, we propose dynamic multistage organizations (DMO), inspired by dynamic selection of classifiers. The main idea consists of defining the multistage structure that best adapts to each test sample. In this case, not only the fusion function adapts to each problem, but also, to each test sample. Such a structure also takes into account an automatic weighting approach, which selects the best weight for each classifier output based on the current test sample.

One approach that is closely related to the idea of DMO is Dos Santos et al's (DSA) approach (Dos Santos et al., 2008). In this case, one EoC is dynamically selected, from a pool of EoCs, by means of evaluating only the outputs yielded by the members of each ensemble. If we can, for example, select more than one ensemble at a time, we can better generalize the DMO concept, by implementing a two-stage DMO structure. Given these standpoints, we propose two original frameworks based on DSA.

The first framework, named DSA$^m$, consists of validating the DMO concept, in which we exploit the use of a set of dynamic selection functions to create a DMO structure. In this case, each function performs the selection of an EoC. Note that the main advantage of this method lies on its simplicity. In the second framework, namely DSA$^c$, we use contextual information to find the best DMO structure based on problem-related knowledge. The evidences produced by the validation set are taken into account in this case, whereas the structure is defined by considering the most similar validation samples using case-based reasoning. The architecture of DSA$^c$ is not only easily adaptable to different problems, but also is incremental-learning ready.

This work aims at accomplishing two main objectives during the experimental evaluation. The first objective is to evaluate both DSA$^c$ and DSA$^m$ against static methods, to observe whether the proposed DMO concept can result in better performance or not. In addition, we aim at evaluating the conditions under which dynamic selection might outperform static selection. Given that in the literature dynamic selection methods are generally compared to static methods for recognizing a given problem, in a single static condition in terms of recognition problem, the goal of these experiments is to provide more insights related to which conditions a dynamic selection approach might be more preferable than a static one. The *NIST-digits* database allows us to simulate these different conditions, as explained later.

The remainder of this chapter is organized as follows. In Section 2.2, we describe static and dynamic selections, providing more details about Dos Santos et al's approach, to support the content of the subsequent sections. In Section 2.3 we describe the proposed DMO concept

with greater detail. Both DSA$^c$ and DSA$^m$ are described in Section 2.4, and in Section 2.5, we present the experimental protocol and the results that were obtained. Finally, in Section 2.6, we present conclusions and point out the future work.

## 2.2 Background theory

In this section, we present an overview of dynamic selection methods (DS), in which we also describe Dos Santos et al's approach (DSA) in detail.

### 2.2.1 Dynamic Selection (DS)

Suppose a multiple classifier system is composed of a pool of base classifiers, to which we refer as $C$. The goal of dynamic selection is to find a subset of classifiers $C'_i$, where $C'_i \subset C$, which is the best one, by considering all local criteria, to classify the test sample $x_{i,test}$. Note that, in static selection, a single subset $C'$, where $C' \subset C$, is globally selected to recognize all test samples.

In the literature, dynamic selection is divided into dynamic selection of classifiers (DSC), where only a single classifier is selected for each test sample (Woods et al., 1997; Giacinto and Roli, 2001; Zhu et al., 2004), and dynamic selection of ensembles of classifiers (DSEoC), where an EoC is selected for each test sample (Dos Santos et al., 2008; Soares et al., 2006; Ko et al., 2008).

Usually, the main goal of the systems for both DSC and DSEoC is to find the best subset of classifiers $C'_i$ to classify $x_{i,test}$. This best set is generally associated with the highest level of competence, which is computed by means of, for instance, K nearest neighbors (Woods et al., 1997), clustering (Kuncheva, 2000), and multiple training datasets (Singh and Singh, 2005). In order to compute the level of competence by using one of these methods, we must deal with the following issues: a robust feature set must be defined for a desirable reliability, which is not trivial; these approaches are very expensive in terms of computational complexity; and it is not possible to use some types of base classifiers, such as human experts or HMMs, since they

do not use feature vectors to conduct the classification task. The KNORA algorithm (Ko et al., 2008), however, is an example of an approach that tries to overcome some of these issues. The only information this method requires from the base classifiers is whether or not they correctly classify a given validation sample. Nonetheless, KNORA also depends on a very robust feature set to compute similarity between validation samples and the test sample.

A more general approach, though, is Dos Santos et al's, which dynamically selects EoCs, whose levels of competence are computed by using only the outputs of their members, based on the extent of consensus. This property makes it a very general approach in terms of base classifier and feature set. However, many sources of knowledge embedded in the structure of DSA have not been exploited yet, for instance, the outputs produced by the base classifiers. Thus, we believe the performance of this method can be improved, resulting in an approach that is both robust and general at the same time. For the sake of completeness, in the remainder of this section we present this method in greater detail.

### 2.2.2 Dos Santos et al's approach (DSA)

The overall architecture of DSA is depicted in Figure 2.1. The main objective of this method is to dynamically find the best EoC, whose members are a subset of $C = \{c_1, c_2, \ldots, c_N\}$, to recognize the test sample $x_{i,test}$. This task is performed by considering only the recognition outputs $O_i = \{o_{i,1}, \ldots, o_{i,N}\}$ computed from $C$. Each output corresponds to a class label from the set $\Omega = \{\omega_1, \ldots, \omega_M\}$.

DSA is divided into two phases: the design phase and the operational phase.

During the design phase, which is performed off-line, it creates the architecture that supports the dynamic selection of EoCs. In other words, the pool of EoCs $C^{*\prime} = \{C'_1, \ldots, C'_W\}$, where $C'_j \subset C, 1 \leq j \leq W$, is created during this phase. Given that $C^{*\prime}$ is a subset of all possible EoCs $C^*$, the main objective is to reduce the complexity for the operational phase since $|C^*|$ is much larger than $|C^{*\prime}|$ and the time needed to find the best EoCs in considering $C^*$ would be impractical in most applications. The pool $C^{*\prime}$ is generated by a search algorithm, which

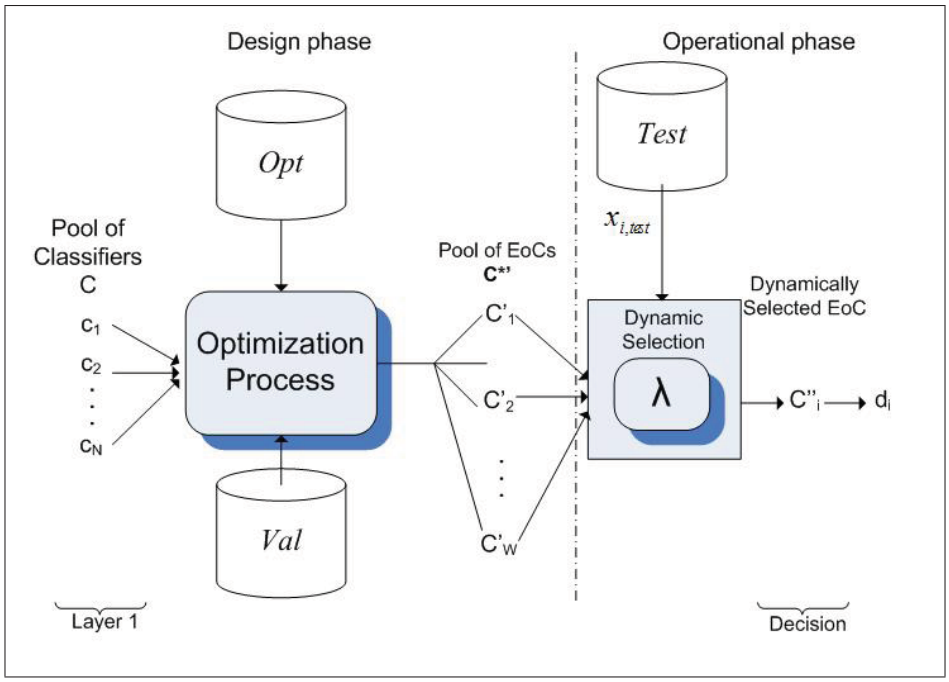



Figure 2.1 Dos Santos et al's approach (DSA). The pool of classifiers is organized into another pool of EoCs during the design phase. During the operational phase, the EoC, which is dynamically selected by $\lambda$, produces the final decision

is a genetic algorithm in this work. Each individual is represented by a binary vector of $N$ positions, where each bit represents whether or not a classifier is selected as a member of an EoC. The fitness function, which has to be minimized, uses the error rate on the optimization set $Opt$, by applying the majority voting method on the EoCs assigned by each individual. In order to avoid overfitting, each individual is also evaluated on the validation set $Val$, and the best solutions are saved into an archive whose size is $W$. The archive is then used as $C^{*\prime}$.

Throughout the operational phase, the dynamic selection of the best EoC $C''_i$ is performed, which consists of a member of the pool of EoCs $C^{*\prime}$, to recognize the test sample $x_{i,test}$. After the outputs $O_i$ of the set of base classifiers $C$ are computed, we check which member of the pool of EoCs $C^{*\prime}$ is best to recognize $x_{i,test}$. For each EoC, we apply the dynamic selection function $\lambda$ to evaluate whether it is the best ensemble or not. The best EoC is then stored in $C''_i$, the dynamically selected EoC. Finally, the ensemble that was dynamically selected is used to compute the class with the highest number of votes, which is the final decision $d_i$.

Note that $\lambda$ can be related to one of the five functions described in Section 2.4.1.1. In this work, $\lambda$ is computed by taking into account the extent of consensus, as defined in Equation 2.2 (Dos Santos et al., 2008).

## 2.3 Dynamic Multistage Organizations (DMO)

The main inspiration for dynamic multistage organizations is multistage organizations (MO). MO consists of structuring classifiers into relevant multistage layers. The outputs of the classifiers are reorganized into subsequent levels, and these outputs are re-evaluated at each level. By structuring classifiers in multi-steps, the main premise is that the influence of individual errors on the final error of the combined systems can be reduced, since the outputs are transformed to another space corresponding to the fusion of some selected classifiers. Hence, given the fact that both selection and fusion are conducted at the same time, the diversity among the classifiers is better exploited, and the limits of majority voting error are widened.

The main advantage of MO is that the whole structure can be defined for a given problem. For example, in (Ruta and Gabrys, 2002) a genetic algorithm is used to optimize the MO structure given problem-related training data. Nonetheless, a single structure is defined for all test samples, which, as a consequence, might not cover the different difficulties presented by all test samples in a complex recognition problem. To deal with this issue, we propose DMO, inspired by dynamic selection of classifiers.

DMO basically consists of defining the best multistage structure for each test sample. In this case, the relationships between the outputs are dynamically defined, according to the current test sample $x_{i,test}$. It also takes into account a dynamic weighting approach for further improvements. Note that, instead of using the same structure to recognize all test samples, which might be suboptimal, we define the structure that better models the relationships among the base classifiers, according to the information provided by $x_{i,test}$. By doing so, we may enhance the overall performance of the system not only by using a multi-stage approach, but also by using a dynamic approach that better fits the difficulties presented by each test sample.

In order to illustrate DMO, we use a synthetic recognition example with five binary classifiers. In Figure 2.2(a), we present a test sample, whose correct label is 1, being recognized by MO. Suppose this MO structure has been considered optimal during the design phase. We can see, though, that this structure does not correctly recognize this test sample. However, as shown in Figure 2.2(b), by using a DMO approach, we might be able to define a MO structure specifically for this test sample, which can correctly compute the correct class. In this case, given that an EoC that provides the correct answer is selected twice (i.e. it has a heavier weight) to compose the final layer, the correct answer is successfully computed.

One existing method that partially implements the DMO concept is Dos Santos et al's approach (see Section 2.2.2), as depicted in Figure 2.2(c). In this case, one EoC is dynamically selected, from a pool of EoCs, by means of evaluating only the outputs yielded by the members of each ensemble. If we can, for example, select more than one EoC at a time, we can better generalize the DMO concept, by implementing a two-stage DMO structure. For this reason, we extend the architecture of DSA to implement DMO.

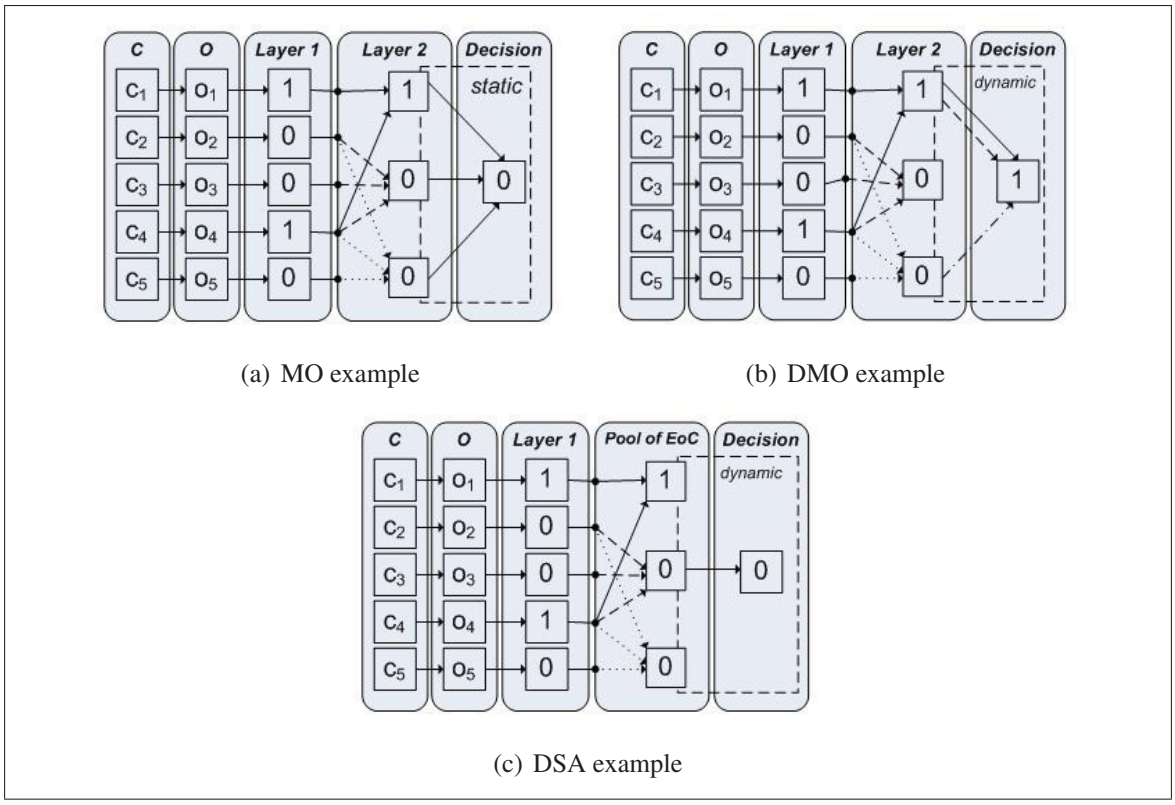
## 2.4 Extending Dos Santos et al's Approach to Implement DMO

We propose two methods to extend Dos Santos et al's approach to implement a dynamic multi-stage organization. These methods, named DSA$^m$ and DSA$^c$ respectively, are described in the following sections.

### 2.4.1 DSA$^m$: introducing DMO and high-level decision making

The first framework consists of adding two main extensions to DSA. We refer to this framework as DSA$^m$, since the use of multiple dynamic selection functions has enabled the implementation of the first extension.

The first extension consists of characterizing the main DSA structure as dynamic multistage organizations. Instead of selecting a single EoC, as in DSA, we now have to select a set of EoCs. The main idea is to compose the second layer of a DMO structure by using this set of




(a) MO example

(b) DMO example

(c) DSA example

Figure 2.2 (a) The sequence of stages processed by multistage organizations (MO), for an example with five classifiers with binary outputs. In this case, each member of layer 2 always provides one vote for the final decision. (b) The same example with dynamic multistage organization (DMO), whereas a member from layer 2 may provide none, one, or more than one vote. (c) The same example using Dos Santos et al's approach (DSA), where only a single member of layer 2 gives a vote. Class 1 is the right output in this example

EoCs. To achieve this task, we adapt some components of the operational phase. To recognize $x_{i,test}$ we select the set of EoCs $C^{*\prime\prime}_i = \{C''_{i,1}, \ldots, C''_{i_U}\}$, as presented in Figure 2.3.

Algorithm 2.1 describes each step of the proposed method. Once the outputs of the base classifiers $O_i$ are computed in step 2, we evaluate each EoC individually. By considering the set of functions $\Lambda = \{\lambda_1, \ldots, \lambda_U\}$, we evaluate each member of $C^{*\prime}$. The best EoCs, according to $\Lambda$, form the set of dynamically selected EoCs $C^{*\prime\prime}_i$. Note that $|C^{*\prime\prime}_i| = U$, since each $\lambda_k$ selects an EoC, i.e. $C''_{i,k}$. It is also worth noting that an EoC may be selected more than once, which results in the automatic weighting approach demonstrated in Figure 2.2(b). In this case all the

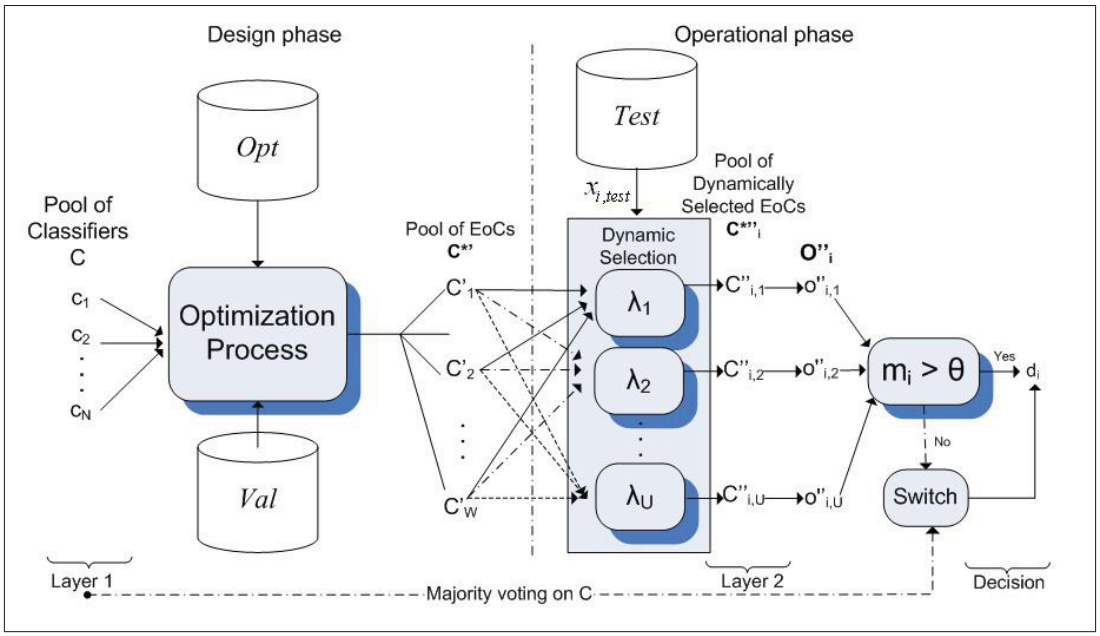



Figure 2.3 An overview of the DSA$^m$ approach. This method uses the set of dynamic selection functions $\Lambda$ to dynamically select a set of EoCs, which results in a two-layer DMO structure

functions described in Section 2.4.1.1 are used to compose $\Lambda$, thus $\Lambda = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\}$ and $|\Lambda| = 5$. In the example presented in Figure 2.2(b), in contrast, we consider $|\Lambda| = 3$.

After $C^{*\prime\prime}_i$, the set of dynamically selected EoCs, is defined, the outputs of these EoCs $O''_i = \{o''_{i,1}, \ldots, o''_{i,U}\}$ are computed (step 16 of Alg. 2.1). These outputs represent the majority voting class computed from each member in $C^{*\prime\prime}_i$. Then, $O''_i$ is submitted to the switch module. The proposed switch mechanism represents the second extension to DSA. This mechanism, which is represented by steps 18 to 22 in Algorithm 2.1, is explained in detail in the following paragraphs.

Despite the expected improvements that a dynamic multistage structure can bring to DSA, we have no guarantee that this complex structure is really better than the pool of base classifiers. In some cases, for example, the dynamically selected EoCs, i.e. $C^{*\prime\prime}_i$, might provide low-confidence results, yielding a tie or the answers below some acceptable confidence level. Note

Algorithm 2.1 $DSA^m$. $best\_score(k)$ and $score(k)_{j,i}$ represent temporary variables to compute the best EoC, for each of the five functions presented in Section 2.4.1.1

1: **for** each data point $x_{i,test}$ on $Test$ **do**
2: Compute $O_i = \{o_1, \ldots, o_N\}$ by considering $C = \{c_1, \ldots, c_N\}$
3: Initialize $best\_score(k)$, $\forall \lambda_k$ in $\Lambda$
4: **for** each $C'_j$ in $C^{*'}$ **do**
5: **for** each $\lambda_k$ in $\Lambda$ **do**
6: Compute $score(k)_{j,i}$ by considering $\lambda_k$.
7: **if** $score(k)_{j,i}$ is better than $best\_score(k)$ **then**
8: $C''_{i,k} = C'_j$
9: $best\_score(k) = score(k)_{j,i}$
10: **end if**
11: **end for**
12: **end for**
13: **for** each $\lambda_k$ in $\Lambda$ **do**
14: $o''_{i,k}$ = most voted class from $C''_{i,k}$
15: **end for**
16: Compute $m_i$ from $O''_i = \{o''_{i,1}, \ldots, o''_{i,U}\}$ *# see Equation 2.1*
17: *# Switch mechanism*
18: **if** $m_i > \theta$ **then**
19: $d_i$ = most voted class from $O''_i$
20: **else**
21: $d_i$ = most voted class from $C$
22: **end if**
23: **end for**

that it is important to detect these cases to avoid random decisions, and select a better source of knowledge, that may be the base classifiers. For this reason, we propose a switch mechanism.

Here is the main idea of the switch. First, we employ the concept of margin (Hansen et al., 1997) (see Equation 2.1, where $v1_i$ and $v2_i$ are, respectively, the most voted and the second most voted classes for $x_{i,test}$) to identify whether or not the answers provided by $C^{*''}_i$ are confident enough, as shown in step 18 of Algorithm 2.1. When the margin $m_i$ computed by the outputs $C^{*''}_i$ is above the threshold $\theta$, e.g. $m_i > \theta$, we consider that the dynamically selected EoCs are reliable enough and simply use the most voted class in considering $O''_i$ as the final decision $d_i$ (step 19). In contrast, when $m_i \leq \theta$, we switch to the pool of base classifiers and use $O_i$, i.e.

the outputs of $C$, to compute the most voted class (step 21). This most voted class is used as the final decision $d_i$. Note that one advantage of the switch mechanism is that instead of relying on random guess, since in the case of a tie we would have to randomly pick one class as the final decision, we use another source of knowledge that is embedded in the architecture of the system to compute such a decision.

$$m_i = v1_i - v2_i \tag{2.1}$$

In the next section, we describe the dynamic selection functions that are used in step 6 of Algorithm 2.1 to compute the corresponding score of each $\lambda_k$.

#### 2.4.1.1 Consensus-based dynamic selection functions

The five functions involved in this work, are computed by taking into account the number of votes for each class in $\Omega$, provided by each candidate $C'_i$. We aimed at using only functions which can compute the level of competence of each EoC based on the votes of the base classifiers. One reason is to avoid the complexity of functions that compute regions of competence based on evaluating distances between $x_{i,test}$ and prototypes in the feature space, as in (Woods et al., 1997; Soares et al., 2006). Another reason is to enable this approach to deal with any category of base classifier that can output votes.

In this section we use the following notation: $v_{k,j,i}$ is the number of votes for class $\omega_k$ provided by $C'_j$ given the test sample $x_{i,test}$, $p_j$ is the global performance of $C'_j$, and $p_{j,k}$ is the performance of $C'_j$ for class $\omega_k$, both measured on the validation set $Val$; $mv_{j,i}$ represents the majority voting class provided by $C'_j$ given the sample $x_{i,test}$, e.g. $mv_{j,i} = \text{argmax}\ v_{k,j,i}\ \forall k$. The cardinality of $C'_j$ is represented by $|C'_j|$.

#### 2.4.1.1.1 $\lambda_1$: Ambiguity-guided dynamic selection (ADS)

This function is presented in (Dos Santos et al., 2008). It selects the solution whose outputs produce the lowest ambiguity, represented by the number of classifiers in disagreement with the majority voting class.

The ambiguity $\gamma_{j,i}$, given $C'_j$ and the test sample $x_{i,test}$, can be computed by the minimization of the following equation:

$$\gamma_{j,i} = \frac{\sum_1^k v_{k,j,i}}{|C'_j|}, \text{ where } k \neq mv_{j,i} \tag{2.2}$$

#### 2.4.1.1.2 $\lambda_2$: Margin-based dynamic selection (MDS)

This function selects the solution with the highest margin (Dos Santos et al., 2008). The margin represents the difference between the majority voting and the second highest number of votes. The main idea is to select the solution that produces the largest difference in number of votes between the highest consensus and the second highest.

The maximization of the following equation, given $C'_j$ and the sample $x_{i,test}$, allows us to dynamically select the most competent candidate by using the margin $\mu_{j,i}$:

$$\mu_{j,i} = \frac{v_{k,j,i} - \max_{l \neq k} v_{l,j,i}}{|C'_j|}, \text{ where } k = mv_{j,i} \tag{2.3}$$

#### 2.4.1.1.3 $\lambda_3$: Class-strength dynamic selection (CSDS)

This function weights the selection of the best solution (Dos Santos et al., 2008). In this case, the margin, as described in Equation 2.3, is multiplied by $p_{j,k}$. The main idea is to select the candidate that provides the best trade-off between the margin and the performance for recognizing the class with the highest number of votes.

In considering the margin as $\mu_{j,i}$ and the class performance as $p_{j,k}$, the maximization of the following equation leads us to find the most competent $C'_j$ for $x_{i,test}$ by using CSDS:

$$\Theta_{j,i} = \mu_{j,i} * p_{j,k}, \text{ where } k = mv_{j,i} \tag{2.4}$$

#### 2.4.1.1.4 $\lambda_4$: Pair of votes dynamic selection (PVDS)

We propose a new function aiming at selecting EoCs that concentrate their decisions on only two classes. In this case, both values for margin and consensus might be very low, which is counter-intuitive according to other DSFs such as ADS and MDS. However, we suppose that these EoCs are likely to produce less random guesses and wrong decisions, since they concentrate their decisions on reduced boundaries, e.g. only two classes.

In order to implement this idea, we simply sum the number of votes for the top-two classes, and maximize this value. This is represented by $\eta_{j,i}$. Given $C'_j$ and the sample $x_{i,test}$, $\eta_{j,i}$ can be computed by using the following equation:

$$\eta_{j,i} = \frac{v_{k,j,i} + \max_{l \neq k} v_{l,j,i}}{|C'_j|}, \text{ where } k = mv_{j,i} \tag{2.5}$$

#### 2.4.1.1.5 $\lambda_5$: Global-strength dynamic selection (GSDS)

This function is a modification of CSDS. In this case, we consider the global performance $p_j$ of $C'_j$ to weigh the value provided by the margin. The main supposition is that the global performance is more robust than the performance to recognize a specific class to indicate the most competent solution.

Given $p_j$, $C'_j$, and $x_{i,test}$, this function can be computed by maximizing the following equation:

$$\iota_{j,i} = \mu_{j,i} * p_j \tag{2.6}$$

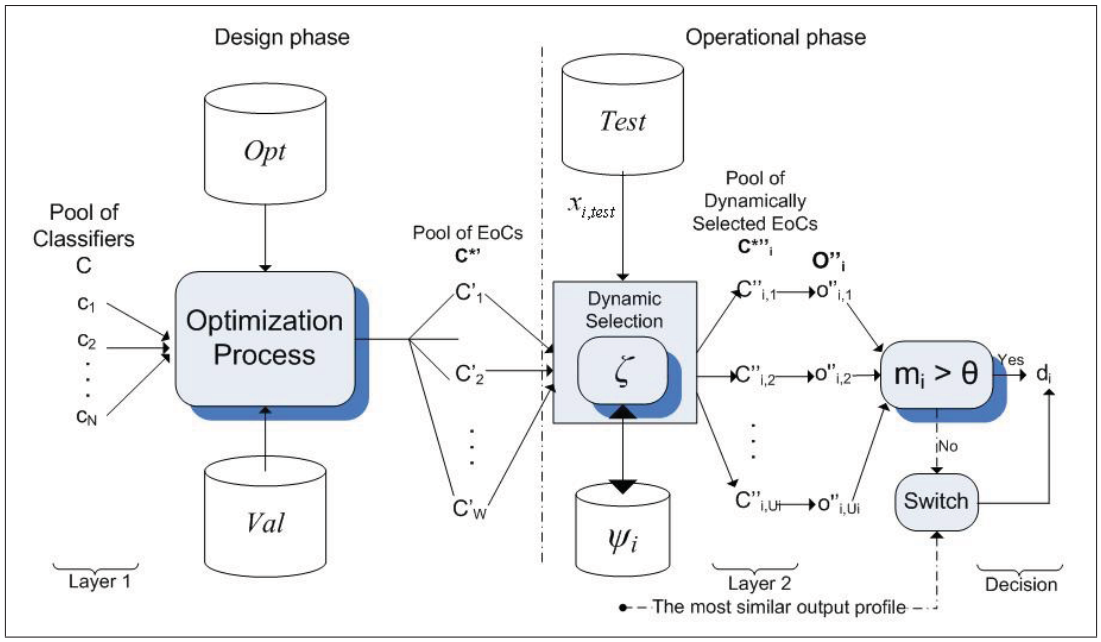



Figure 2.4 An overview of the DSA$^c$ approach. This method uses the knowledge provided by *Val* (converted into the set of output profiles *Val'*)

In the next section, we present the second method proposed in this work, whose main goal is to replace these dynamic selection functions by a context-based approach.

### 2.4.2 DSA$^c$: enhancing dynamic selection by using contextual information

Both DSA and DSA$^m$ dynamically select EoCs by considering dynamic selection functions based on the extent of consensus. Despite that the extent of consensus is a well studied concept in the literature (Hansen et al., 1997), only the outputs of the most voted and the second most voted classes are used to select the ensemble. However, the information related to the other classes is wasted, even though such information could help this task. In order to overcome this drawback, we propose DSA$^c$, which is depicted in Figure 2.4.

DSA$^c$ is inspired by both decision templates (Kuncheva et al., 2001) and the KNORA algorithm (Ko et al., 2008). The main objective is to use the validation database, transformed into output profiles, to point out which EoCs are the most competent to recognize the test sample $x_{i,test}$. An output profile is computed by transformation $T$ in Equation 2.7, where $x_i \in \Re^D$, $\tilde{x}_i \in \mathbb{Z}^{N+}$,

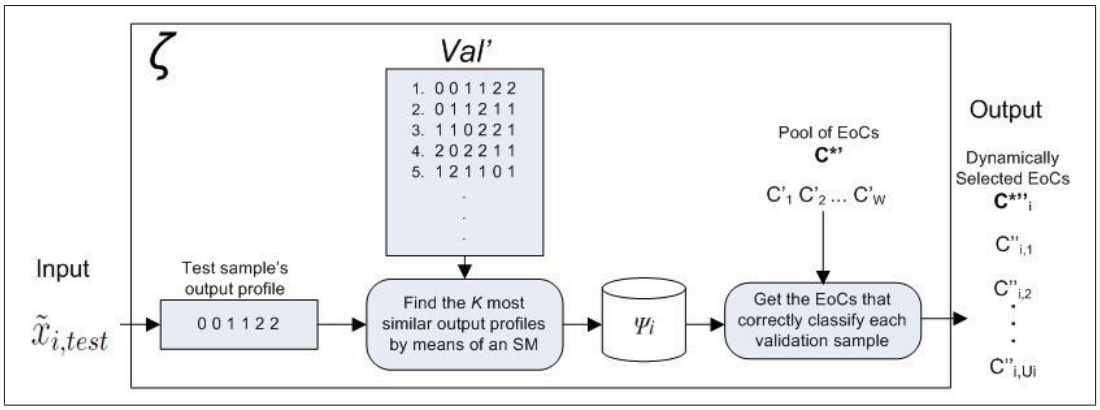



Figure 2.5 DSF $\zeta$. For each test sample, we find $K$ validation samples with the most similar output profiles, to form the set $\psi_i$. The EoCs that correctly classify the validation samples in $\Psi_i$ are used to compose the set $C^{*\prime\prime}$, which is then used to compute the final decision of DSA$^c$

and $N$ is the size of the pool of base classifiers $C$. Given that we know which EoC correctly recognizes each validation sample, a DMO structure is defined by computing which validation samples are the ones most similar to the test samples in considering the output profiles, and composing the dynamically selected set of EoCs with the EoCs that correctly classify these validation samples.

$$T : x_i \Rightarrow \tilde{x}_i, \tag{2.7}$$

In greater detail, this approach works as follows. Consider the pool of EoCs $C^{*\prime}$, generated during the design phase. For each test sample $x_{i,test}$, we compute the best set of EoCs ${C^{*\prime\prime}}_i$, composed of members from $C^{*\prime}$. Each EoC from $C^{*\prime}$ may appear several times in ${C^{*\prime\prime}}_i$, resulting in an automatic weighting approach. This task is achieved by considering the function $\zeta$, which is depicted in Figure 2.5.

Algorithm 2.2 describes this method in detail. The first few steps represent the function $\zeta$. First, in step 3 we apply $T$ on $x_{i,test}$, resulting in $\tilde{x}_{i,test}$. Next, as presented in step 4, we compare $\tilde{x}_{i,test}$ to each output profile in $Val'$, which is a database containing the output profiles of all validation samples in $Val$, e.g. $\tilde{x}_{j,val}\ \forall x_{j,val} \in Val$, computed in step 1. We compare these samples in terms of similarity, and save the degree of similarity between $\tilde{x}_{i,test}$ and $\tilde{x}_{j,val}$ in the variable $\delta_{i,j}$. Note that we use the similarity measure presented in Equation 2.8 to compute

Algorithm 2.2 $DSA^c$

1: Compute $Val'$ using transformation $T$ on all samples in $Val$
2: **for** each data point $x_{i,test}$ in $Test$ **do**
3: Compute $O_i = \{o_1, \ldots, o_N\}$ by considering $C = \{c_1, \ldots, c_N\}$, and use transformation $T$ to compute $\tilde{x}_{i,test}$
4: Find the $K$ $\tilde{x}_{j,val}$ most similar to $\tilde{x}_{i,test}$ and put into $\Psi_i$
5: $C^{*\prime\prime}_i = \emptyset$
6: **for** each $\tilde{x}_{j,val}$ in $\Psi_i$ **do**
7: **for** each $C'_k$ in $C^{*\prime}$ **do**
8: **if** $C'_k$ correctly recognizes $x_{j,val}$ **then**
9: Insert $C'_k$ into $C^{*\prime\prime}_i$ (re-insert another instance if $C'_k$ is already in the pool)
10: **end if**
11: **end for**
12: **end for**
13: Compute $m_i$ from $O''_i$
14: *# Switch mechanism*
15: **if** $m_i > \theta$ **then**
16: $d_i =$ most voted class from $O''_i$
17: **else**
18: $d_i =$ the label of the most similar $\tilde{x}_{j,val}$ from $\Psi_i$
19: **end if**
20: **end for**

$\delta_{i,j}$. The $K$ most similar output profiles $\tilde{x}_{j,val}$, e.g. the validation samples related to the highest values of $\delta_{i,j}$, are stored in $\Psi_i$. Next, as shown in steps 7 to 11, for each sample in $\Psi_i$ and each member of the pool of EoCs $C^{*\prime}$, we compute if the EoC provides the correct recognition result for this sample. In the case of a positive answer, this EoC is included in $C^{*\prime\prime}{}_i$, worth noting that an EoC appears in $C^{*\prime\prime}{}_i$ as many times as the number of samples that it correctly recognizes. Finally, $C^{*\prime\prime}{}_i$ is submitted to the switch mechanism $DSA^c$.

Steps 15 to 19 in Algorithm 2.2 represent the switch module in Figure 2.4, which corresponds to the previously mentioned switch mechanism. Similar to $DSA^m$, it is computed whether the margin $m_i$, in considering the dynamically selected EoCs $C^{*\prime\prime}{}_i$, is above the threshold $\theta$ or not. If $m_i > \theta$, then we use the most voted class indicated by $C^{*\prime\prime}{}_i$ (step 16). Otherwise, we use the

label of the most similar validation sample from $\Psi_i$ (step 18). The main goal of this scheme is to use contextual information also in the switch mechanism to avoid random decisions.

In order to compute the similarity of output profiles to perform step 4 in Algorithm 2.2, we use the template matching measure. In considering two output profiles, this measure computes how many classifiers will provide exactly the same output. We can implement this measure by maximizing Equation 2.8, which depends on Equation 2.9.

$$\delta_{i,j} = \frac{\sum_{k=1}^{N} \alpha_{i,j,k}}{N} \tag{2.8}$$

$$\alpha_{i,j,k} = \begin{cases} 1, & \text{if } \tilde{x}_{i,test,k} = \tilde{x}_{j,val,k} \\ 0, & \text{otherwise} \end{cases} \tag{2.9}$$

**DSA$^c$ for Incremental Learning**

One by-product of this approach is the ability to adapt to knowledge acquired over time. Such a task is realized by simply adding more data to *Val*, and computing the corresponding output profiles for $Val'$. In this case, we can conduct incremental learning without the need to change the parameters of the base classifiers. As a consequence, this system can be used with virtually any type of base classifier.

The computation time of the operational phase of DSA$^c$, however, depends heavily on the size of *Val*. Also, the application of this approach in an incremental scenario can slow down very significantly the operational phase since the larger the size of *Val*, the slower is the recognition module. Nevertheless, if we control the inclusion of new samples in *Val* by only injecting those that provide really useful information, we might reduce very significantly the increase of complexity resulting from incremental learning. For this reason, we present a control mechanism to avoid continuously appending new samples to *Val* during incremental learning. This mechanism works as follows.

The control mechanism selects samples, to compose *Val*, only when they are below a threshold $\vartheta$, in considering the margin of the base classifiers, e.g. $m_i < \vartheta$. Note that $m_i$ is defined in Equation 2.1. In this case, we suppose that only the samples that possess uncommon output profiles are appended to *Val*, since the contrary is likely to result in the addition of redundant samples. As a consequence, *Val* will only acquire new samples if uncommon samples are observed.

## 2.5 Experiments

In this section we present a series of experiments with the following objectives. First, the main goal is to compare the performance of the proposed approaches, i.e. DSA$^m$ and DSA$^c$, against existing methods. By comparing them against DSA, which provides the baseline architecture for the proposed methods, we aim at observing the impact of the proposed enhancements. By conducting the same comparisons against state-of-the-art static methods, on the other hand, we can observe the advantages of dynamic methods over static ones.

The aforementioned static methods are the followings:

- *All features*: the original classifier with full representation space (all original features).
- *Best from C*: the best base classifier from *C*.
- *MV all C*: fusion of all base classifiers in *C* by majority voting (MV).
- *DT all C*: fusion of all base classifiers in *C* using decision templates (DT), by considering template matching. The decision templates are computed by using *Val*$'$.
- *Best from C*$^{*\prime}$: the best EoC from $C^{*\prime}$.

All methods are evaluated using seven datasets, divided into two large and five small ones. The small datasets represent problems with a different number of features, generally with a small amount of samples. The datasets considered as small are: the DNA and Satimage datasets provided by Project Stalog on www.niaad.liacc.up.pt/old/stalog; Feltwell dataset, which is a

Table 2.1 Experimental setup. (NC: number of classes; *Train*, *Opt*, *Val*, and *Test*: number of samples in these respective sets; NF: number of features; NE: number of features in the ensemble, after applying the RSS method; VM: validation method; KF: k-fold validation; HO: hold-out validation). Each dataset of the methods using KF had ten different re-samplings, with no overlapping among the sets

| **Problem** | **NC** | *Train* | *Opt* | *Val* | *Test* | **NF** | **NE** | **VM** |
|---|---|---|---|---|---|---|---|---|
| DNA | 3 | 2,232 | 318 | 318 | 318 | 180 | 45 | KF |
| Feltwell | 5 | 7,662 | 1,094 | 1,094 | 1,094 | 15 | 8 | KF |
| Satimage | 6 | 4,506 | 643 | 643 | 643 | 36 | 18 | KF |
| Ship | 8 | 1,780 | 255 | 255 | 255 | 11 | 6 | KF |
| Texture | 11 | 3,850 | 550 | 550 | 550 | 40 | 20 | KF |
| Digits | 10 | 5,000 | 10,000 | 10,000 | *t1* 60,089<br>*t2* 58,646 | 132 | 32 | HO |
| Letters | 26 | 43,160 | 3,980 | 7,960 | 12,092 | 132 | 32 | HO |

multisensor remote-sensing dataset (Serpico et al., 1996); Ship, which is composed of forward-looking infra-red ship images (Park and Sklansky, 1990); and Texture, available in the UCI Machine Learning Repository. These databases, due to their sizes, are divided into ten folds, each time seven folds are used for training, one for optimization, one for validation, and the other one for testing. This process is repeated ten times for each replication, whereas each time a different set of samples was used.

The large datasets represent two handwriting recognition problems, e.g. the recognition of isolated digits and uppercase letters, extracted from the NIST-SD19 database. Two different test sets are used to evaluate digit recognition: *NIST-digits-test1* and *NIST-digits-test2*. For both digits and letters, the original feature set is composed of 132 features, extracted from concavities and contours (Oliveira et al., 2002). Table 2.1 presents a detailed description of each database.

Given the large amount of training samples available in the *NIST-digits* database (in addition to the training samples described in Table 2.1, there are 185,000 additional training samples), and the use of a well studied feature set, we can reduce the size of the training set to increase the level of uncertainty of the recognition problem, and simulate different conditions of uncertainty

(or confusion, which is a term used interchangeably with uncertainty hereafter). Consequently, this database does not only allow for simulating an incremental learning scenario, but also for evaluating how an approach can behave at different degrees of confusion. For this reason, in this section, we also aim at answering the following questions:

A. How can DSA$^c$ behave in an incremental learning scenario, by just appending new samples to *Val*?

B. How dynamic selection, represented by DSA$^c$, performs against static selection when the size of *Val* ranges from small (high level of uncertainty) to high (low level of uncertainty)?

For all experiments, the following parameters were considered. For each dataset, 100 base classifiers, with a pre-defined number of features, are generated from the baseline feature set, based on the random subspaces (RSS) ensemble generation method (Ho, 1998). The base classifiers can be considered weak classifiers in two aspects. First, the two different types of classifiers, e.g. k-nearest neighbors classifiers with $k = 1$ (1NN), and C4.5 decision tree (DTree) classifiers, can be considered very weak for many problems. Second, the reduced number of features used by the RSS method (see Table 2.1 for the number of features used for each problem) greatly contributes to weaken the performance of the classifiers.

To generate the pool of EoCs, a genetic algorithm (GA) is used, in an off-line fashion, to find an archive with the 25 best solutions on *Val*, representing $C^{*\prime}$, guided by the optimization set *Opt*. The following parameters were used in this work: population size: 128; number of generations: 1,000; probability of crossover: 0.8; probability of mutation: 0.01; one-point crossover and bit-flip mutation (Dos Santos et al., 2008). The experiments are replicated 30 times, where in each replication the archive provided by GA is generally different. The results represent the mean error rates over the 30 replications. For each of the sets using k-fold validation, each replication represents the mean over the ten re-samplings of each dataset.

For the large datasets, we also evaluated the method known as Bagging to generate the base classifiers. We used the same scheme employed in (Dos Santos et al., 2008), where 100 DTree

classifiers were generated by dividing the training set into 100 subsets of equal size, where the samples for each set were randomly chosen, with no overlapping among the sets. DTree is used as the base classifier given that Bagging works better with unstable classifiers.

The results are statistically validated by the Kruskal-Wallis nonparametric statistical test. We test the equality among the mean values, using a confidence level of 95%. Dunn-Sidak correction is applied to critical values.

### 2.5.1 Results and discussion

The results from the evaluation of small datasets are presented in Tables 2.2 and 2.3, for 1NN and DTrees, respectively. Results from the evaluation of large datasets are presented in Table 2.4. In all tables, we present only the error rates of DSA$^c$ with $K = 30$. The impact of $K$ will described later.

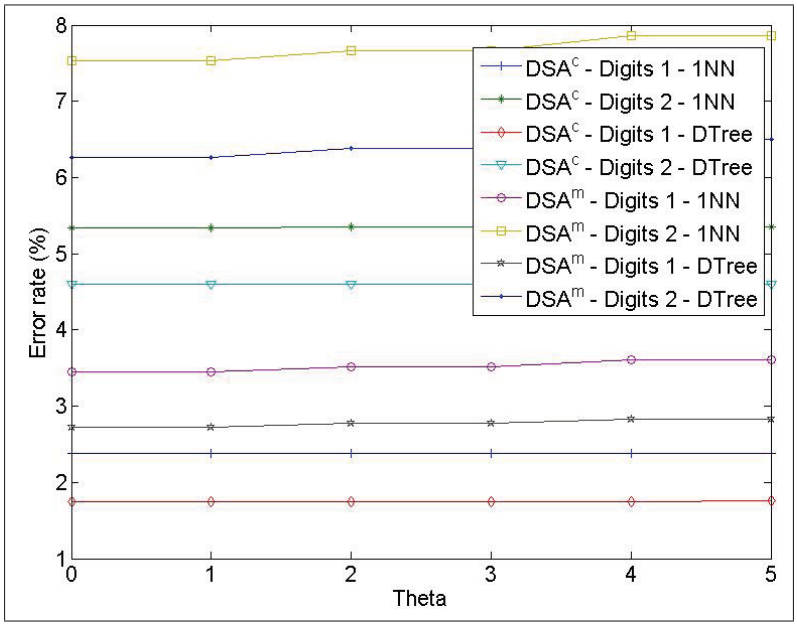



Figure 2.6 Evaluation of the parameter $\theta$ for the switch mechanism

For both DSA$^m$ and DSA$^c$, $\theta = 0$, since this was the best value after preliminary evaluations as shown in Figure 2.6. As demonstrated, the switch works very well as a tie-breaking mechanism

Table 2.2 Error rates on small datasets using 1NN classifiers. Results in bold present the best approach among static MO, DSA, DT, and the proposed DSA$^m$ and DSA$^c$, with $K$ set to 30. Underlined results represent the statistically-significant best method. Highlighted by * are the proposed approaches. Between parentheses is the standard deviation of each approach ($\times 10^{-2}$)

| **Method** | **Dna** | **Felt** | **Sat** | **Ship** | **Text** |
|---|---|---|---|---|---|
| *Static selection* | | | | | |
| Oracle $C$ | 0.03 (-) | 0.67 (-) | 0.36 (-) | 0.28 (-) | 0.04 (-) |
| All features | 26.30 (-) | 12.35 (-) | 9.84 (-) | 11.24 (-) | 1.13 (-) |
| Best from $C$ | 23.10 (-) | 9.46 (-) | 8.95 (-) | 10.26 (-) | 0.62 (-) |
| MV all $C$ | 6.87 (-) | 10.44 (-) | 8.59 (-) | 9.94 (-) | 1.11 (-) |
| Best from $C^{*'}$ | 9.14 (1.60) | 9.37 (2.09) | 8.19 (2.89) | 9.41 (1.66) | 0.71 (1.74) |
| DT $C$ | 8.53 (-) | 14.76 (-) | 8.97 (-) | 10.03 (-) | 4.56 (-) |
| *Dynamic selection* | | | | | |
| Oracle $C^{*'}$ | 1.12 (0.86) | 6.06 (2.46) | 3.98 (0.83) | 3.92 (1.40) | 0.40 (0.24) |
| DSA | 10.47 (3.17) | 10.76 (4.90) | 9.17 (0.99) | 11.21 (3.48) | 1.03 (0.34) |
| $*$DSA$^m$ | 5.57 (1.33) | 9.35 (4.62) | 7.61 (0.87) | 8.80 (2.30) | 0.93 (0.37) |
| $*$DSA$^c$ | **5.46** (0.26) | **8.93** (0.30) | **7.42** (0.31) | **8.10** (0.38) | **0.56** (0.10) |

for both approaches. It is worth noting that when we increase the value of $\theta$, the final error rates also increase. This fact suggests that by relying more on the decisions provided by the main structure of either DSA$^m$ or DSA$^c$ (note the higher the value of $\theta$, the more often the switch is used), and only using the base classifiers when a tie occurs, the final approach is more reliable.

The error rates resulting from the evaluation of small databases show that both DSA$^m$ and DSA$^c$ are very promising for problems presenting a high level of confusion. The only database for which neither of the proposed methods resulted in the lowest error rates was the Feltwell database, using DTree as base classifiers. On all the other databases, DSA$^c$ achieved the lowest recognition rates.

For the large databases, DSA$^c$ yielded the lowest error rates on all databases. DSA$^m$, in contrast, has performed poorly compared to other static methods. Note that DSA$^c$ uses the validation dataset to compute the DMO structure, and given the larger amount of training samples

Table 2.3 Error rates on small datasets using DTree classifiers. Results in bold present the best approach among static MO, DSA, DT, and the proposed DSA$^m$ and DSA$^c$, with $K$ set to 30. Underlined results represent the statistically-significant best method. Highlighted by * are the proposed approaches. Between parentheses is the standard deviation of each approach ($\times 10^{-2}$)

| **Method** | **Dna** | **Felt** | **Sat** | **Ship** | **Text** |
|---|---|---|---|---|---|
| *Static selection* | | | | | |
| Oracle $C$ | 0.03 (-) | 0.60 (-) | 0.22 (-) | 0.24 (-) | 0.02 (-) |
| All features | 6.85 (-) | 16.81 (-) | 14.17 (-) | 10.92 (-) | 7.56 (-) |
| Best from $C$ | 11.33 (-) | 11.86 (-) | 11.83 (-) | 10.45 (-) | 6.07 (-) |
| MV all $C$ | 5.05 (-) | 11.86 (-) | 8.64 (-) | 6.80 (-) | 2.56 (-) |
| Best from $C^{*'}$ | 5.71 (1.30) | **10.22** (2.11) | 8.35 (1.01) | 7.02 (1.59) | 2.04 (2.60) |
| DT $C$ | 4.53 (-) | 13.93 (-) | 8.96 (-) | 7.74 (-) | 1.34 (-) |
| *Dynamic selection* | | | | | |
| Oracle $C^{*'}$ | 1.07 (0.92) | 5.82 (3.94) | 3.78 (0.78) | 3.18 (1.76) | 0.81 (0.24) |
| DSA | 7.55 (2.47) | 12.52 (5.28) | 10.29 (2.16) | 10.16 (4.38) | 2.42 (0.82) |
| $*$DSA$^m$ | 4.07 (1.07) | 10.77 (4.82) | 7.42 (0.76) | 5.89 (1.82) | 2.13 (0.78) |
| $*$DSA$^c$ | **3.05** (0.34) | 10.32 (0.41) | **7.11** (0.30) | **5.52** (0.45) | **1.11** (0.17) |

compared to the small datasets, we believe that this approach has been able to take better advantage of the lower level of uncertainty of large problems, so that it reaches the best performance in this evaluation. Given the better performance of DSA$^c$ over DSA$^m$, hereafter, we pursue the experimental evaluation by considering only the former for the sake of simplicity.

In order to provide a broader overview of the performance of DSA$^c$, we show the impact of the value of $K$, in a range between 1 and 30. Such an evaluation is presented in Figs. 2.7 and 2.8 for small problems, with 1NN and DTrees, respectively. In Figs. 2.9 and 2.10, we present the same evaluation in large problems, with 1NN and DTrees, respectively. We observe that the best value for this parameter is problem-dependent. Databases that generate higher error rates, such as Feltwell, require high $K$ values, and databases with very low error rates, such as Texture, require very low $K$ values. Consequently, even though by setting $K = 30$ DSA$^c$ is able to perform well, this value could be adapted to either improve performance or reduce complexity.

Table 2.4 The same evaluations in error rates as in Table 2.2, but considering both 1NN and DTrees with large datasets. The standard deviation in this case was multiplied by $10^{-3}$. In addition, we present the evaluation of DTree classifiers created by bagging

| **Classifier** | **1NN - RSS** | | | **DTree - RSS** | | | **DTree - Bagging** | | |
|---|---|---|---|---|---|---|---|---|---|
| **Method** | **Digits** | | **Letters** | **Digits** | | **Letters** | **Digits** | | **Letters** |
| | *test1* | *test2* | | *test1* | *test2* | | *test1* | *test2* | |
| *Static selection* | | | | | | | | | |
| Oracle $C$ | 0.05 (-) | 0.17 (-) | 0.18 (-) | 0.01 (-) | 0.04 (-) | 0.04 (-) | 0.24 (-) | 0.63 (-) | 0.29 (-) |
| All features | 6.66 (-) | 9.76 (-) | 7.82 (-) | 11.07 (-) | 18.20 (-) | 13.50 (-) | 6.66 (-) | 9.76 (-) | 7.82 (-) |
| Best from $C$ | 7.52 (-) | 13.99 (-) | 14.47 (-) | 10.30 (-) | 19.18 (-) | 17.13 (-) | 9.70 (-) | 16.62 (-) | 14.31 (-) |
| MV all $C$ | 3.72 (-) | 8.10 (-) | 6.60 (-) | 2.92 (-) | 6.67 (-) | 6.06 (-) | 5.65 (-) | 10.99 (-) | 7.63 (-) |
| Best from $C^{*^l}$ | 3.60 (1.95) | 7.77 (2.78) | 6.56 (2.59) | 2.98 (2.23) | 6.77 (1.05) | 6.21 (2.79) | 5.31 (0.06) | 10.28 (0.03) | 7.62 (0.02) |
| DT $C$ | 2.55 (-) | 5.74 (-) | 4.95 (-) | 2.00 (-) | 5.00 (-) | 4.64 (-) | 3.65 (-) | 7.65 (-) | 6.49 (-) |
| *Dynamic selection* | | | | | | | | | |
| Oracle $C^{*^l}$ | 1.97 (0.14) | 4.59 (0.37) | 3.87 (2.10) | 1.87 (1.01) | 4.39 (2.08) | 4.53 (1.57) | 3.72 (0.04) | 7.42 (0.02) | 4.68 (0.01) |
| DSA | 3.61 (0.28) | 7.87 (0.41) | 6.43 (0.69) | 2.87 (0.24) | 6.61 (0.54) | 6.06 (0.64) | 5.33 (0.05) | 10.45 (0.06) | 7.11 (0.03) |
| $*\text{DSA}^m$ | 3.45 (0.22) | 7.53 (0.40) | 6.12 (0.66) | 2.72 (0.42) | 6.26 (0.76) | 5.83 (0.61) | 5.10 (0.08) | 9.96 (0.05) | 7.23 (0.04) |
| $*\text{DSA}^c$ | **2.37** (0.14) | **5.34** (0.21) | **4.62** (0.41) | **1.76** (0.14) | **4.36** (0.20) | **4.20** (0.22) | **2.98** (0.04) | **6.17** (0.03) | **5.58** (0.05) |

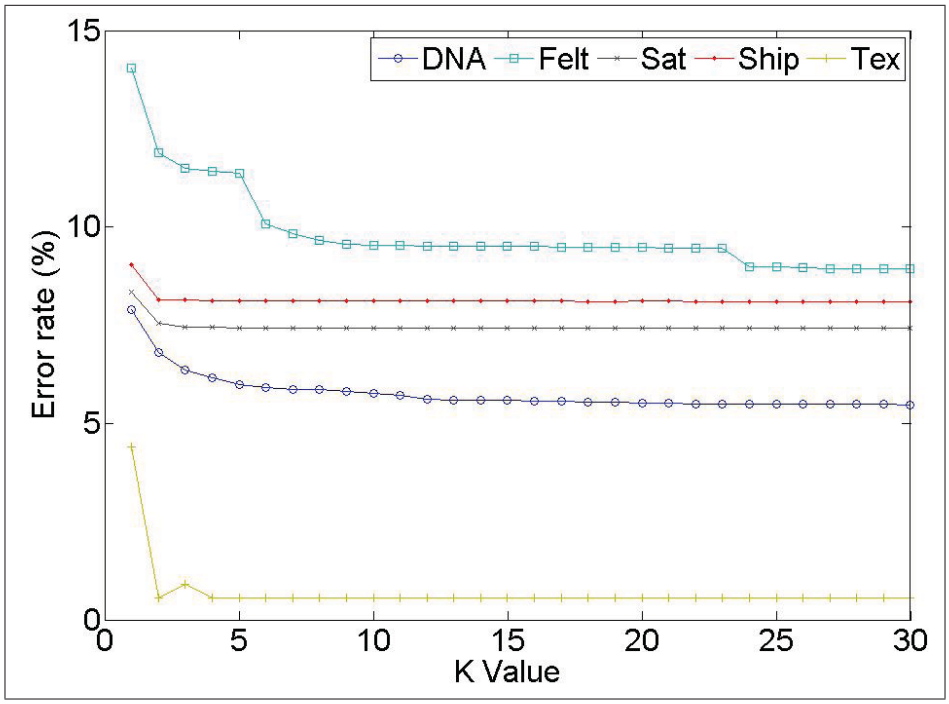



Figure 2.7 Evaluation of DSA$^c$ on small datasets with 1NN classifiers, $K$ varying from 1 to 30

Even though the main goal of this paper was to improve the performance of fusion functions, in Table 2.5 we present a summary of the results presented by DSA$^c$ against the best results reported in the literature for the same databases evaluated in this work. This table can provide us an idea to what level of performance a multiple classifier system, using weak classifiers, can attain by using a very robust combination approach.

In considering small databases, DSA$^c$ has been able to outperform the best results thus far published in the literature, on all databases. It is worth noting that none of the methods presented in Table 2.5 uses exactly the same experimental protocol, so this comparison is not as accurate as for large databases. However, the use of data from the same database provides a good idea on the difference in performance among the different methods.

For large databases, the error rates presented by DSA$^c$ are slightly higher than the lowest error rates reported in the literature. However, the best results so far have been achieved by

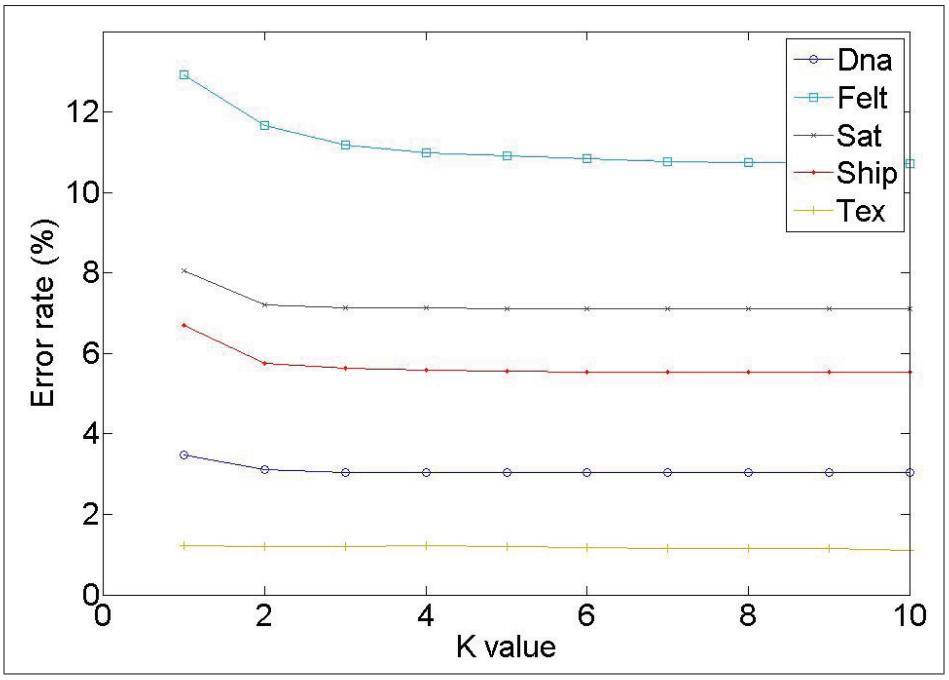



Figure 2.8 Evaluation of DSA$^c$ on small datasets with DTree classifiers, $K$ varying from 1 to 10

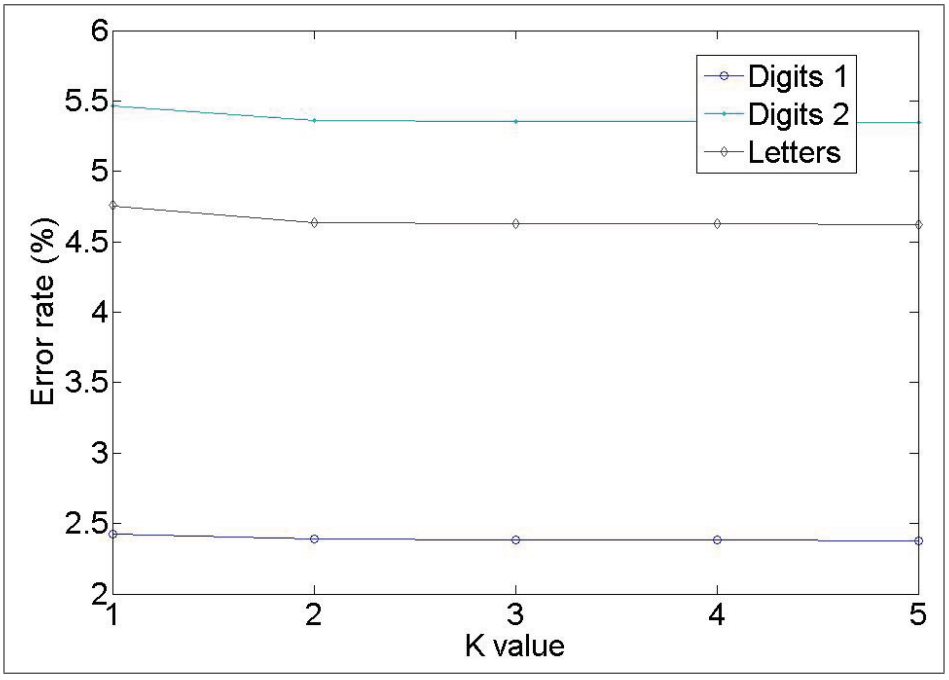



Figure 2.9 Evaluation of DSA$^c$ on large datasets with 1NN classifiers, $K$ varying from 1 to 5

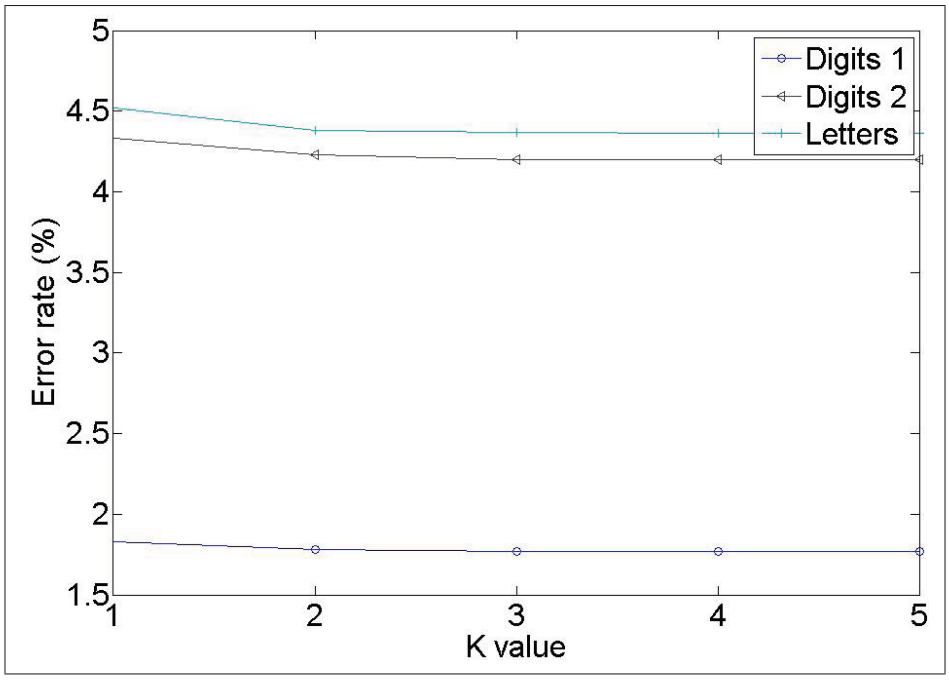



Figure 2.10 Evaluation of DSA$^c$ on large datasets with DTree classifiers, $K$ varying from 1 to 5

Table 2.5 Error analysis, in which we compare the results of the proposed method DSA$^c$ with the best results published in the literature. The second column represents the average over 30 replications

| | **Proposed method** | | **Literature** | |
|---|---|---|---|---|
| **Database** | **Average (Variance)** | **Best result** | **Method** | **Result** |
| DNA | 3.05 (0.12) | 2.88 | EoC+DS (Dos Santos et al., 2008) | 4.59 |
| Feltwell | 8.85 (0.12) | 8.72 | EoC+DS (Dos Santos et al., 2008) | 11.50 |
| Satimage | 6.89 (0.11) | 6.78 | EoC+DS (Dos Santos et al., 2008) | 8.64 |
| Ship | 5.51 (0.27) | 5.32 | EoC (Rheaume et al., 2002) | 5.68 |
| Texture | 0.56 (0.01) | 0.52 | EoC+DS (Woods et al., 1997) | 0.66 |
| NIST-digits-test1 | 1.76 (0.02) | 1.08 | Single Classifier (Milgram et al., 2006) | 0.63 |
| NIST-digits-test2 | 3.31 (0.04) | 3.28 | EoC+SS (Radtke, 2006) | 2.33 |
| NIST-letters | 3.89 (0.06) | 3.87 | SC (Milgram et al., 2006) | 3.18 |

using strong classifiers, such as Support Vector Machines (SVM) (Milgram et al., 2006) and Multilayer Perceptron (MLP) Neural Networks (Radtke, 2006), which generally deal very well with large training sets. In this paper, we limited the scope of the work to consider only weak classifiers and small training datasets in order to better observe the behavior of combination approach in conditions that might generate a high level of confusion for the base classifiers.

The results from the literature, in contrast, might have dealt with lower levels of confusion due to the much larger amount of samples used for training.

As a consequence, the remainder of this section aims at comparing the performance of DSA$^c$ against MLP and SVM, which are state-of-the-art static approaches, under the same conditions. First, we evaluate what level of performance DSA$^c$ can reach if we incrementally learn the information provided by the remaining training samples in the *NIST-digits* database. Next, we retrain MLPs and SVMs at different levels of uncertainty, which are achieved by downsizing the *NIST-digits* database, and compare their results against the ones produced by DSA$^c$.

#### 2.5.1.1 Evaluation of DSA$^c$ in an incremental learning scenario

In this section, we evaluate the impact of increasing the size of *Val* to improve the overall performance of DSA$^c$, by simulating an incremental scenario. Such a simulation consists of gradually adding new samples to *Val*, as previously discussed in Section 2.4.2. We take advantage of the large set of digits available in the NIST SD19 database, by increasing the size of *Val* from 10,000 to 180,000 samples. Those are the remaining samples in the hsf_{1-3} series of the database.

The results of these experiments are shown in Figure 2.11, considering both 1NN and DTrees with RSS, and both *NIST-digits-test1* and *NIST-digits-test2*. Note that these evaluations do not only aim at evaluating the behavior of the approach in the incremental scenario, but also aim at comparing the final results against the literature, since the best results thus far consider methods that used all samples from this database. As a consequence, in the following paragraphs we discuss the first topic, while the second topic is discussed afterwards.

Generally, the impact of the size of *Val* is more significant when the size of *Val* is relatively small, and it tends to gradually converge with larger validation sets. Nevertheless, with any increase in *Val* we can observe some improvement. This fact shows that the approach can incrementally acquire knowledge by only increasing the size of this set, so that it can be a

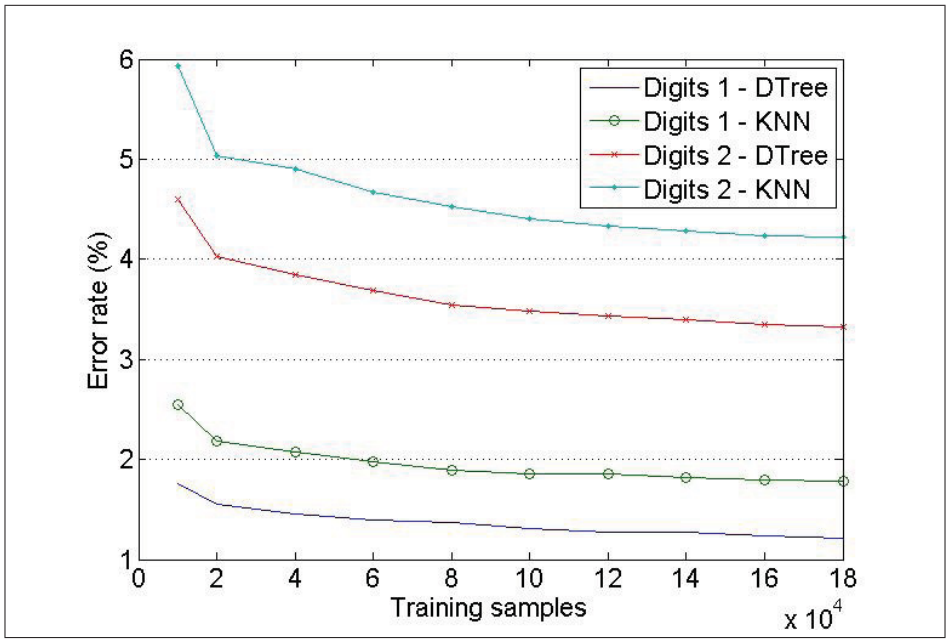



Figure 2.11 Incremental evaluation of DSA$^c$, with $K = 30$, using validation set sizes from 10,000 to 180,000, on both *NIST-digits-test1* and *NIST-digits-test2*

generic approach for incremental learning. This allows us to use a heterogeneous pool of classifiers in the incremental learning process.

Figure 2.12 plots the results of the evaluation of different values for $\vartheta$ using the control mechanism described in Section 2.4.2. Compared to the performance of the system using all 180,000, we see that the control mechanism is able not only to maintain the performance of the system, but also to reduce the final error rates. With $\vartheta = 40$, the final error rates are reduced to about 1.1%. In addition, we demonstrate in Figure 2.13 this mechanism on the size of *Val*. The best approach, represented by $\vartheta = 40$, used only 25,948 samples in *Val*. Comparing with the use of all 180,000 samples, we can reach better results by using only around 15% of this set and drastically reduce the search space of DSA$^c$ for recognition.

The final results can be summarized as follows. With 1NN, the error rates have been reduced from about 2.55% to about 1.78% on *NIST-digits-test1*, and from about 5.9% to about 4.2% on *NIST-digits-test2*. With DTrees, the error rates decreased from about 1.75% to about 1.1%

on *NIST-digits-test1*, and from about 4.6% to about 3.31% on *NIST-digits-test2*. Note that on *NIST-digits-test1*, the best results reported in the literature are around 0.63% (Milgram et al., 2006), using 132 features, 195,000 samples for training, and MLP as classifier. In this work we could get very close (only 0.47% below) to these results by using weak classifiers, trained with only 10,000 samples, of which the range of individual error rates is, for example with 1NN, between 15.92% and 7.53%. Even though in the end we have used the same number of samples to get these results, we have shown that our approach is able to improve weak classifiers to a level which is comparable to the best classification methods in the literature, without changing their parameters.

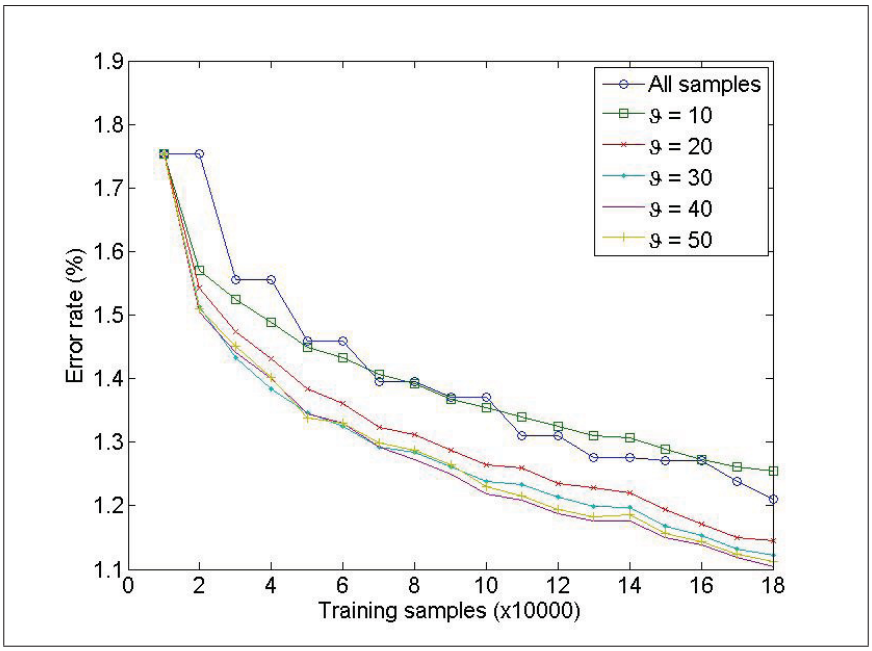



Figure 2.12 Incremental evaluation of DSA$^c$ ($K = 30$) with DTree classifiers on *NIST-digits-test1* using a control mechanism

### 2.5.1.2 Evaluation of DSA$^c$ against MLP and SVM at varied conditions

As demonstrated in the previous section, by using all the training samples provided by *NIST-digits*, DSA$^c$ can attain a level of performance that is close to state-of-the-art classifiers such as MLP and SVM, consisting of static approaches. However, the higher complexity of DSA$^c$,

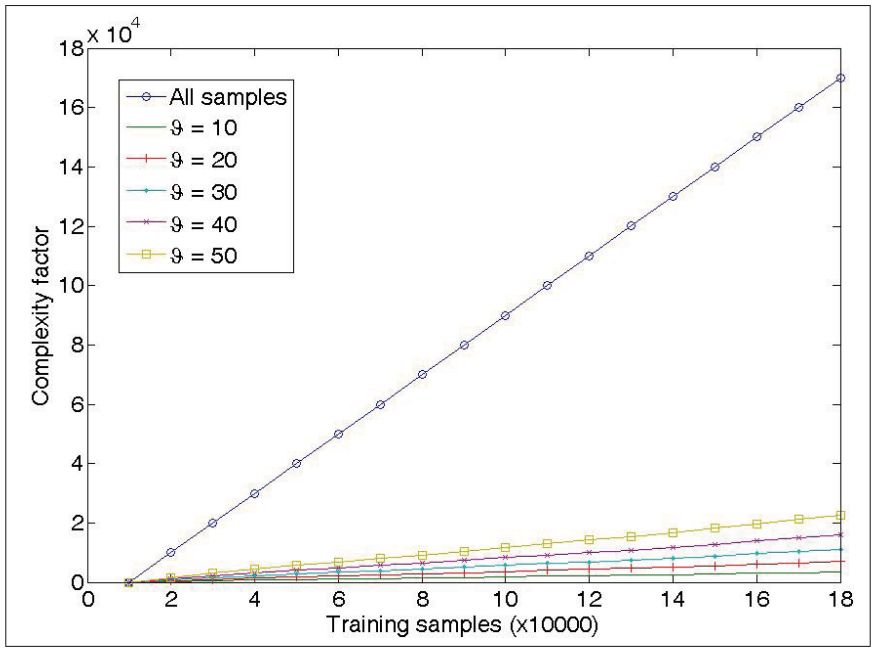



Figure 2.13 Size of the validation set for the evaluation presented in Figure 2.12

in both the design and operational phases, might be a barrier for its application in the real world. For this reason, the main goal of this section is to compare the proposed method, which is a dynamic approach, against MLP and SVM, which are static approaches, under various conditions created by downsizing the *NIST-digits* database. The idea is to observe under which condition dynamic selection might be worth the higher complexity. As we previously mentioned, such a downsizing allows for increasing the level of uncertainty of the problem by simply reducing its training set, since the empirical lower-bound of the *NIST-digits* database is known.

By using a setup similar to that described in the previous section, the training database was reduced to these sizes: 5,000, 10,000, 15,000, 20,000, and 25,000. However, for each training set, we did 15 different resamplings so that we could conduct 15 replications for each size of the training set. The parameters for both SVM and MLP were set to the same as reported in (Milgram, 2007), which were found as the best parameters for this database. Note that for DSA$^c$ we conduct the incremental learning of *Val*. For MLP and SVM, in contrast, batch learning is considered, since for each training set size, we retrain the classifiers. In addition, it

is worth noting that *Val* and *Opt* are merged together to define a single set of samples, which is used as hold-out validation set by MLP and SVM.

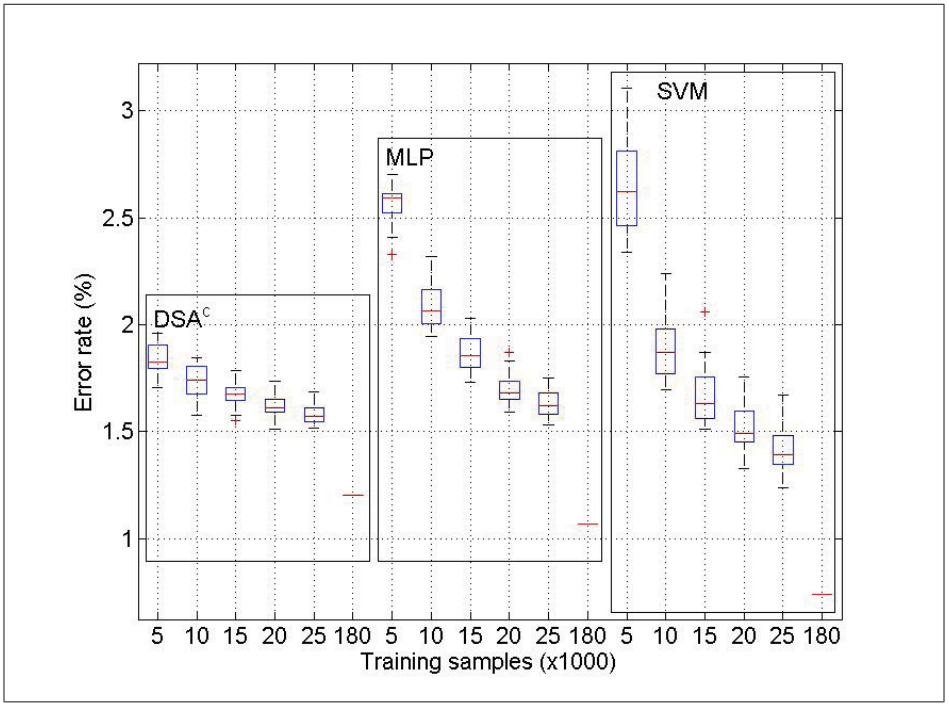



Figure 2.14 Evaluation of different sizes of the training set for NIST-digits, using *NIST-digits-test1*. These experiments were replicated 15 times by resampling the training set each time (a single replication for 180,000 samples, which corresponds to the entire dataset). Note that the experiments are grouped by approach, e.g. DSA$^c$, MLP, and SVM, respectively, and for each approach, we evaluated training sets with 5,000, 10,000, 15,000, 20,000, 25,000, and 180,000 samples, respectively

The main results are presented in Figure 2.14, for *NIST-digits-test1*, and Figure 2.15 *NIST-digits-test2*. The most remarkable observation lies in the experiments using only 5,000 samples for training. In this case, DSA$^c$ was significantly superior to both MLP and SVM, showing that the proposed approach can deal better with a high level of uncertainty under these conditions. However, this gap becomes narrower and narrower as we increase the size of the training set, e.g. when we decrease the level of confusion. As a result, the main observation from these experiments is that dynamic selection, despite generally presenting higher complexity than static selection, may be the most recommended approach to attain high performance when the level of confusion of the recognition problem is high. When the level of confusion is low, on

the other hand, a static approach may work very well without all the complexity brought by dynamic selection.

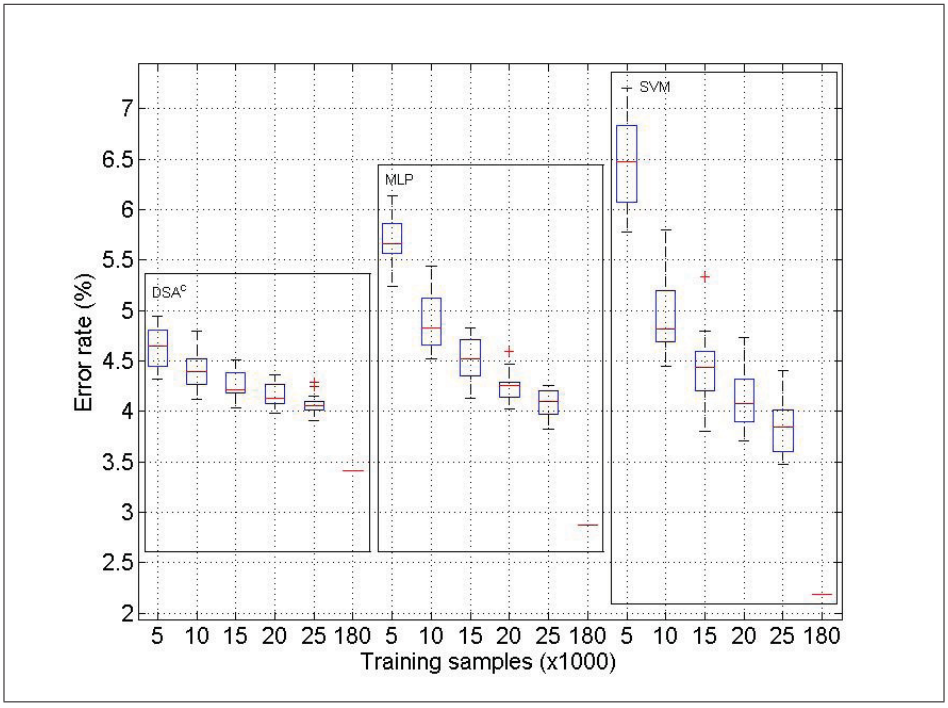



Figure 2.15 The same evaluations as in Figure 2.14, but using *NIST-digits-test2*

## 2.6 Conclusion and future work

In this paper we first proposed dynamic multistage organizations to enhance classifier fusion. Based on Dos Santos et al's approach, we first implemented DSA$^m$ to validate these concepts by using multiple dynamic selection functions. Next, we extended DSA$^m$ to use the knowledge provided by the output profiles of validation samples to create DMO, resulting in DSA$^c$.

Experiments conducted on both small and large databases have confirmed that the proposed DMO concept looks really promising in improving the use of multiple classifiers, since the proposed enhancements have been effective in improving DSA. We also observed a significant improvement in performance of DSA$^c$ over DSA$^m$, due to the use of contextual information. The use of simulated incremental learning scenario showed that we can improve the performance of DSA$^c$ by only increasing the size of the validation set, without changing the parame-

ters of the base classifiers. Although other classification approaches such as SVMs and MLPs can present better performances than DSA$^c$ when large training sets are available, we demonstrated that the proposed approach results in better performance when one can use only small training databases, e.g.when the level of confusion for recognition is high.

As future work, many directions can be followed. The most important, in our opinion, is to better investigate the observation that DSA$^c$ is better suited to problems presenting a high level of uncertainty. We can evaluate, for example, the current system on other recognition problems. We can, as well, implement the system with other base classifiers and different methods to generate the pool of base classifiers, to evaluate whether the system maintains the same behavior with a different baseline architecture or not. In addition, reducing the complexity of DSA$^c$ is a key point to better justify its deployment in real-life systems. In this work we simply performed a flat search on *Val*, but other more time-efficient methods can be investigated, for instance some ideas proposed to reduce the complexity of 1NN classifiers (Cui et al., 2003) to conduct the search for the most similar samples.

## 2.7 Discussion

In this chapter, we present DS approaches that can be used with various types of classifiers, including HMM-based classifiers. This is possible because only the outputs yielded by the members of the pool *C* are used to conduct the DS task. Consequently, promising ideas have been presented for the definition of adaptive systems, which could be used in conjunction with the ideas presented in Chapter 1.

In some ways, DSA$^c$ can be viewed as an AS. It can perform adaptation during the generalization phase by means of DS, and it can adapt the classifiers during learning by adding new samples to its validation set. Nevertheless, the pool of base classifiers remains static during the lifetime of the system as a whole. An AS should fully adapt to new sources of knowledge. To achieve this task, we believe that the base classifiers need to be updated over time as well.

In the next chapter, we present a framework designed to overcome the above issues and define a robust AS. For this, we pursue the work presented in this chapter by including solutions to better adapt the system in the learning phase. We also present ways to improve the generalization level, in order to avoid reliance on hard-decisions made during the design phase.

# CHAPTER 3

# LOGID: AN ADAPTIVE FRAMEWORK COMBINING LOCAL AND GLOBAL INCREMENTAL LEARNING FOR DYNAMIC SELECTION OF ENSEMBLES OF HMMS

In this work, we propose the LoGID (**Lo**cal and **G**lobal **I**ncremental Learning for **D**ynamic Selection) framework, the main goal of which is to adapt hidden Markov model-based pattern recognition systems during both the generalization and learning phases. Given that the baseline system is composed of a pool of base classifiers, adaptation during generalization is performed through the dynamic selection of the members of this pool that best recognize each test sample. This is achieved by the proposed K-nearest output profiles algorithm, while adaptation during learning consists of gradually updating the knowledge embedded in the base classifiers, by processing previously unobserved data. This phase employs two types of incremental learning: local and global. Local incremental learning involves updating the pool of base classifiers by adding new members to this set. The new members are created with the Learn++ algorithm. Global incremental learning, in contrast, consists of updating the set of output profiles used during generalization. The proposed framework has been evaluated on a diversified set of databases. The results indicate that LoGID is promising. For most databases, the recognition rates achieved by the proposed method are higher than those achieved by other state-of-the-art approaches, such as batch learning. Furthermore, the simulated incremental learning setting demonstrates that LoGID can effectively improve the performance of systems created with small training sets as more data are observed over time.

## 3.1 Introduction

In the past, pattern recognition systems have relied extensively on off-line optimization to fine-tune classifier parameters. Such an approach usually requires a training set large enough to contain a number of different samples. These samples must represent most of the variability

to be observed during recognition, otherwise the system will yield poor generalization results. However, it may not always be possible to acquire such a training set off-line.

Moreover, without appropriate training data, classifier parameters might be poorly estimated, resulting in a great deal of uncertainty[1] during recognition. That is, the final recognition decision may be based on random guesses for some 'difficult' samples, i.e. samples that the current classification scheme cannot recognize with enough confidence. It might be possible to overcome this issue, however, if the classifiers incorporate new knowledge that becomes available over time. This knowledge is represented by the data that are processed during operation of the system. Presumably, the more data are observed, the better the estimate of the classifier parameters, and, consequently, the lower the degree of difficulty faced by these classifiers. On this basis, various incremental learning (IL) algorithms have been proposed (Polikar et al., 2001; Cavalin et al., 2009; Mongillo and Deneve, 2008).

The use of ensembles of classifiers (EoCs) (Rokach, 2010) in IL algorithms has been shown to be effective. New classifiers, trained on new data, can be appended to an existing pool to incorporate new knowledge without losing previous information (Polikar et al., 2001; Cavalin et al., 2009; Yu-Shu and Yi-Ming, 2009; Ulas et al., 2009; Kapp et al., 2010). That new knowledge is represented by the new classifiers, while the previous knowledge is embedded in the existing ones. Although this method is suitable for a broad range of systems, and can be therefore applied to different types of classifiers, most of these algorithms rely on static methods to combine classifiers during generalization. Static methods can be useful for dealing with some issues, such as the negative effect of using small datasets for training. However, other issues, such as high intra-class variability, call for a combination method that can select the best classifiers for recognizing each test sample.

As demonstrated in (Cavalin et al., 2011a), the use of EoCs in dynamic selection may provide better performance than static selection in settings involving a high level of uncertainty. The

[1]The terms *uncertain*, *difficult*, and *confused* are used, interchangeably, to refer to the same concept: the recognition problem is ill-defined, i.e. there are not enough data to model classifiers to deal with a large feature set, and these classifiers are likely to perform poorly during generalization.

main approach, called the Dos Santos et al approach with Contextual Information ($DSA^c$), can be incrementally updated by appending new samples to its validation set. However, the pool of classifiers remains static during this process, meaning that one module of the system is adapted to new sources of knowledge, but its main components, the classifiers, remain static. Where problems are ill-defined, for instance problems where there are not enough data for training, an approach is required that is not only able to control a baseline recognition system during generalization, but also to adapt the parameters of the system as new data are observed during learning.

To address this issue, we propose a new framework called LoGID (**Lo**cal and **G**lobal **I**ncremental Learning for **D**ynamic Selection), which integrates EoC-based incremental learning with a dynamic selection approach inspired by $DSA^c$. The framework is designed to adapt a pool of base classifiers to the data processed by the system at both the learning and generalization levels. During generalization, the main idea is to select the best classifiers for recognizing each test sample. During learning, the focus is to update the knowledge embedded in the classifiers, using the data that become available over time. Given the structure of LoGID's generalization phase, the learning phase involves two different types of incremental learning:

a. *Local*: incremental learning of the pool of base classifiers;

b. *Global*: updating of the parameters of the dynamic selection algorithm based on newly-observed data.

LoGID consists of the following components. For the generalization phase, we propose a new mechanism for dynamic selection: K-nearest Output Profiles (KNOP), which combines the completely dynamic architecture of the KNORA algorithm (Ko et al., 2008) and the more general architecture of $DSA^c$ (more general because of its use of output profiles[2]). Local incremental learning uses the Learn++ algorithm (Polikar et al., 2001) to incrementally generate a diverse pool of classifiers. Given the KNOP architecture, global incremental learning is re-

[2] An output profile consists of a vector containing the outputs yielded by the base classifiers (Kuncheva et al., 2001).

alized by appending new samples to the dynamic selection dataset. It is worth noting that we focus in this work on optimizing LoGID for classifiers based on hidden Markov models (HMMs). This allows us to pursue the evaluations using the ensembles presented in (Cavalin et al., 2009), and to observe the possible boost from using the proposed approach in incremental learning settings. However, LoGID can be adapted to other types of classifiers with minor modifications.

The proposed method is evaluated on a varied set of databases, involving problems such as handwriting recognition, speech recognition, and speaker identification. These databases vary greatly in terms of the numbers of input features, classes, and training samples, which allow us to evaluate the proposed approach on different types of HMM-related recognition problems, each of which presents a different level of difficulty. During the evaluations, an incremental learning scenario is simulated. The goal is to observe how the proposed method evolves as new data are observed, and to observe its effect on the resulting recognition rates.

The remainder of this paper is organized as follows. Section 3.2 presents an overview of incremental learning and dynamic selection. In section 3.3, the proposed LoGID approach is described in greater detail. The experimental evaluation is presented in section 3.4, and the conclusions and future work are discussed in section 3.5.

## 3.2 Related Work

In this section we present an overview of state-of-the-art approaches to both incremental learning and dynamic selection, which complements the description of the main motivation for this work.

### 3.2.1 Incremental Learning (IL)

This type of learning involves the updating of an existing classifier, or pool of classifiers, which, for the sake of simplicity, we refer to as a classification scheme. The main goal of IL is to

incorporate the knowledge that is intrinsically present in previously unobserved chunks of data into an existing system.

Ideally, an algorithm that conducts IL will meet the following requirements (Polikar et al., 2001):

A. Incorporation of new knowledge into an existing classification scheme;

B. No loss of previous knowledge in the process; if there is a loss, the system is said to suffer from *catastrophic forgetting*;

C. Reduction of the complexity overhead of batch learning (BL), the requirements of which, in terms of memory and time, increase as the the size of the training set increases; if there is no reduction, this type of algorithm would be meaningless.[3]

In the past, researchers have focused on developing algorithms to meet the above requirements, for different types of classifiers. Most of these focus exclusively on single-classifier systems, in an attempt to change the parameters of a given type of classifier directly (Mongillo and Deneve, 2008; Mizuno et al., 2000; Florez-Larrahondo et al., 2005). Many authors have proposed one-pass versions of BL counterparts (Mizuno et al., 2000; Florez-Larrahondo et al., 2005). However, given that BL algorithms generally rely on an appropriate number of training iterations, one-pass training usually results in lower classifier performance. Also, many decisions are required based on the current state of the classifier at a given time, potentially introducing bias either through the current chunk of data or the current state of the classifier.

The issues associated with single classifier-based IL algorithms have drawn the attention of many experts on this subject to the use of ensembles of classifiers (EoCs) (Polikar et al., 2001; Cavalin et al., 2009; Yu-Shu and Yi-Ming, 2009; Ulas et al., 2009; Kapp et al., 2010). When new data are available, new members can be appended to an existing pool of classifiers. Since

[3]Some authors maintain that the algorithm should use no previous data at all (Polikar et al., 2001), but this constraint is often relaxed, since in many cases a global overview might be necessary for making certain decisions, which may be aided by some data that have been stored, such as a validation set (Cavalin et al., 2009).

the existing members had been trained on old data and the new members were trained on new data, the new pool combines both old and new information. In this case, the new classifiers can be trained with the same algorithms used for BL. Combining different types of classifiers into a single pool can also be useful for enhancing the recognition capability of an EoC. Furthermore, this approach does not suffer from catastrophic forgetting, since once a classifier has been trained, its set of parameters remains the same. In addition, the knowledge modeled from useless or noisy data can be filtered after enough training data have been observed, since each classifier models a different time step of the learning process.

Nevertheless, the diversity of EoC members might be better exploited in many situations. Most of the existing IL algorithms based on EoCs use a static approach to combine classifiers (Polikar et al., 2001; Cavalin et al., 2009; Ulas et al., 2009). This approach is suboptimal, however, since not all classifiers are useful for recognizing all the test samples. Dynamic selection approaches, in contrast, may improve the potential of ensembles in IL. It has been previously demonstrated that dynamic weighting of classifier outputs may result in better recognition rates than static classifier combination (Gangardiwala and Polikar, 2005; Muhlbaier et al., 2009). However, these approaches select classifiers by considering only local points of view, i.e. only the information related to each classifier is used to weight their outputs. Therefore, a dynamic selection approach based on the use of output profiles might be more appropriate for this problem, since it would evaluate the behavior of the base classifiers working together.

### 3.2.2 Dynamic Selection (DS)

A multiple classifier system is composed of a pool of base classifiers, which we refer to as $C$. The dynamic selection of classifiers consists of finding a subset of classifiers $C'_i$, where $C'_i \subset C$, which contains the best members for recognizing the test sample $x_{i,test}$ (Ko et al., 2008; Woods et al., 1997; Giacinto and Roli, 2001; Zhu et al., 2004; Dos Santos et al., 2008; Soares et al., 2006). In the literature, the best subset of classifiers $C'_i$ is generally associated with the highest level of competence, which can be computed by, for instance, K nearest neighbors (Ko et al., 2008; Woods et al., 1997), clustering (Kuncheva, 2000), multiple training datasets (Singh and

Singh, 2005), or measures considering the outputs produced by the base classifiers (Dos Santos et al., 2008).

Recently, instance-based DS approaches (Ko et al., 2008; Cavalin et al., 2011a, 2010) have been proposed. These approaches are able not only to robustly select a classification scheme dynamically, but also to allow for the parameters of the system to be adapted to new data, in an IL setting, by including new samples in their dynamic selection set. As a consequence, this type of method is promising not only for conducting DS, but also to be combined with an EoC-based IL algorithm and define an adaptive framework which can: 1) conduct IL with the base classifiers; 2) dynamically select the best classifiers to recognize each test sample; and 3) improve the DS algorithm by appending new examples to the set of instances.

Among the existing instance-based methods, the DSA$^c$ approach stands out since it can be used with different types of base classifiers and can be applied to different pattern recognition problems easily, owing to its use of output profiles. This approach compute the best classification scheme for recognizing a test sample, which, in this case is a structure called dynamic multi-stage organization, by evaluating the similarity between the output profile of the test sample and that of each validation sample. The disadvantage of this method is that it depends on an off-line optimization phase to generate a pool of EoCs. This dependency is suboptimal for designing adaptive systems, since numerous computations should be performed to update the EoC pool after a new member has been included. For this reason, we propose a new DS approach in this work, called the K-nearest Output Profiles (KNOP), which is described in the next section. This approach is designed to embed the steps used by the KNORA algorithm (Ko et al., 2008) into the architecture of DSA$^c$ to define EoCs during the operational phase. KNORA is able to define EoCs in a completely dynamic fashion. By combining the advantages of both the DSA$^c$ and KNORA approaches, the proposed KNOP method can be used with various types of base classifiers, can be easily adapted to different pattern recognition problems, and can define EoCs in a completely dynamic fashion.

## 3.3 The LoGID Framework

The main objective of our proposed framework, LoGID, is to adapt an EoC-based system during both the learning and generalization phases to make it better able to deal with factors that may lead to recognition uncertainty, such as small training sets. In other words, consider a pool of classifiers $C$ as the current state of the baseline system, the training data stream containing labeled samples, and the test data consisting of unlabeled samples. Suppose that $C$ has been deployed and new chunks of training data become available over time. This framework is designed to update the knowledge embedded in the base classifiers $C$ whenever a block of unprocessed training data, represented by $D_t$, is available at a given time $t$, and select the best components for recognizing a given test sample $x_{i,test}$.

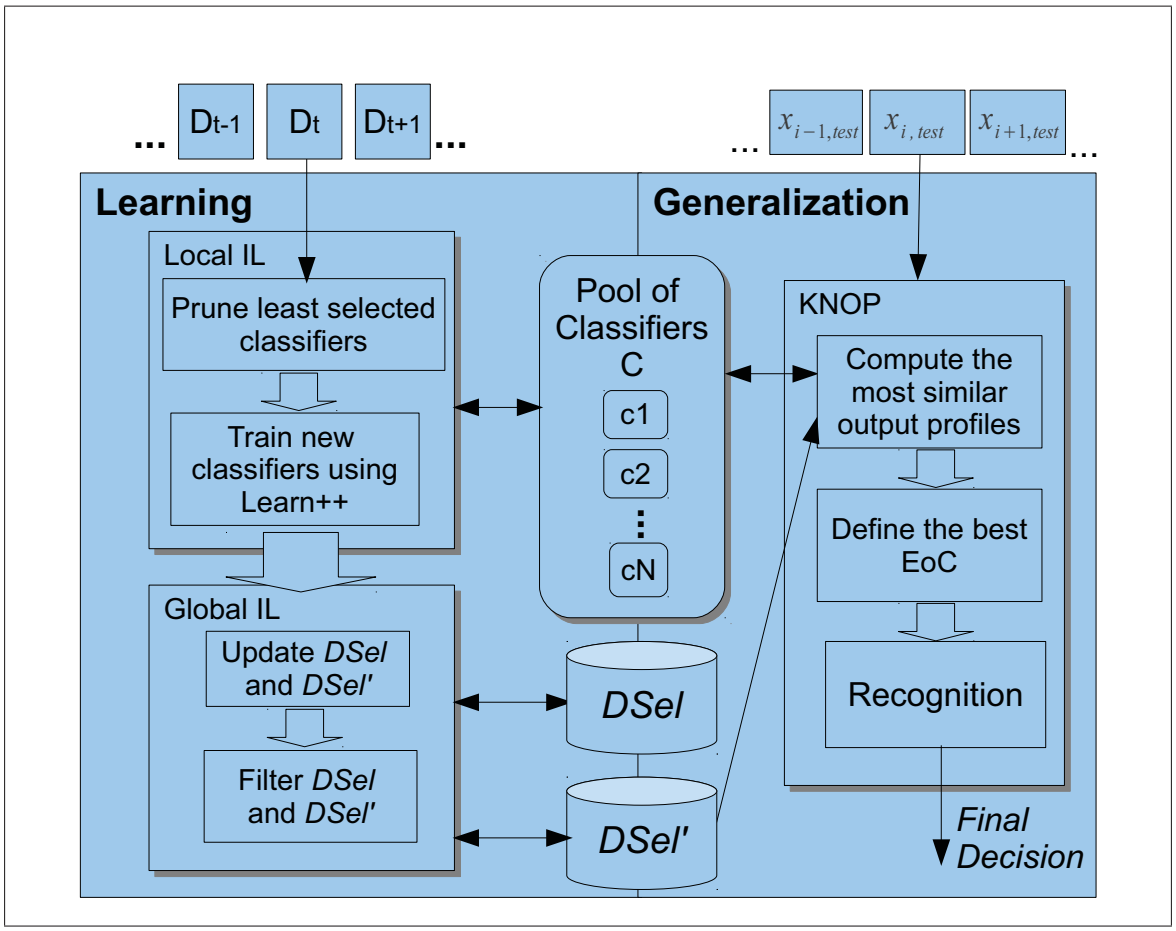



Figure 3.1 General overview of the LoGID architecture.

The LoGID framework is divided into two phases: learning and generalization. These phases become active according to the data presented to the framework. A general overview of this framework is depicted in Figure 3.1, and its main steps are formalized in Algorithm 3.1. The generalization phase involves inputing the test sample $x_{i,test}$ to LoGID, as defined in step 2. In this phase, the KNOP method is used to dynamically select the best EoC in step 5, containing members of the pool of classifiers $C$, for recognizing $x_{i,test}$. This task relies on comparing the output profile of the test sample $x_{i,test}$ with the output profiles stored in the set $DSel'$, which are related to the samples stored in the dynamic selection set $DSel$, as indicated in step 4. Then, the final recognition is conducted (step 6). The learning phase is activated in step 8, if a block of unprocessed training data $D_t$ is presented to the framework. Note that using the KNOP algorithm during generalization allows for conducting two types of incremental learning during the learning phase: local and global. Local incremental learning consists of updating the pool of classifiers $C$ by discarding the classifiers considered the least useful, and by appending new members to the pool, trained with the current block of data $D_t$, in steps 11 and 12, respectively. Global incremental learning involves updating the knowledge used to conduct dynamic selection. In step 14, the set $DSel'$ is updated by including the output profiles computed from the samples in $D_t$. In step 15, $DSel'$ is then filtered to remove irrelevant samples.

In the remainder of this section, we present the learning and generalization phases in greater detail.

### 3.3.1 Learning Phase - Local and Global Incremental Learning

During the learning phase, at a given time $t$, the previously unobserved block of data $D_t$ is processed by LoGID. The main goal here is to update the sources of knowledge used during the generalization phase, which is explained in Section 3.3.2. First, local incremental learning is conducted to update the pool of base classifiers $C$. Next, global incremental learning is carried out to update the set of output profiles, $DSel'$.

Algorithm 3.1 The main steps of the LoGID framework.

**Input:** $C$, the base classifiers, and either $D_t$, the unseen block of data, or $x_{i,test}$, the test sample.
**if** $x_{i,test}$ is inputed **then**
    *# The generalization phase is called*
    Find the $K$ output profiles in $DSel'$ which are the most similar to the output profile of $x_{i,test}$
    Define the best EoC
    Conduct the recognition of $x_{i,test}$ and provide the final decision
**else**
    **if** $D_t$ is inputed **then**
        *# The learning phase is called*
        *# First, local incremental learning is conducted*
        Prune the least selected classifiers from $C$
        Train new classifiers and add them to $C$
        *# Then, global incremental learning is done*
        Update $DSel$ and $DSel'$
        Filter $DSel$ and $DSel'$
    **end if**
**end if**

#### 3.3.1.1 Local Incremental Learning - Updating the pool of base classifiers

In this module, new knowledge is introduced to the pool of classifiers $C$ by appending to it a new set of classifiers. In other words, if we suppose that $C_{t-1}$ corresponds to the current state of $C$ at a given time $t$, and that $C_t$ corresponds to the newly generated set of classifiers, then $C$ is updated by concatenating $C_{t-1}$ with $C_t$, i.e. $C = C_{t-1} \cup C_t$.

The new classifiers $C_t$ are generated with the data provided by the current block $D_t$. For this task, we consider the Learn++ algorithm (Polikar et al., 2001). This algorithm can create a set of classifiers for each new block of data, using a distribution that weights the selection of a sample from the current block of data $D_t$. The resulting EoC is likely to be diverse since samples that have not been previously observed, or samples that have not been properly modeled in $C$, have a greater chance of being selected.

Algorithm 3.2 Local incremental learning.

1: **Input:** $C$, the base classifiers, and $D_t$, the unseen block of data.
2: **if** $N > N_{max}$ **then**
3: By considering the control mechanism defined in Section 3.3.1.1.1, prune the $N - N_{max}$ least used classifiers from $C$
4: **end if**
5: Call Algorithm 3.3 to update $C$
6: **Output:** the updated pool of classifiers $C$

Prior to the creation of the classifier pool $C_t$, a pruning method is used to eliminate from the current pool $C$ the members that are considered the least useful. This pruning is aimed at avoiding the performance of useless computations during the recognition phase. To define the usefulness of a classifier, the usage statistics of the pool $C$ are computed on the block $D_t$.

The main steps of local incremental learning are presented in Algorithm 3.2. First, a predefined threshold, denoted $N_{max}$, is used to evaluate the size of $C$ (steps 2 to 4). If $N$ is larger than $N_{max}$, the least useful classifiers are removed from $C$ (step 3). Then, Learn++ is called upon to create the new set of classifiers $C_t$ to update the pool of classifiers $C$ (step 5).

#### 3.3.1.1.1 Pruning the pool of base classifiers

To prune $C$, the predefined threshold $N_{max}$ and the usage statistics of the classifiers, computed on $D_t$, are considered. If the size of $C$ is above that threshold, i.e. $N > N_{max}$, only the $N_{max}$ most used classifiers are kept in $C$, while the remaining $N - N_{max}$ members are discarded.

The usage statistics of a classifier correspond the number of times this member of $C$ has been selected to conduct recognition during the generalization phase. To compute these statistics, a validation step is conducted by taking into account the current pool of classifiers $C$, the block of data $D_t$, and the KNOP algorithm (see Section 3.3.2). The samples in the set $D_t$ are used to validate the current state of $C$ with the KNOP algorithm. That is, each sample in $D_t$ is recognized by the KNOP algorithm, and the number of times each classifier in $C$ has been used to compose the dynamically selected EoC is stored. After evaluating the entire set $D_t$, the

usage statistics of a classifier correspond to the total number of times it has been selected to compose EoCs, considering all the EoCs that have been dynamically defined in this process. Ultimately, we assume that the more often a base classifier is selected to recognize the samples in $D_t$ (i.e. the higher its value based on usage statistics), the more useful this classifier is. At the same time, the classifiers that are used least should be replaced by new ones, trained with samples from $D_t$.

#### 3.3.1.1.2 The Learn++ algorithm

The Learn++ algorithm is used to update $C$ with the newly generated classifiers $C_t$ trained with the data present in the block of data $D_t$. The algorithm processes this block over $T_k$ iterations, where $T_k$ corresponds to the number of new classifiers to be generated at each time $t$. During each iteration $k$, where $1 \leq k \leq T_k$, a new classifier $c_k$ is created and put into $C$. For each classifier $c_k$, two disjoint subsets of $D_t$, denoted $TR_k$ and $VL_k$, are considered as training and validation subsets, respectively. The samples for $TR_k$ and $VL_k$ are chosen based on the distribution $DIST_t$. This distribution is first initialized uniformly, then updated based on the performance of the current pool of classifiers to ensure that examples misclassified by the current ensemble have a high probability of being sampled to compose the training set for the next classifier. In an incremental learning setting, the examples with a high probability of being subjected to error are those that are unknown, or are yet to be used to train the classifier.

The main steps of Learn++ are formalized in Algorithm 3.3. Each block of data $D_t$, at a given time $t$, is associated with the distribution $DIST_t$. At the beginning of each iteration $k$, i.e. in step 4, $DIST_t$ is updated according to current weights stored in $w_k$. Next, in step 5, this distribution is used to select two subsets of samples from $D_t$: $TR_k$ and $VL_k$. These subsets are used as training and hold-out validation sets respectively, to generate a new classifier $c_k$ during step 6. The next steps consist of evaluating whether or not $c_k$ is a sufficiently accurate classifier, first individually and then as a member of the pool $C$. Throughout step 7, the individual error rate $\varepsilon_k$ of this classifier is computed on $D_t$. If $\varepsilon_k$ is above $1/2$, $c_k$ is discarded and the algorithm jumps back to step 3 (step 9). If $c_k$ is not discarded during the individual evaluation, this classifier is

Algorithm 3.3 The Learn++ algorithm.

1: **Input:** the block of data $D_t$, the pool of classifiers $C$
2: Initialize $w_1(i) = 1/|D_t|$
3: **for** $k = 1$ **to** $T_k$ **do**
4: Set $DIST_t = w_k / \sum_{i=1}^{|D_t|} w_k(i)$, so that $DIST_t$ is a distribution
5: Choose the subsets $TR_k$ and $VL_k$ from $D_t$, according to $DIST_t$
6: Train a new classifier $c_k$, providing it with $TR_k$ for training and $VL_k$ for validation
7: Considering $c_k$, calculate its individual error rate $\varepsilon_k$ on $D_t$
8: **if** $\varepsilon_k > 1/2$ **then**
9: Set $k = k-1$, discard $c_k$ and go to step 5.
10: **else**
11: Put $c_k$ in $C$
12: **end if**
13: Considering $C$, compute the composite error rate $E_k$ on $D_t$
14: **if** $E_k > 1/2$ **then**
15: Set $k = k-1$, remove $c_k$ from $C$, and go to step 5.
16: **end if**
17: Set $B_k = E_k/(1-E_k)$ (normalized composite error), and update the weights of the instances $w_{k+1}(i) = w_k(i) \times \begin{cases} B_k, \text{ if } C \text{ provides the correct decision for the sample } x_i \\ 1, \text{ otherwise} \end{cases}$, where $x_i$ is a sample from $D_t$.
18: **end for**
19: **Output:** the updated pool of classifiers $C$

added to the pool of classifiers $C$ (step 11), and the composite error rate $E_k$ of this updated pool is computed on $D_t$ (step 13). If $E_k$ is above $1/2$, $c_k$ is discarded (step 15) and the algorithm goes back to step 3. Otherwise, $c_k$ is kept in $C$, and the algorithm keeps iterating until all $T_k$ new classifiers have been added to $C$. However, before jumping to the next iteration, the weights $w_k$ are updated by considering the normalized composite error $B_k$ (step 17). As a result, these weights can be used to compute the distribution $DIST_t$ in the next iteration, which will in turn be used in selecting the next training and validation subsets, $TR_{k+1}$ and $VL_{k+1}$ respectively. Note that $w_k$ is initialized uniformly, as in step 2. For a detailed description of Learn++, see (Polikar et al., 2001).

#### 3.3.1.2 Global Incremental Learning - Updating the set of output profiles

Global incremental learning involves updating the set of output profiles $DSel'$ to accommodate new output profiles, when a new block of data $D_t$ is available at a given time $t$. This process is designed to improve the knowledge used by the KNOP algorithm (see Section 3.3.2) to dynamically select EoCs.

This phase, formalized in Algorithm 3.4, consists of the following steps. First, the current set of output profiles $DSel'$ is updated to incorporate the knowledge introduced by the new classifiers in $C$, created with local incremental learning (steps 2 to 4). The outputs of these new classifiers are computed from the corresponding sample stored in $DSel$. Second, the output profiles computed from the current block of data $D_t$ are appended to $DSel'$, and their corresponding observation sequences are saved in $DSel$ (steps 5 to 9). That is, for each sample $x_{j,unseen}$ in $D_t$, the corresponding output profile $\tilde{x}_{j,unseen}$ is added to $DSel'_t$. Accordingly, $\tilde{x}_{j,unseen}$ is added to $DSel'_t$. Then, $DSel$ is concatenated with $DSel_t$, and $DSel'$ is concatenated with $DSel'_t$ (steps 10 and 11). Finally, a filtering mechanism removes the samples that are considered the least relevant from $DSel'$, and consequently from $DSel$ (steps 12 to 18).

##### 3.3.1.2.1 Filtering dynamic selection samples

The proposed mechanism keeps in $DSel$ and $DSel'$ only the samples that belong to the "zone of relevance". This zone is computed by considering the normalized margin presented in Equation 3.1. Note that $v1_j$ corresponds to the number of votes received by the winning class, computed from the outputs yielded by $C$ for the sample $\tilde{x}_{j,dsel}$ in $DSel'$. Similarly, $v2_j$ corresponds to the number of votes received by the class placing second.

$$m_j = \frac{v1_j - v2_j}{N}, \text{ where } 0 \leq m_j \leq 1 \text{ and } N = |C| \tag{3.1}$$

Two predefined thresholds, ranging from 0 to 1.0 and denoted $\vartheta_{min}$ and $\vartheta_{max}$, define the "zone of relevance". If the normalized margin $m_j$ is within the region defined by $\vartheta_{min} \leq m_j \leq \vartheta_{max}$,

Algorithm 3.4 The global incremental learning algorithm.

**Input:** the block of data $D_t$, the pool of classifiers $C$, the current dynamic selection set $DSel$, and the current set of output profiles $DSel'$
**for** each $x_{j,dsel}$ in $DSel$ **do**
    Update its corresponding output profile in $DSel'$, i.e. $\tilde{x}_{j,dsel}$, to store the outputs of the new classifiers in $C$
**end for**
**for** each $x_{j,unseen}$ in $D_t$ **do**
    Compute $\tilde{x}_{j,unseen}$, i.e. the output profile of $x_{j,unseen}$
    Put $x_{j,unseen}$ in $DSel_t$.
    Put $\tilde{x}_{j,unseen}$ in $DSel'_t$.
**end for**
$DSel = DSel \cup DSel_t$
$DSel' = DSel' \cup DSel'_t$
**for** each $x_{j,dsel}$ in $DSel$ **do**
    Compute $m_j$ for $\tilde{x}_{j,dsel}$, using Equation 3.1
    **if not**($\vartheta_{min} \leq m_j \leq \vartheta_{max}$) **then**
        Remove $x_{j,dsel}$ from $DSel$.
        Remove $\tilde{x}_{j,dsel}$ from $DSel'$.
    **end if**
**end for**
**Output:** the updated sets $DSel$ and $DSel'$

the sample $\tilde{x}_{j,dsel}$ is kept in $DSel'$. Otherwise, the sample is discarded. These thresholds define the minimum and maximum margins that a sample in $DSel'$ must present. This mechanism allows samples that present a larger than expected margin value, that is $m_j > \vartheta_{max}$, to be excluded from $DSel'$. These may be redundant samples, and would negatively affect recognition time. In addition, the mechanism is useful for excluding samples with too low a margin value, i.e. $m_j < \vartheta_{min}$. These samples could negatively affect recognition performance by introducing noise.

### 3.3.2 Generalization Phase - The KNOP Algorithm

The generalization phase involves the application of the K-nearest Output Profiles (KNOP) algorithm, depicted in Figure 3.2, to recognize each test sample $x_{i,test}$. The main steps of this algorithm are presented as Algorithm 3.5. First, the test sample $x_{i,test}$ is converted into an

output profile, denoted as $\tilde{x}_{i,test}$ (step 2). In this work, an output profile contains the scores yielded by all the HMM-based classifiers belonging to the pool $C$. In step 3, the output profiles in $DSel'$ that are the $K$ most similar to $\tilde{x}_{i,test}$ are stored in $\Psi_i$. Next, the samples in $\Psi_i$ are used to define the best ensemble $C_i^*$ for recognizing $x_{i,test}$ (steps 5 through 11), where $|C_i^*| = U_i$. For each sample in $\Psi_i$, if a classifier $c_k$ correctly recognizes this sample, it is added to $C_i^*$.

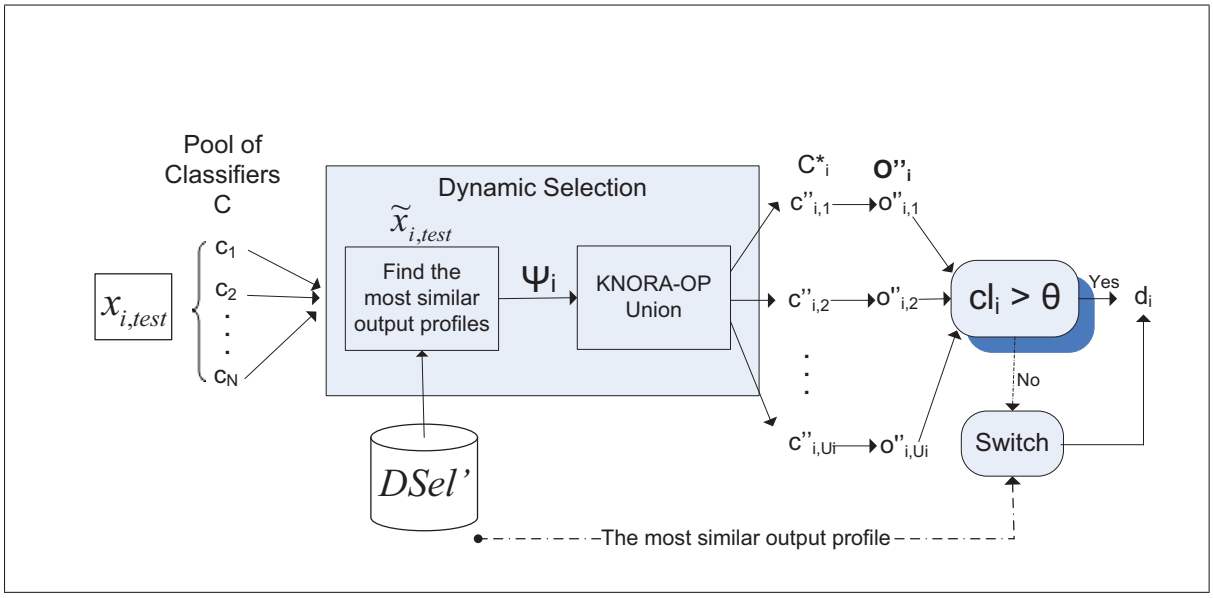



Figure 3.2 Overview of the KNOP approach

Before computing the final decision $d_i$, the switch mechanism is used (steps 12 to 18), the main objective of which is to avoid relying on low-confidence decisions. In this case, if the confidence level $cl_i$ of the voting provided by the ensemble $C_i^*$ is above the predefined threshold $\theta$, the decision made by this ensemble is considered as the final one. Otherwise, the label of the most similar output profile in $DSel'$ represents $d_i$.

We describe the main modules of the KNOP algorithm in greater detail below.

#### 3.3.2.1 Computing output profiles using scores

In this work, we have defined an output profile as the vector containing a concatenation of the scores yielded by all the HMM-based classifiers in the pool $C$. The main goal is to compute

Algorithm 3.5 Complete KNOP algorithm for HMMs.

**for** each data point $x_{i,test}$ in $Test$ **do**
  Compute $\tilde{x}_{i,test}$ using transformation $T$, as defined in Equation 3.2
  Considering $DSel'$, find the $K$ $\tilde{x}_{j,dsel}$ most similar to $\tilde{x}_{i,test}$ and put into $\Psi_i$
  $C_i^* = \emptyset$
  **for** each $\tilde{x}_{j,dsel}$ in $\Psi_i$ **do**
    **for** each $c_k$ in $C$ **do**
      **if** KNORA-Union's rules are satisfied **then**
        Insert $c_k$ into $C_i^*$
      **end if**
    **end for**
  **end for**
  Compute $cl_i$ from $C_i^*$ using Equation 3.3
  *# Switch mechanism*
  **if** $cl_i > \theta$ **then**
    $d_i =$ most voted class from $C_i^*$
  **else**
    $d_i =$ the label of the most similar $\tilde{x}_{j,dsel}$ from $DSel'$
  **end if**
**end for**

the similarity between the samples in the decision space, considering information related to all classes. In this case, classifiers that do not conduct recognition with enough confidence for some samples can still contribute to the similarity computation. This would not be possible with output profiles formed only by the crisp label outputs, as in (Cavalin et al., 2011a).

Consider the HMM-based classifier $c_j$ as the set of HMMs $\Lambda_j = \{\lambda_{j,1}, \ldots, \lambda_{j,M}\}$, where $M$ corresponds to the number of classes. Consider, too, the set of base classifiers as the set of $N$ HMM-based classifiers $C = \{\Lambda_1, \ldots, \Lambda_N\}$, and the following transformation:

$$T : x_i \Rightarrow \tilde{x}_i, \tag{3.2}$$

Let $L_{i,j}^* = \{L_{i,j,1}(O|\lambda_{j,1}), \ldots, L_{i,j,M}(O|\lambda_{j,M})\}$ be the set of likelihoods produced by $\Lambda_j$ for all classes, given $x_{i,test}$. Also, consider the set of scores produced by $\Lambda_j$ for the same $x_{i,test}$ as

$S^*_{i,j} = \{S_{i,j,1}, \ldots, S_{i,j,M}\}$, where $S_{i,j,k} = L_{i,j,k}(O|\lambda_{j,k}) / \sum_{l=1}^{M} L_{i,j,l}(O|\lambda_{j,l})$. We denote an output profile as $\tilde{x}_i = \{S^*_{i,1}, \ldots, S^*_{i,N}\}$. Given that $x_{i,test}$ represents an observation sequence that is to be processed by the HMMs, $\tilde{x}_{i,test}$ represents the vector of scores produced by all of the $N \times M$ HMMs in $C$ for this observation sequence.

#### 3.3.2.2 KNORA-OP-Union: Dynamically defining the best EoC

Consider $\tilde{x}_{i,test}$ to be the output profile of $x_{i,test}$ and $DSel'$ to contain the output profiles of all samples in $DSel$, i.e. for each $x_{j,dsel}$ the corresponding $\tilde{x}_{j,dsel}$ is stored in $DSel'$. The dynamically selected ensemble $C^*_i$ is computed in two steps. First, the $K$ output profiles $\tilde{x}_{j,dsel}$ in $DSel'$ that are the most similar to $\tilde{x}_{i,test}$, considering the Euclidean distance, are stored in $\Psi_i$. Then, a selection algorithm inspired by the KNORA-OP-Union method (Batista et al., 2011) uses the output profiles in $\Psi_i$ to choose the best members belonging to $C$ to compose the EoC $C^*_i$.

The above selection algorithm works as follows. Let $O = \{o_1, \ldots, o_N\}$ be the crisp label outputs of the classifiers in $C$. Given the output profiles $\tilde{x}_{j,dsel}$ stored in $\Psi_i$, suppose each $\tilde{x}_{j,dsel}$ has been correctly classified by a set of classifiers $C'_j$. Every classifier $c_i \in C'_j$ must be contained in the final ensemble $C^*_i$ and should submit a vote on the sample $\tilde{x}_{i,test}$. Note that a classifier may be present in $C^*_i$ more than once if it correctly classifies more than one sample in $\Psi_i$.

After $C^*_i$ is computed, the final decision is evaluated by the switch mechanism.

#### 3.3.2.3 The switch mechanism

The switch mechanism depends on the confidence level $cl_i$ of the outputs of the dynamically selected EoC $C^*_i$ to recognize the test sample $x_{i,test}$. This confidence level is computed using Equation 3.3.

$$cl_i = \frac{v1_i - v2_i}{K \times N} \tag{3.3}$$

$cl_i$ is based on the margin (Hansen et al., 1997) of the ensemble $C_i^*$, which is the difference between the number of votes of the two classes with the most votes, i.e. $v1_i$ and $v2_i$, given $x_{i,test}$ and $C_i^*$. When the margin is high enough, $C_i^*$ performs the recognition with a high level of confidence. By analogy, when that margin is low, the confidence level of the EoC is low. Consequently, the use of a threshold $\theta$ makes it possible to reject low confidence decisions outputted by the ensemble and to rely on another source of knowledge. In other words, if the confidence level is above the predefined threshold $\theta$, i.e. $cl_i > \theta$, the outputs of the members of $C_i^*$ provide the final decision. Otherwise, the switch uses the most similar output profile in $DSel'$.

Note that the size of $C_i^*$ is dependent on the cardinality of $C$, i.e. $N$. Given that this size might change over time, in Equation 3.3 the margin is normalized by the maximum possible size for $C_i^*$, i.e. $K \times N$. In this case, a single continuous value for $\theta$, in the 0 to 1 range, can be used for pools of classifiers with different cardinalities.

## 3.4 Experiments

In this section, we present our experimental evaluation of the proposed LoGID approach. Since the implementation of this framework focuses on HMMs, the experimental protocol includes observation sequences extracted from four different databases. These databases are listed in Table 3.1.

Table 3.1 The databases considered in this work

| Database | Problem | Source |
|---|---|---|
| *Japanese Vowels* | Voice recognition | UCI (Frank and Asuncion, 2010) |
| *Arabic Spoken Digits* | Speech recognition | UCI (Frank and Asuncion, 2010) |
| *Isolated Uppercase Letters* | Handwriting recognition | NIST SD19 |
| *Isolated Digits* | Handwriting recognition | NIST SD19 |

The parameters of the proposed approach are presented in Table 3.2 for each database. This table also presents the number of features, the size of the datasets, and the number of classes for each database. For NIST Letters and NIST Digits, the system proposed in (Britto et al., 2003)

is implemented as the baseline recognition system, extracting features from both the columns and the rows of the images. For the remaining databases, a single left-right HMM-based system is considered (Rabiner, 1989). The number of states for each left-right HMM is computed with Wang's method (Wang, 1994). The value of $K$ is set to 30 for the KNOP algorithm, since this value worked well for many applications in (Cavalin et al., 2011a). It is worth noting that, two distinct test sets are considered for the NIST Digits database, *test1* being generally known to be more difficult than *test2*. It is also worth noting that the Baum-Welch algorithm is used to train the HMMs in local incremental learning. To train each HMM, the set $TR_k$ is used to estimate parameters and $VL_k$ is used as a hold-out validation subset (see Algorithm 3.3).

Table 3.2 The main parameters for each database. The training data are equally distributed for all classes. Train, DSel, and Test: number of samples in the training, dynamic selection, and test set, respectively; Bl: number of blocks into which Train is divided for incremental learning; C: number of classes; Feat: number of input features; and CB: codebook size. $\theta$, $T_k$, $\vartheta_{min}$, $\vartheta_{max}$, and $N_{max}$: the main parameters for LoGID, set by evaluations on the dynamic selection set.

| **Database** | **Train** | **DSel** | **Test** | **Bl** | **C** | **Feat** | **CB** | $T_k$ | $\theta$ | $\vartheta_{min}$ | $\vartheta_{max}$ | $N_{max}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Japanese | 216 | 54 | 370 | 3 | 9 | 12 | 24 | 10 | 0.1 | 0.2 | 1.0 | 15 |
| Arabic | 5,500 | 1,100 | 2,200 | 10 | 10 | 13 | 64 | 10 | 0.0 | 0 | 1.0 | 100 |
| Letters | 43,160 | 11,941 | 12,092 | 10 | 26 | 47 | 256 | 5 | 0.3 | 0.2 | 0.8 | 100 |
| Digits | 180,000 | 10,000 | *test1* - 60,089 | 18 | 10 | 47 | 256 | 5 | 0.3 | 0.2 | 0.8 | 200 |
| | | | *test2* - 58,646 | | | | | | | | | |

Incremental learning settings are simulated by dividing the training sets into smaller chunks of data, which are equally distributed according to the number of blocks defined in Table 3.2. The block numbers have been empirically defined, with the aim of balancing them with a reasonable number of samples for training.

For a better statistical evaluation, ten different replications are conducted for each dataset, each of which considers a unique set of initialization parameters. The results are statistically validated by the Kruskal-Wallis nonparametric statistical test, and equality among the mean values is tested using a confidence level of 95%, with Dunn-Sidak correction applied to critical values.

### 3.4.1 Results

In this section, we first explain how the parameters have been set for the proposed method. This task considers only the dynamic selection set *DSel* and the first training block $D_1$. Next, we present the results on the test sets, processing of all the training blocks $D_t$.

#### 3.4.1.1 Parameter setting

In order to compute the best configuration for LoGID, the parameters $T_k$, $\theta$, $\vartheta_{min}$, $\vartheta_{max}$, and $N_{max}$ are evaluated with the following methodology:

A. Each parameter is evaluated in this sequence: $T_k$, $\theta$, $\vartheta_{min}$ and $\vartheta_{max}$, then $N_{max}$

B. Suppose that, during the design of the system the only data available are those in the first chunk of training data $D_1$ and in the initial dynamic selection set *DSel*, the size of which appears in Table 3.2. To set the configuration parameters, the performance is evaluated by dividing *DSel* into two distinct subsets of equal size. The first subset is used to compute the set of output profiles $DSel'$, and the second is used to evaluate the performance of the system. This scheme is repeated by swapping the subsets, and the average recognition rate represents the overall performance.

To evaluate the impact of $T_k$ and $\theta$, LoGID is implemented with no pruning of either the pool of classifiers or the dynamic selection set. For the former, the following values were considered: $(3, 5, 10, 15, 20)$. The parameter $\theta$ is evaluated in the 0.0 to 1.0 range, with an interval of 0.1 between each evaluation. Owing to space constraints, only the best values for these parameters are listed in Table 3.2.

The parameters $\vartheta_{min}$ and $\vartheta_{max}$ are evaluated by considering the following set of values (0, 0.2, 0.4, 0.6, 0.8, 1.0), and the results presented in Figures 3.3, 3.4, 3.5, and 3.6, for Japanese Vowels, Arabic Spoken Digits, NIST Letters, and NIST Digits, respectively. The best values for each database, in the same order, were: (0.2, 1.0), (0, 0.6), (0.2, 0.8), and (0.2, 0.8), for $\vartheta_{min}$

and $\vartheta_{max}$ respectively. Note that when two configurations yield similar results, the smallest difference between $\vartheta_{min}$ and $\vartheta_{max}$ is considered as the best configuration. The smallest value represents the narrowest region of relevance, so that fewer samples are kept in $DSel'$.

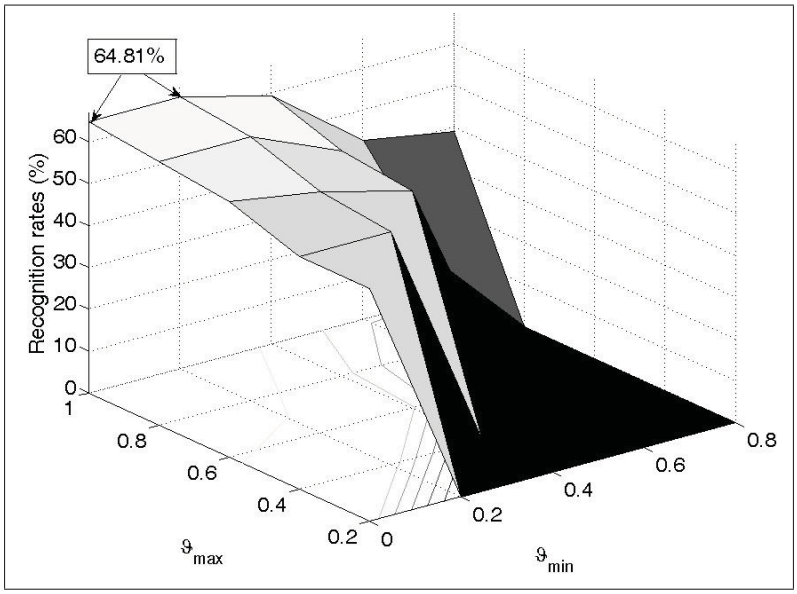



Figure 3.3 Evaluation of different values for $\vartheta_{min}$ and $\vartheta_{max}$ on the dynamic selection set of Japanese Vowels. The best recognition rates are reached with $\vartheta_{min} = 0.2$ and $\vartheta_{max} = 1.0$.

For each database, we evaluated a set of five different values for $N_{max}$. These values are based on empirically defined minimum and maximum sizes for $C$. The results are presented in Figures 3.7, 3.8, 3.9, and 3.10, for Japanese Vowels, Arabic Digits, NIST Letters, and NIST Digits, respectively. The best value for each database, in the same order, is: 10, 100, 80, and 120. We note that base classifier pruning works better on Japanese Vowels, with $N_{max} = 10$, and on the NIST Digits database, with $N_{max} = 120$. On Arabic Spoken Digits, however, the best value for $N_{max}$ is equal to the maximum size for $C$, i.e. 100. This means that the best option for this database is not to prune.

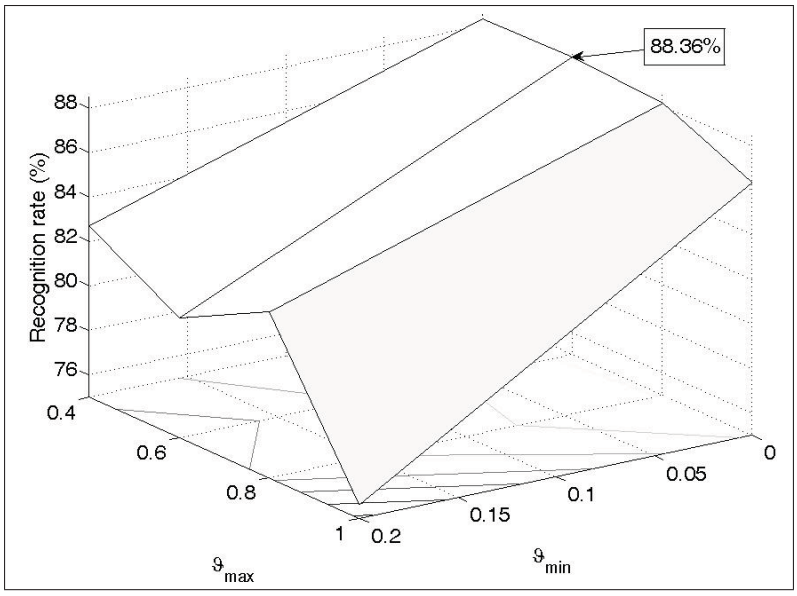



Figure 3.4 Evaluation of different values for $\vartheta_{min}$ and $\vartheta_{max}$ on the dynamic selection set of Arabic Digits. The best recognition rates are reached with $\vartheta_{min} = 0$ and $\vartheta_{max} = 0.6$.

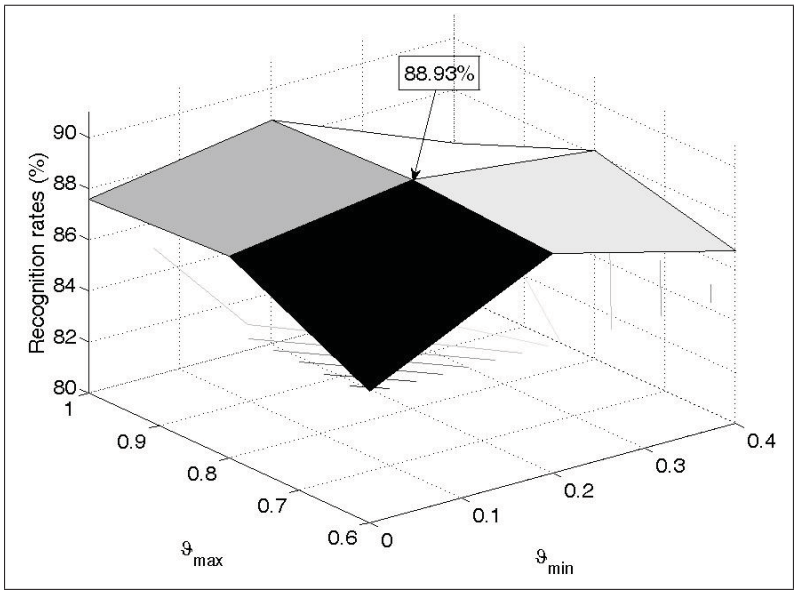



Figure 3.5 Evaluation of different values for $\vartheta_{min}$ and $\vartheta_{max}$ on the dynamic selection set of NIST Letters. The best recognition rates are reached with $\vartheta_{min} = 0.2$ and $\vartheta_{max} = 0.8$.

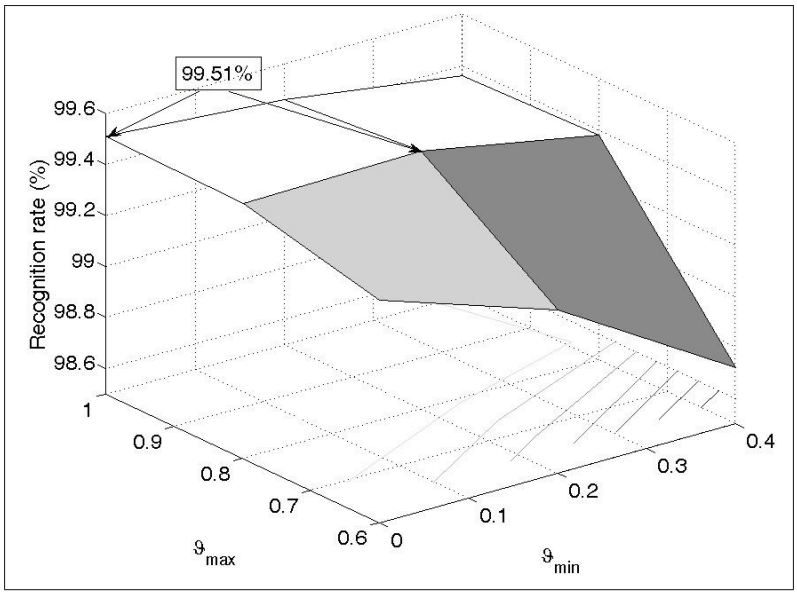



Figure 3.6 Evaluation of different values for $\vartheta_{min}$ and $\vartheta_{max}$ on the dynamic selection set of NIST Digits. The best recognition rates are reached with $\vartheta_{min} = 0.2$ and $\vartheta_{max} = 0.8$.

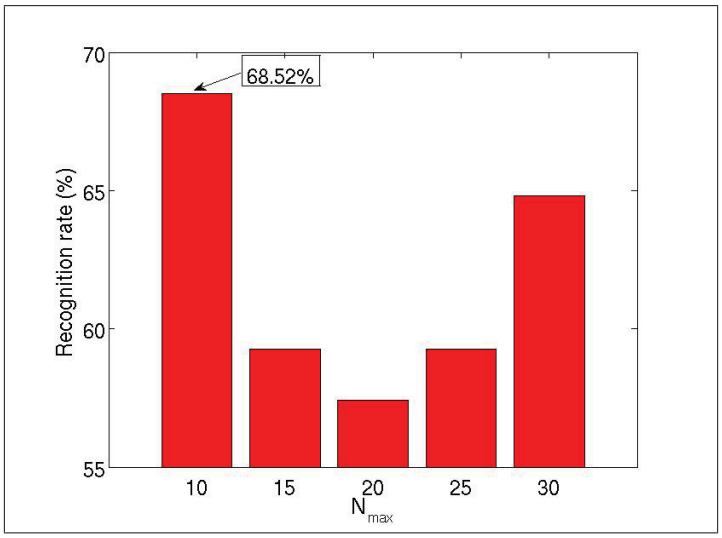



Figure 3.7 For Japanese Vowels, $N_{max} = 10$ provides the best recognition rates on the dynamic selection set.

#### 3.4.1.2 Performance evaluation

In this section, after computing the best parameters for LoGID using only the first block of data $D_1$, we evaluate the performance on the test set. The results of the proposed method are compared with:

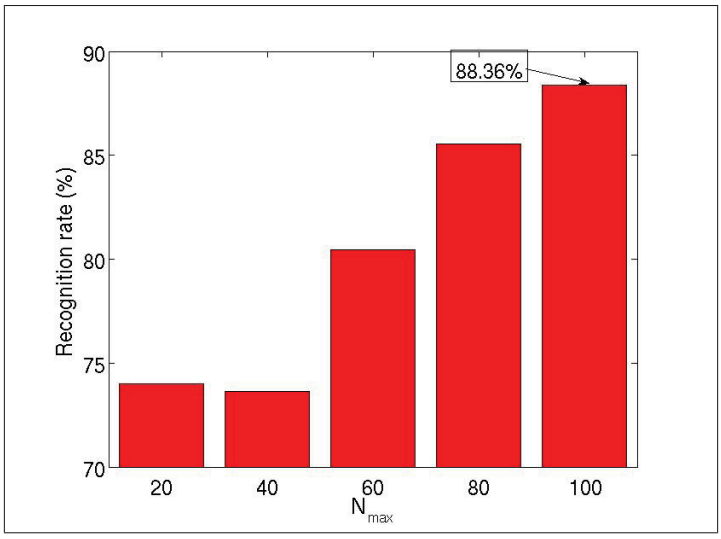



Figure 3.8 For Arabic Digits, $N_{max} = 100$ yields the best results on the dynamic selection set.

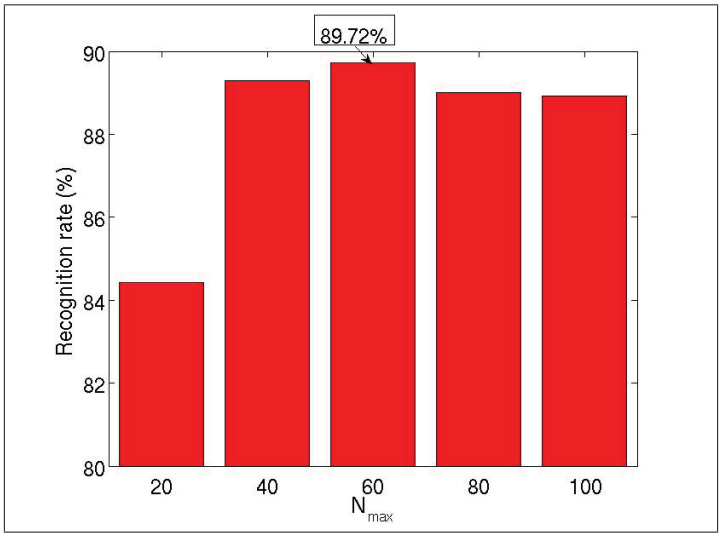



Figure 3.9 For NIST Letters, $N_{max} = 60$ yields the best results on the dynamic selection set.

- *Batch learning*: at each time $t$, a classifier trained with the current block of data and the data from all previous blocks replaces the current classifier. This approach provides an estimation of the empirical error bound on the pattern recognition problems, considering the same learner;

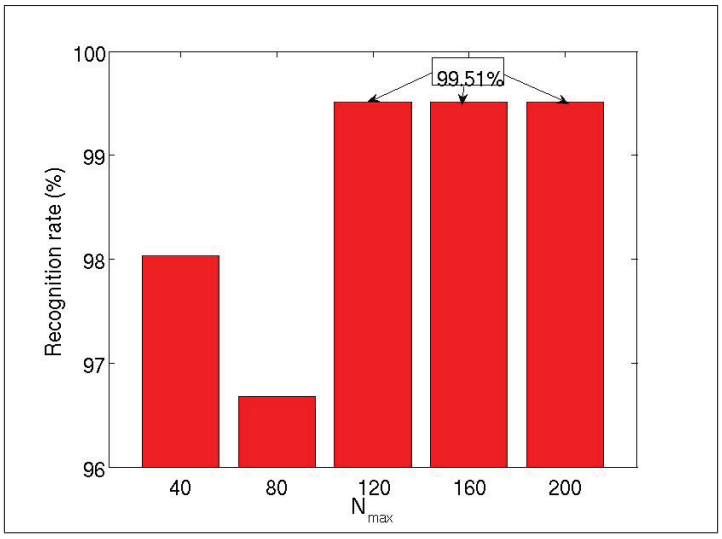



Figure 3.10 For NIST Digits, three values for $N_{max}$: 120, 160, and 200, yield the best results on the dynamic selection set. The smallest value, i.e. $N_{max} = 120$, is the preferred one.

- *Local IL*: a partial implementation of LoGID. Local incremental learning alone is conducted and the classifiers are statically combined during the generalization phase (we consider the product of the likelihoods (Britto et al., 2003) produced by the HMMs to be the fusion function). This approach mainly consists of the Learn++ algorithm, and allows comparison of the proposed method with a well-known EoC-based incremental learning algorithm (Polikar et al., 2001; Muhlbaier et al., 2009);
- *Global IL*: also a partial implementation of LoGID. The KNOP algorithm is used during generalization and only global incremental learning is considered during learning. In other words, an initial pool of classifiers is trained with the first training block, but this pool remains static during the system's lifetime. At the same time, however, new samples are appended to *DSel* and $DSel'$, so that new knowledge is incrementally added to the system. With this method, it is possible to evaluate how the dynamic selection algorithm evolves in an incremental learning setting using a fixed pool of base classifiers.

The results obtained for Japanese Vowels are depicted in Figure 3.11. LoGID achieves the best results on this database. Batch learning was the second best approach, but its final recognition

rates were about 10% lower than those of LoGID. The lowest recognition rates were presented by Local IL. Global IL performed slightly better than Local IL. However, the level of performance of the former decreased after new blocks of data had been learned. This indicates that global incremental learning works well for this problem only if it is combined with local incremental learning.

We depict the results of the evaluation of Arabic Digits in Figure 3.12. For this database, Batch learning achieved the second highest final recognition rates, at about 90.36%. However, this method achieved the lowest recognition rates with small amounts of data. This demonstrates that Batch learning may not perform very well when the degree of uncertainty is high, i.e. when only a small training set is available. LoGID, in contrast, demonstrated its ability to adapt to different levels of confusion. With both small and large training sets, the proposed approach yields the best performance.

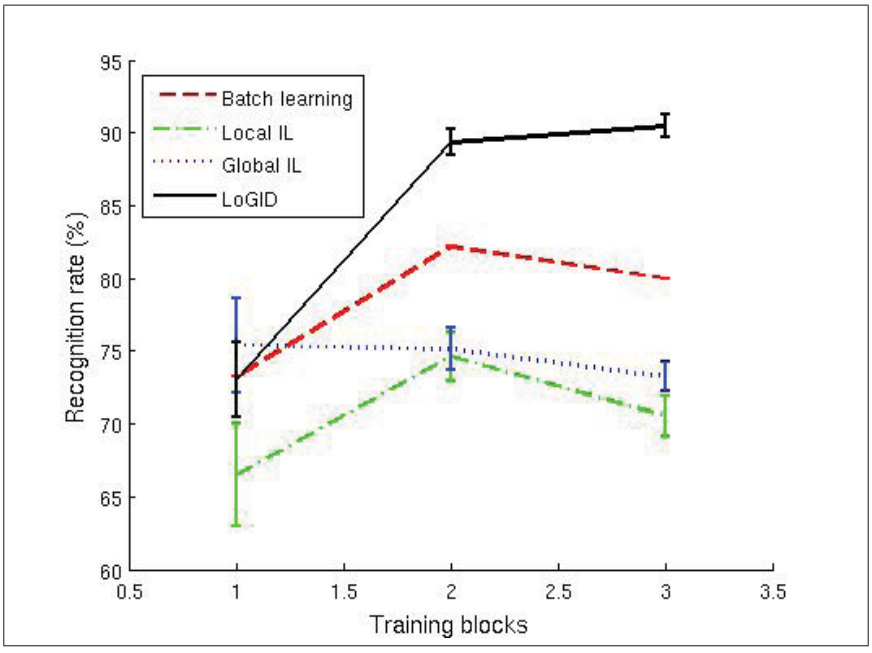



Figure 3.11 Performance comparison for Japanese Vowels.

The performance comparison for NIST Letters is presented in Figure 3.13. For this database, LoGID also achieves the best final recognition rates, at about 94.10%. The second best method was Batch learning, with 92.69% of recognition rates. The performance of LoGID with small

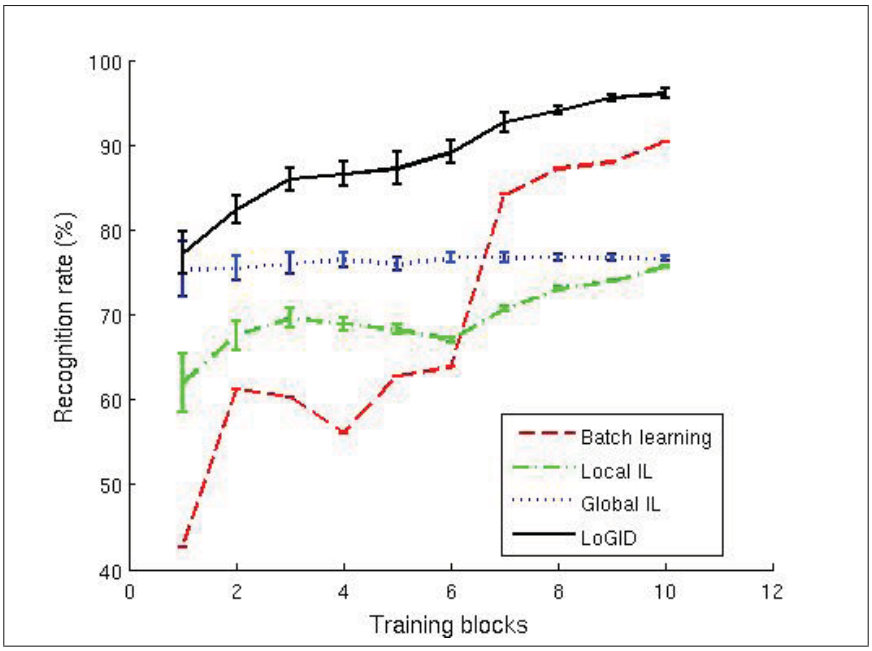



Figure 3.12 Performance comparison for Arabic Digits.

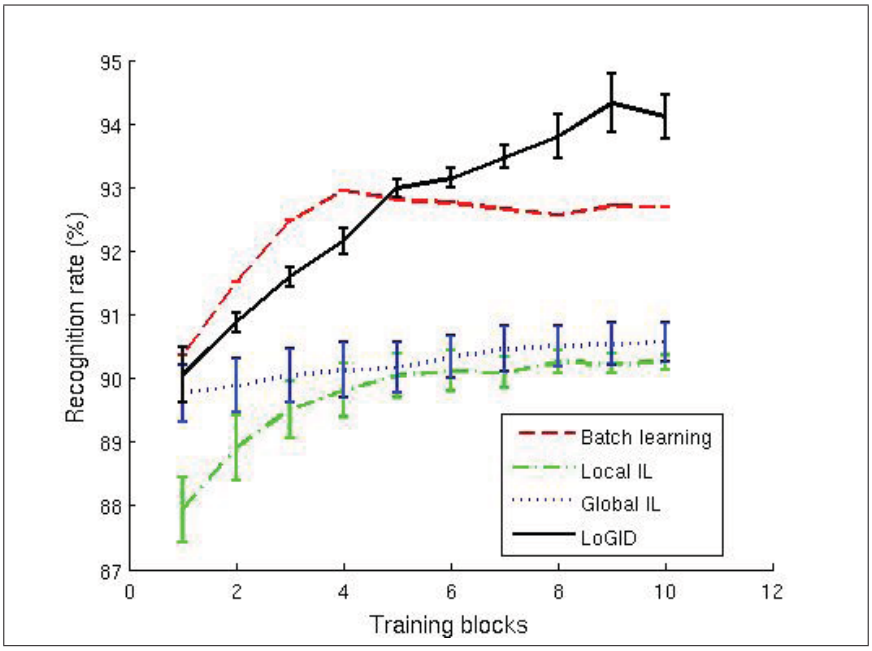



Figure 3.13 Performance comparison for NIST Letters.

training sets, though, was worse than the performance of Batch learning. But after learning the fifth block of data, LoGID began to present the best performance. Local IL (at 90.24%)

and Global IL (at 90.57%) performed similarly. The latter, however, achieves better recognition rates with fewer training data, showing that the dynamic selection may result in better performance when the level of uncertainty is high.

The performance comparison for NIST Digits in *test1* is presented in Figure 3.14. LoGID achieved the best recognition rates on this database, at about 98.84%. Global IL yielded the second best result, at 98.53%. We observe that the performance of both LoGID and Global IL evolves significantly after learning the first few blocks of data, indicating that global incremental learning plays an important role in addressing this problem. The results presented by Local IL were the worst. Given that LoGID performed better that Global IL, though, we conclude that local incremental learning works well for this problem when combined with global incremental learning. This is similar to what we observed with the Japanese Vowels database.

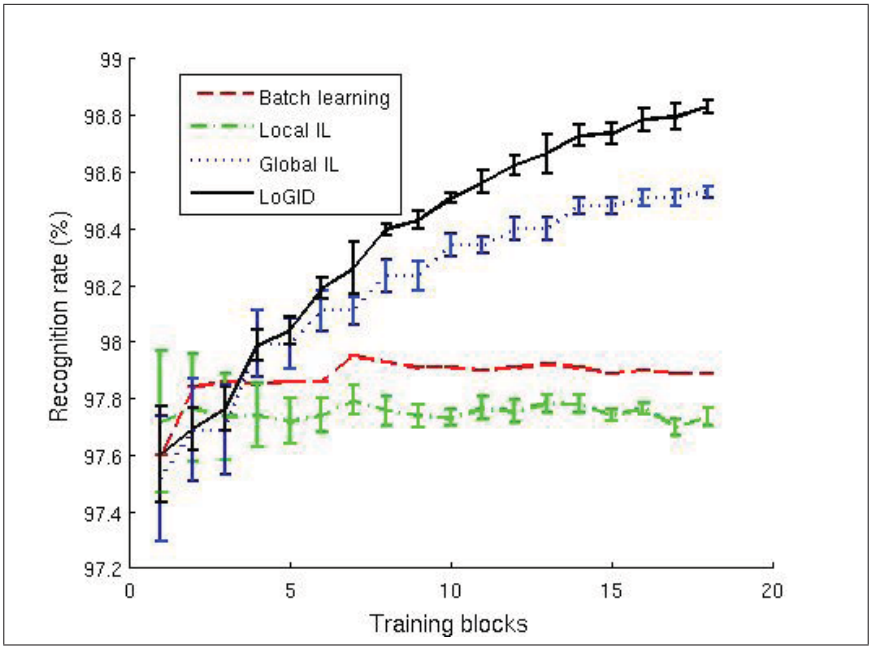



Figure 3.14 Performance comparison for NIST Digits in *test1*.

Figure 3.15 presents the performance comparison for NIST Digits in *test2*. LoGID yields the best final recognition rates, at 96.91%, followed by Local IL, at 94.14%. The latter demonstrated its ability to work well on this problem, yielding the best performance with a small amount of data, i.e. when only the first block is learned. Nonetheless, LoGID surpassed the

performance of Local IL after learning two blocks. This shows that the use of EoCs for incremental learning is promising, and that the adaptation procedure applied by LoGID is capable of improving the use of multiple classifiers even more.

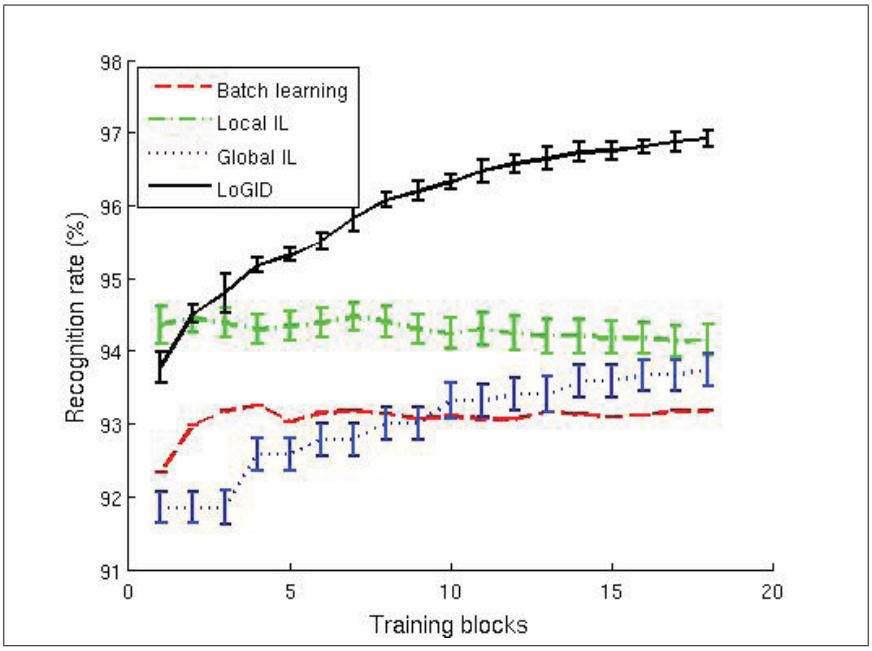



Figure 3.15 Performance comparison for NIST Digits in *test2*.

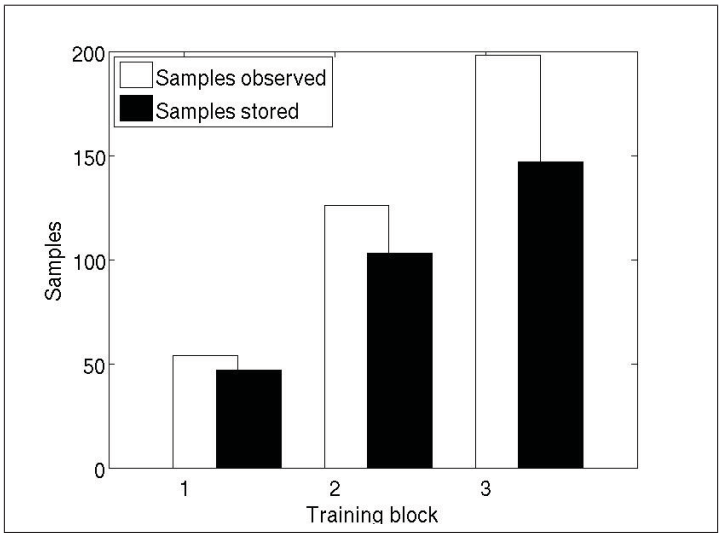



Figure 3.16 Comparison of the number of samples held in $DSel'$ with all the samples observed, on Japanese Vowels.

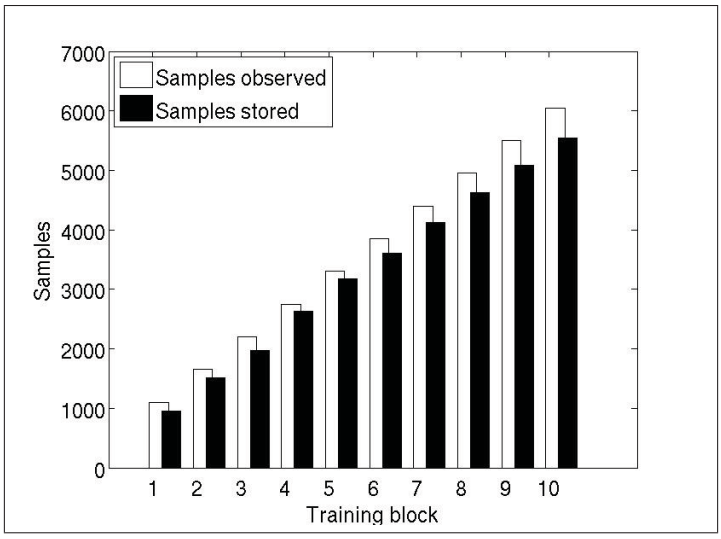



Figure 3.17 Comparison of the number of samples held in $DSel'$ with all the samples observed, on Arabic Digits.

#### 3.4.1.3 Impact of the filtering mechanism on the size of $DSel'$

To demonstrate the impact of the filtering mechanism described in section 3.3.1.2.1, we compare the size of $DSel'$ that results from processing each block of data $D_t$ with the total number of samples observed by the system, i.e. the sum of all samples in $\{D_1, D_2, \ldots, D_t\}$. This comparison is depicted in Figures 3.16, 3.17, 3.18, and 3.19 for Japanese Vowels, Arabic Digits, NIST Letters, and NIST Digits, respectively.

We see that the filtering mechanism works effectively on all databases. On NIST Letters and NIST Digits, after all the training data have been observed, only 13% and 4% of all the samples observed were kept in $DSel'$, respectively. On the Japanese Vowels and Arabic Digits databases, 74.25% and 91.6% of the samples were kept in $DSel'$ respectively. These results show that the mechanism works better when a significant number of samples has been observed. When more training samples are observed, it might be easier for the proposed filtering mechanism to define compact clusters of samples and keep only those samples that are really useful for recognition. Clearly, a larger training set allows for a better estimate of the boundaries in the decision space.

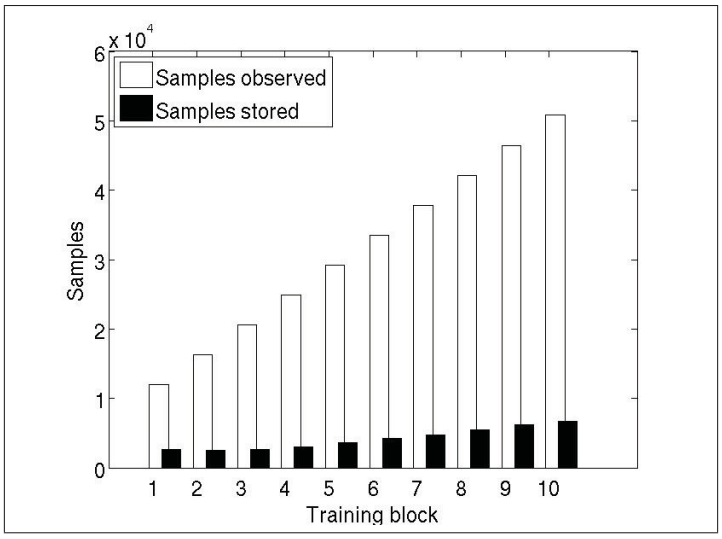



Figure 3.18 Comparison of the number of samples held in *DSel′* with all the samples observed, on NIST Letters.

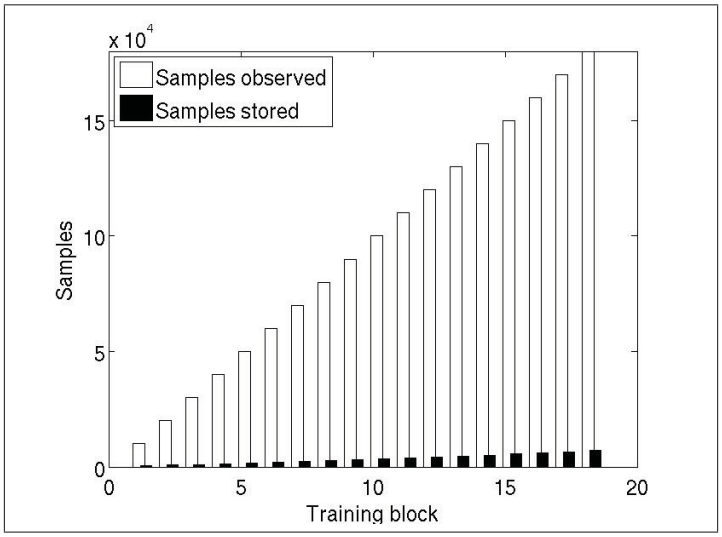



Figure 3.19 Comparison of the number of samples held in *DSel′* with all the samples observed, on NIST Digits.

It is also interesting to note that the use of this mechanism can also successfully replace samples that are no longer considered useful. In Figure 3.18, for example, we see that there is no increase in the size of the dynamic selection set during the learning of blocks 1 to 3. We see, though, that a considerable number of samples is observed and that the performance of the system improves, as shown in Figure 3.13. Therefore, the definition of the zone of relevance

is not only useful for avoiding the excessive growth of the dynamic selection set, but also to define a region that can help re-evaluate previously stored samples.

### 3.4.2 Discussion

In Table 3.3 we present the final recognition rates achieved by the proposed method, the other methods evaluated in this paper, and the results published in the literature. By considering only the methods evaluated in this work, we could claim that the performance of LoGID is promising. These experiments have demonstrated that the LoGID framework can perform better than Learn++, which is a state-of-the-art algorithm for incremental learning (Polikar et al., 2001; Muhlbaier et al., 2009). In this paper, Learn++ corresponds to the Local IL approach. This approach has been outperformed by our framework in all the databases considered here. The results also indicate that LoGID is better than using Global IL alone, represented by the KNOP algorithm. This demonstrates that the full adaptation conducted by the framework, i.e. the combination of both local and global incremental learning, can lead to higher recognition rates than the use of only one of these types of incremental learning. To enrich this overview of performance, we complement these results by including the evaluation of ensembles created with the use of Learn++ on the entire training set (like the AdaBoost algorithm). This evaluation was aimed at comparing LoGID with ensembles created in a batch learning setting. Given that LoGID yielded better recognition rates than these ensembles, the results of which are provided in the Batch/Learn++ column of Table 3.3, we can conclude that the proposed approach can also perform better than both ensembles and single classifiers trained with batch learning.

Table 3.3 A summary of the recognition rates on each database. NA: not available; Other: a discriminant classifier like an Support Vector Machine (SVM), a Neural Network (NN), or a Tree Model (TM). Considering only the methods evaluated in this work, the statistically significant best results (according to Kruskal-Wallis method) are presented in bold. The best results in terms of recognition rates, considering results published in the literature as well, are underlined.

| **Database** | **LoGID** | **Local IL** | **Global IL** | **Batch** | | **Literature** | |
|---|---|---|---|---|---|---|---|
| | | Learn++ | KNOP | Single | Learn++ | HMM | Other |
| Digits *test1* | **98.84 (0.02)** | 97.73 (0.03) | 98.53 (0.02) | 97.88 (-) | 98.41 (0.01) | 98.88 (Ko et al., 2009a) | 99.37 (Milgram et al., 2006) (SVM) |
| Digits *test2* | **96.91 (0.11)** | 94.15 (0.21) | 93.74 (0.22) | 93.18 (-) | 96.00 (0.08) | NA | 97.67 (Radtke et al., 2006) (NN) |
| Letters | **94.10 (0.34)** | 90.25 (0.14) | 90.57 (0.31) | 92.69 (-) | 91.63 (0.40) | 93.24 (Cavalin et al., 2009) | 98.09 (Ciresan et al., 2011) (NN) |
| Arabic | **95.99 (0.57)** | 75.68 (0.11) | 76.54 (0.31) | 90.36 (-) | 86.09 (0.60) | NA | 93.12 (Hammami and Bedda, 2010) (TM) |
| Japanese | **90.43 (0.78)** | 70.54 (1.40) | 73.24 (1.01) | 80.00 (-) | 80.54 (0.70) | NA | NA |

When we consider the results published in the literature, we observe that there is a gap between the recognition rates achieved by LoGID and those achieved by state-of-the-art methods, especially for the NIST database. However, the best results from the literature are presented by systems that consider discriminant classifiers, such as Support Vector Machines and Neural Networks. These classifiers are trained in batch learning settings, considering the entire training database for setting parameters. LoGID is used in an incremental learning setting, and its parameters are computed by taking into account only the first block of data (a very small training set), and so it is not easy to compare our results with the results of those classifiers since both the type of base classifier and the learning setting are different. The comparison is fairer, though, if we consider only HMM-based methods. On the NIST Digits database, the recognition rates yielded by LoGID are very close to those of the best HMM-based methods (Ko et al., 2009a,b). On the $test1$ set, the latter methods present recognition rates of about 98.88% and 98.86%, respectively, while LoGID achieves about 98.84%. This indicates that our approach has been able to move very close to the upper bound of NIST Digits database. In other words, the performance of LoGID was similar to that of the best batch learning methods. Furthermore, LoGID presents the best recognition rates on the remaining two databases, which indicates that the proposed adaptive framework can perform even better than the best state-of-the-art methods on various pattern recognition problems.

## 3.5 Conclusion

In this paper we proposed the LoGID approach, which consists of a framework for the adaptation of a pool of base classifiers during two phases: learning and generalization. During generalization, the KNOP algorithm considers a set of output profiles to select the best classifiers for recognizing each test sample. By considering the Learn++ algorithm to generate a set of diverse members to update the current pool of classifiers, we defined the local incremental learning module. Global incremental learning is achieved by updating the dynamic selection set, and the corresponding output profiles, used by the KNOP algorithm.

Experiments have been carried out on four different databases. These databases consist of two handwriting recognition problems, i.e. the recognition of isolated digits and isolated uppercase letters, as well as two speech recognition problems. The results demonstrated that LoGID can effectively take advantage of the data presented in both the learning and the generalization phases to better generate and use a pool of classifiers. The base classifiers controlled by LoGID achieve better performance than the classifiers generated and used by other types of approach, such as batch learning and static selection. In addition, the mechanism proposed to control the increase in memory required by the baseline system have proved to be useful in most problems.

Future work might involve the validation of LoGID with other classifiers. As demonstrated by some experiments, LoGID has been able to create a pool of HMMs that could surpass the upper bound reached by the best HMM-based systems in the literature. Consequently, it may be of interest to try other types of base classifiers on problems where we know HMMs may not be the best choice as the base classifier. In addition, we should also focus on reducing the overall complexity of the KNOP algorithm. Some ideas proposed to reduce the complexity of instance-based classifiers are useful in this regard (Cui et al., 2005).

# CONCLUSION

This thesis has focused on the design of adaptive systems (AS), the main goal of which is to adapt a baseline pattern recognition system to different conditions in both the learning and generalization phases. Such adaptations are achieved through the observation of data that become available over time. For this purpose, we presented a series of investigations related to both incremental learning (IL) and dynamic selection (DS), and to the integration of both into a single framework, called LoGID. These investigations resulted in the contributions described below.

The first contribution consists of an evaluation of IL algorithms for HMMs applied for the recognition of alphanumeric characters. This study demonstrates the effectiveness of ensemble of classifier (EoC)-based algorithms in IL settings. The ensembles are able to yield a performance that is comparable to that of batch learning (BL) algorithms. In addition, we empirically demonstrate the benefits of saving samples to a short-term memory, represented by the validation set. This memory is useful both for improving the recognition performance and for avoiding the use of useless classifiers, which may negatively affect recognition time.

Then, we presented various investigations related to DS algorithms, the main goal being to evaluate algorithms that could be integrated with EoC-based IL algorithms, pursuant to our previous research on IL. We focused on methods that could select classifiers by evaluating only their outputs, so that these algorithms would be suitable for a broad range of classifiers. This work resulted in the proposal of the DMO concept, and two implementations for it: the methods denoted DSA$^m$ and DSA$^c$. Experimental evaluations on several databases demonstrate that the DMO concept is promising, since both DSA$^m$ and DSA$^c$ have been able to perform better than other approaches in the literature. The latter presented the best recognition rates, owing to its use of output profiles. Furthermore, DSA$^c$ is IL-ready, and is suitable for composing an AS. Also, we have demonstrated that DS might be preferred over static methods such as SVMs and MLP when the degree of recognition uncertainty is high, due, for instance, to the use of small-size training sets.

Finally, we focused on the full definition of an AS by proposing the LoGID framework. During the generalization phase, we proposed the use of the KNOP algorithm to select an EoC based on finding the output profiles that are the most similar to the test sample output profile, the former being computed from a DS set. This scheme allows two types of IL to be conducted during the learning phase: local and global. The former consists of updating the pool of classifiers, by incorporating into it new members trained with the current block of data and by removing the least relevant ones. The latter involves the addition of the most relevant output profiles, also computed from the current block of data, to the set of output profiles used by KNOP. Experiments on a diverse set of databases demonstrate the effectiveness of the proposed framework. In most problems, the approach has been able to reach or surpass the upper bound of the best HMM-based methods in the literature. Moreover, the individual evaluation of local and global IL has shown the effectiveness of combining the two schemes to improve the overall recognition performance of the system.

**Future Work**

In future work, several directions can be followed, among them:

- *Evaluation of the use of other types of classifiers with LoGID*. Although in this thesis we present a case study on HMMs, this framework is general and can be adapted to other types of classifiers. So, it would be interesting to evaluate the behavior of such a framework with other configurations. With classifiers such as SVMs and MLPs, we investigate how closely the performance of LoGID can approach the upper bound of some databases, for instance, the NIST Digits and NIST Letters databases.
- *Performance improvement*. It will be important to pursue our investigation on methods that could improve the recognition performance of the proposed framework. In the case of dynamic selection, for example, we could look at ways to better use the information provided by output profiles, such as additional similarity measures. We could also study different ways in which EoCs can be dynamically defined.

- *Reduction of complexity*. In the proposed LoGID framework, both the learning and generalization phases are demanding in terms of computation time. Consequently, an effort should be made to reduce this complexity. Indexing the output profile set may be one way to achieve this.
- *Evaluation of other schemes for pruning classifiers and output profiles*. Such pruning is directly related to the previous topic. Good pruning results in complexity reduction without performance degradation. In this regard, the use of zones of relevance has been shown to be promising for selecting of the most relevant output profiles. One interesting way to enhance this mechanism would be the dynamic definition of zones of relevance. This would lead to the definition of a mechanism that adapts based on the data that are observed. In terms of the pool of classifiers, a better approach could be global evaluation of the pool, as a global view might provide a better idea of which classifiers are really useful and which are not.

# APPENDIX I

# THE IMPACT OF THE SIMILARITY MEASURE ON THE DSA$^C$ APPROACH

In this appendix, we present our evaluation of additional similarity measures (SMs) for the DSA$^c$ approach (see Chapter 2). These measures are used to compare the level of similarity between two output profiles. The main idea of this research is to verify the impact of each SM on the recognition performance achieved by DSA$^c$.

In our evaluation, we implemented three different versions of DSA$^c$, each using a different SM. One version considered template matching as the SM, as originally proposed in Chapter 2. The other two considered the Euclidean distance and the proposed oracle-based template matching, respectively. For a better understanding of the three SMs, see Section I.1.

All three versions of DSA$^c$ were experimentally evaluated using the protocol described in Section 2.5. The results are described and discussed in Section I.2.

## I.1 Similarity measures (SMs)

From here on, we use the following additional notations: $\tilde{x}_{i,test,k}$ and $\tilde{x}_{j,val,k}$ represent the output of the $k$th classifier for $x_{i,test}$ and $x_{j,val}$, respectively. In addition, for each $x_{j,val}$, the set of flags $CC_j = \{cc_{j,1}, cc_{j,2}, \ldots, cc_{j,W}\}$, where each $cc_{j,k}$ is a binary value, shows whether or not $C'_k$ has correctly classified $x_{j,val}$. In other words, $cc_{j,k} = 1$ if $C'_k$ correctly classifies $x_{j,val}$, otherwise, $cc_{j,k} = 0$.

The three different SMs we consider are described below. Note that they are individually used by $\zeta$, to perform step 4 in Algorithm 2.2.

### I.1.1 Euclidean distance (ED)

This SM simply computes the Euclidean distance between the output profile of $\tilde{x}_{i,test}$ and each $\tilde{x}_{j,val}\ \forall j$. To implement this measure, we have to minimize the following equation:

$$ED_{i,j} = \sqrt{\sum_{k=1}^{N} (\tilde{x}_{i,test,k} - \tilde{x}_{j,val,k})^2} \tag{I.1}$$

### I.1.2 Template matching (TM)

In considering two output profiles, this SM computes how many classifiers provide exactly the same output. Given that a class index generally has no mathematical meaning in pattern recognition problems, this SM might be more accurate than ED because it computes only the number of identical outputs. We can implement this measure by maximizing Equation I.2, which depends on Equation I.3.

$$TM_{i,j} = \frac{\sum_{k=1}^{N} \alpha_{i,j,k}}{N} \tag{I.2}$$

$$\alpha_{i,j,k} = \begin{cases} 1, & \text{if } \tilde{x}_{i,test,k} = \tilde{x}_{j,val,k} \\ 0, & \text{otherwise} \end{cases} \tag{I.3}$$

### I.1.3 Oracle-based template matching (OTM)

In considering that each $\tilde{x}_{j,val}$ is related to the correct class label $correct_{j,val}$, we compute the number of classifiers that produce the correct class label for $\tilde{x}_{j,val}$ and provide the same output as $\tilde{x}_{i,test}$. In this case, only the classifiers that produce the correct output for the validation sample are taken into account. Equation I.4, which has to be maximized, computes this SM mathematically.

$$OTM_{i,j} = \frac{\sum_{k=1}^{N} \beta_{j,i,k}}{\sum_{k=1}^{N} \gamma_{j,k}} \tag{I.4}$$

$$\beta_{i,j,k} = \begin{cases} 1, & \text{if } \tilde{x}_{i,test,k} = \tilde{x}_{j,val,k} \text{ and } \tilde{x}_{j,val,k} = correct_{j,val} \\ 0, & \text{otherwise} \end{cases} \tag{I.5}$$

$$\gamma_{j,k} = \begin{cases} 1, & \text{if } \tilde{x}_{j,val,k} = correct_{j,val} \\ 0, & \text{otherwise} \end{cases} \tag{I.6}$$

These SMs result in three different versions of DSA$^c$:

A. DSA$^c_{ED}$, where $\delta_{i,j} = 1 - \text{ED}_{i,j}$;

B. DSA$^c_{TM}$, where $\delta_{i,j} = \text{TM}_{i,j}$;

C. DSA$^c_{OTM}$, where $\delta_{i,j} = \text{OTM}_{i,j}$.

## I.2 Experiments

In this section, we present our evaluation of the different versions of DSA$^c$, as described in the previous section, considering the experimental protocol presented in Section 2.5. This protocol takes into account seven distinct databases, divided into both small and large datasets, and two different classifiers: 1-Nearest Neighbors (1NN) and Decision Trees (DTrees).

In Table I.1 we present the results on the small datasets, considering 1NN classifiers as the base classifier. With this configuration, DSA$^c_{OTM}$ achieved the lowest error rates on three out of the five small databases: Dna, with 5.22%; Satellite, with 7.01%; and Ship, with 7.82%. In contrast, DSA$^c_{ED}$ achieved the best results for the Feltwell database, with an error rate of 8.85%, and DSA$^c_{TM}$ was the best method for the Texture database, yielding an error rate of 0.56%. These results demonstrate that oracle-based template matching may be a better alternative as an SM in this context, since the version of DSA$^c$ with this measure presented the lowest error rates on the majority of the databases. However, the other measures may also result in the best performance depending on the problem.

The error rates achieved with the evaluation of small datasets with DTree classifiers are presented in Table I.2. DSA$^c_{OTM}$ yielded the lowest error rates on the Feltwell and Texture databases, with 10.11% and 0.89%, respectively. DSA$^c_{ED}$ was the best method on the Satellite and Ship databases, yielding error rates of 6.89% and 5.51%, respectively. Finally, on the Dna

Table I.1 Error rates on small datasets using 1NN classifiers, at zero-level rejection. Results in bold present the best approach among static MO, DSA, DT, and the proposed DSA$^m$ and DSA$^c$, with $K$ set to 30. The underlined results represent the statistically significant best method. Marked with asterisk (*) are the proposed approaches. Between parentheses is the variance of each approach ($\times 10^{-2}$)

| **Method** | **Dna** | **Felt** | **Sat** | **Ship** | **Text** |
|---|---|---|---|---|---|
| *Static selection* | | | | | |
| Oracle $C$ | 0.03 (-) | 0.67 (-) | 0.36 (-) | 0.28 (-) | 0.04 (-) |
| All features | 26.30 (-) | 12.35 (-) | 9.84 (-) | 11.24 (-) | 1.13 (-) |
| Best from $C$ | 23.10 (-) | 9.46 (-) | 8.95 (-) | 10.26 (-) | 0.62 (-) |
| MV all $C$ | 6.87 (-) | 10.44 (-) | 8.59 (-) | 9.94 (-) | 1.11 (-) |
| Best from $C^{*'}$ | 9.14 (2.57) | 9.37 (4.39) | 8.19 (8.39) | 9.41 (2.75) | 0.71 (3.02) |
| DT$_{TM}$ $C$ | 8.53 (-) | 14.76 (-) | 8.97 (-) | 10.03 (-) | 4.56 (-) |
| DT$_{OTM}$ $C$ | 7.11 (-) | 12.89 (-) | 9.48 (-) | 10.16 (-) | 0.67 (-) |
| DT$_{ED}$ $C$ | 12.70 (-) | 14.75 (-) | 9.27 (-) | 11.27 (-) | 4.88 (-) |
| MO | 5.70 (1.89) | 9.74 (22.49) | 8.01 (0.55) | 9.00 (4.93) | 0.98 (0.15) |
| *Dynamic selection* | | | | | |
| Oracle $C^{*'}$ | 1.12 (0.75) | 6.06 (19.05) | 3.98 (0.70) | 3.92 (1.96) | 0.40 (0.06) |
| DSA | 10.47 (10.10) | 10.76 (24.02) | 9.17 (0.99) | 11.21 (12.12) | 1.03 (0.12) |
| $*$DSA$^m$ | 5.57 (1.77) | 9.35 (21.39) | 7.61 (0.77) | 8.80 (5.30) | 0.93 (0.14) |
| *Versions of DSA$^c$* | | | | | |
| $*$**DSA$^c_{TM}$** | 5.46 (0.07) | 8.93 (0.09) | 7.42 (0.10) | 8.10 (0.15) | **0.56** (0.01) |
| $*$**DSA$^c_{OTM}$** | **5.22**(0.06) | 8.90 (0.08) | **7.01** (0.05) | **7.82** (0.19) | 0.60 (0.01) |
| $*$**DSA$^c_{ED}$** | 8.52 (0.01) | **8.85** (0.12) | 7.49 (0.14) | 8.11 (0.16) | 0.81 (0.02) |

database, the best performance was achieved by DSA$^c_{TM}$, producing error rates of 3.05%. The difference between the best two results, though, is very small in some cases. For example, on the Ship database, the error rates achieved by DSA$^c_{ED}$ are only 0.01% higher than those of DSA$^c_{TM}$. Thus, despite this difference, we can say that the two methods performed similarly on that database. Consequently, with this type of classifier, we observe a behavior that is similar to that observed with 1NN classifiers. That is, there is no single measure that we can assume will be the best for these databases. The measure has to be defined based on the problem and on the configuration for DSA$^c$.

The evaluation for large databases is presented Table I.3. In this case, DSA$^c_{TM}$ yielded the lowest error rates on the NIST Digits databases, considering all types of base classifiers and

Table I.2 The same error rate evaluations as in Table I.1, but with DTrees

| **Method** | **Dna** | **Felt** | **Sat** | **Ship** | **Text** |
|---|---|---|---|---|---|
| *Static selection* | | | | | |
| Oracle $C$ | 0.03 (-) | 0.60 (-) | 0.22 (-) | 0.24 (-) | 0.02 (-) |
| All features | 6.85 (-) | 16.81 (-) | 14.17 (-) | 10.92 (-) | 7.56 (-) |
| Best from $C$ | 11.33 (-) | 11.86 (-) | 11.83 (-) | 10.45 (-) | 6.07 (-) |
| MV all $C$ | 5.05 (-) | 11.86 (-) | 8.64 (-) | 6.80 (-) | 2.56 (-) |
| Best from $C^{*l}$ | 5.71 (1.70) | 10.22 (4.45) | 8.35 (1.02) | 7.02 (2.54) | 2.04 (6.77) |
| $\text{DT}_{TM}$ $C$ | 4.53 (-) | 13.93 (-) | 8.96 (-) | 7.74 (-) | 1.34 (-) |
| $\text{DT}_{OTM}$ $C$ | 5.80 (-) | 14.02 (-) | 10.94 (-) | 7.94 (-) | 1.19 (-) |
| $\text{DT}_{ED}$ $C$ | 5.39 (-) | 14.13 (-) | 9.16 (-) | 8.65 (-) | 2.30 (-) |
| MO | 4.02 (1.02) | 11.20 (25.09) | 7.76 (0.61) | 6.20 (4.21) | 2.20 (0.58) |
| *Dynamic selection* | | | | | |
| Oracle $C^{*l}$ | 1.07 (0.85) | 5.82 (15.52) | 3.78 (0.62) | 3.18 (3.09) | 0.81 (0.06) |
| DSA | 7.55 (6.11) | 12.52 (27.90) | 10.29 (4.69) | 10.16 (19.21) | 2.42 (0.67) |
| $\text{DSA}^{m}$ | 4.07 (1.15) | 10.77 (23.28) | 7.42 (0.58) | 5.89 (3.34) | 2.13 (0.61) |
| *Versions* $\text{DSA}^{c}$ | | | | | |
| $*\textbf{DSA}^{c}_{TM}$ | **3.05** (0.12) | 10.32 (0.17) | 7.11 (0.09) | 5.52 (0.21) | 1.11 (0.03) |
| $*\textbf{DSA}^{c}_{OTM}$ | 3.59 (0.11) | **10.11** (0.12) | 6.93 (0.10) | 5.66 (0.22) | **0.89** (0.01) |
| $*\textbf{DSA}^{c}_{ED}$ | 3.12 (0.13) | 10.24 (0.24) | **6.89** (0.11) | **5.51** (0.27) | 1.86 (0.09) |

ensemble generation methods. The error rates on *test1* considering 1NN with RSS, DTree with RSS, and DTree with Bagging were: 2.37%, 1.76%, and 2.98%, respectively. These results were followed by those yielded by $\text{DSA}^{c}_{ED}$, the error rates of which, in the same order, were: 2.43%, 4.64%, and 3.98%. On *test2*, the error rates yielded by $\text{DSA}^{c}_{TM}$, in the same order, were: 5.34%, 4.36%, and 6.17%. On the NIST Letters database, $\text{DSA}^{c}_{OTM}$ achieves the best performance, also considering all types of base classifiers and ensemble generation methods. The error rates yielded by that method were: 4.10%, 3.98%, and 5.36%. Note that $\text{DSA}^{c}_{TM}$ presented the second best results with DTree classifiers, while $\text{DSA}^{c}_{ED}$ was the second best method with 1NN classifiers. These results indicate that on larger databases we may observe some stability regarding the choice of the SM. The same measure may be the best choice for various configurations of $\text{DSA}^{c}$. This stability is likely to be a result of the use of larger training sets.

Table I.3 The same error rate evaluations as in Table I.1, but considering both 1NN and DTrees with large datasets. The variance in this case was multiplied by $10^{-3}$. In addition, we present the evaluation of DTree classifiers created by bagging

| **Classifier** | **1NN - RSS** | | | **DTree - RSS** | | | **DTree - Bagging** | | |
|---|---|---|---|---|---|---|---|---|---|
| **Method** | **Digits** | | **Letters** | **Digits** | | **Letters** | **Digits** | | **Letters** |
| | *test1* | *test2* | | *test1* | *test2* | | *test1* | *test2* | |
| *Static selection* | | | | | | | | | |
| Oracle $C$ | 0.05 (-) | 0.17 (-) | 0.18 (-) | 0.01 (-) | 0.04 (-) | 0.04 (-) | 0.24 (-) | 0.63 (-) | 0.29 (-) |
| All features | 6.66 (-) | 9.76 (-) | 7.82 (-) | 11.07 (-) | 18.20 (-) | 13.50 (-) | 6.66 (-) | 9.76 (-) | 7.82 (-) |
| Best from $C$ | 7.52 (-) | 13.99 (-) | 14.47 (-) | 10.30 (-) | 19.18 (-) | 17.13 (-) | 9.70 (-) | 16.62 (-) | 14.31 (-) |
| MV all $C$ | 3.72 (-) | 8.10 (-) | 6.60 (-) | 2.92 (-) | 6.67 (-) | 6.06 (-) | 5.65 (-) | 10.99 (-) | 7.63 (-) |
| Best from $C^{*^I}$ | 3.60 (3.83) | 7.77 (7.74) | 6.56 (6.73) | 2.98 (4.98) | 6.77 (1.12) | 6.21 (7.82) | 5.31 (0.00) | 10.28 (0.00) | 7.62 (0.00) |
| $DT_{TM}$ $C$ | 2.55 (-) | 5.74 (-) | 4.95 (-) | 2.00 (-) | 5.00 (-) | 4.64 (-) | 3.65 (-) | 7.65 (-) | 6.49 (-) |
| $DT_{OTM}$ $C$ | 4.74 (-) | 9.74 (-) | 7.56 (-) | 2.70 (-) | 6.03 (-) | 7.15 (-) | 7.24 (-) | 12.86 (-) | 8.55 (-) |
| $DT_{ED}$ $C$ | 2.97 (-) | 6.57 (-) | 6.55 (-) | 2.56 (-) | 6.26 (-) | 7.44 (-) | 3.85 (-) | 8.20 (-) | 7.62 (-) |
| MO | 3.51 (0.04) | 7.63 (0.18) | 6.33 (0.32) | 2.85 (0.03) | 6.53 (0.14) | 6.00 (0.03) | 5.35 (0.00) | 10.41 (0.01) | 7.54 (0.01) |
| *Dynamic selection* | | | | | | | | | |
| Oracle $C^{*^I}$ | 1.97 (0.02) | 4.59 (1.14) | 3.87 (4.42) | 1.87 (1.03) | 4.39 (4.36) | 4.53 (2.49) | 3.72 (0.00) | 7.42 (0.00) | 4.68 (0.00) |
| DSA | 3.61 (0.08) | 7.87 (0.17) | 6.43 (0.48) | 2.87 (0.06) | 6.61 (0.29) | 6.06 (0.41) | 5.33 (0.00) | 10.45 (0.00) | 7.11 (0.00) |
| $DSA^m$ | 3.45 (0.05) | 7.53 (0.16) | 6.12 (0.43) | 2.72 (0.18) | 6.26 (0.59) | 5.83 (0.38) | 5.10 (0.00) | 9.96 (0.00) | 7.23 (0.00) |
| *Versions of $DSA^c$* | | | | | | | | | |
| $*\mathbf{DSA}^c_{TM}$ | **2.37** (0.02) | **5.34** (0.04) | 4.62 (0.16) | **1.76** (0.02) | **4.36** (0.04) | 4.20 (0.05) | **2.98** (0.00) | **6.17** (0.00) | 5.58 (0.00) |
| $*\mathbf{DSA}^c_{OTM}$ | 2.63 (0.03) | 5.88 (0.17) | **4.10** (0.25) | 2.16 (0.03) | 4.96 (0.08) | **3.89** (0.06) | 3.03 (0.00) | 6.43 (0.00) | **5.36** (0.00) |
| $*\mathbf{DSA}^c_{ED}$ | 2.43 (0.03) | 5.43 (0.16) | 4.39 (0.28) | 1.83 (0.05) | 4.64 (0.10) | 4.32 (0.09) | 3.98 (0.00) | 7.94 (0.00) | 5.74 (0.00) |

### I.2.1 Discussion

The results presented in this section reveal some interesting clues about how to improve the DSA$^c$ performance. We have demonstrated that choosing an adequate SM may have an impact on the recognition performance of this approach. In this regard, we observe that there is no single SM that works best for all the databases considered in this work. As a consequence, the evaluation of different measures might be important for optimizing the DSA$^c$ approach for each recognition problem, depending on the base classifier and the ensemble generation method.

Nonetheless, we observe that some stability can be gained with larger databases, so that a single SM may present the same behavior across different configurations for DSA$^c$. In contrast, on the small datasets such stability is not evident since for each of these databases the best method with 1NN and DTree classifiers is generally not the same. On the large databases, though, a single SM seems to work best for all types of base classifiers and ensemble generation methods. We can see this on both the NIST Digits and NIST Letters databases, where the use of template matching as an SM always produces the best performance on the first, and oracle-based template matching always results in the best performance on the second, independently of the type of base classifier or the type of ensemble generation method used.

## I.3 Conclusion and Future Work

In this appendix, we presented our evaluation of alternative SMs to be used in the DSA$^c$ approach. These measures are designed for the computation of the degree of similarity between two different output profiles. In addition to template matching, which was proposed and evaluated in Chapter 2, we also evaluated the use of Euclidean distance and oracle-based template matching SMs. As a result, three different versions of DSA$^c$ were proposed, each using one of these SMs: DSA$^c_{ED}$, DSA$^c_{TM}$, and DSA$^c_{OTM}$.

Experiments conducted on small and large databases, and considering various types of base classifiers and ensemble generation methods, allowed us to observe the behavior of each version of DSA$^c$ under varied conditions. The results demonstrate that the use of both the Eu-

clidean distance and the oracle-based template matching SMs might result in a positive impact on the performance of DSA$^c$. Defining the best measure is dependent on the database, and, in some cases, on the base classifier and ensemble generation method as well. On large databases, though, we observe that a single SM may be defined to be used with different DSA$^c$ configurations, owing to the stability gained after large training sets have been processed.

Given the results reported in this appendix, we believe that pursuing these investigations on alternative methods for computing similarity is promising. In this context, other measures could be evaluated using the same idea as presented here, that is, the definition of a different version of DSA$^c$ for each measure. However, it might be also interesting to investigate other mechanisms for computing similarity. For example, inspired by the idea of multiple classifier systems, combining multiple SMs could take advantage of the diversity introduced by these different measures, resulting in a better estimation of the degree of similarity between two output profiles.

## BIBLIOGRAPHY


Baldi, P. and Y. Chauvin, 1994. Smooth on-line learning algorithms for hidden Markov models. *Neural Computation*, 6(2):179–190.

Batista, L., E. Granger, and R. Sabourin, 2011. Dynamic ensemble selection for off-line signature verification. *10th International Workshop on Multiple Classifier Systems*.

Bertolini, D., L.S. Oliveira, E. Justino, and R. Sabourin, 2010. Reducing forgeries in writer-independent off-line signature verification through ensemble of classifiers. *Pattern Recognition*, 43(1):387–396.

Britto, A. S., R. Sabourin, F. Bortolozzi, and C. Y. Suen, 2003. Recognition of numeral strings using a two-stage HMM-based method. *International Journal on Document Analysis and Recognition*, 5(2):102–117.

Britto, A. S., 2001. *A Two-Stage HMM-Based Method For Recognizing Handwritten Numeral Strings*. PhD thesis, Pontifícia Universidade Católica do Paraná.

Brown, G., J. Wyatt, R. Harris, and X. Yao, 2005. Diversity creation methods: A survey and categorization. *Information Fusion*, 6(1):5–20.

Cavalin, P. R., A. S. Britto, F. Bortolozzi, R. Sabourin, and L. E. S. Oliveira, 2006. An implicit segmentation-based method for recognition of handwritten strings of characters. *The 21st Annual ACM Symposium on Applied Computing*, p. 836–840, Dijon, France.

Cavalin, P. R., R. Sabourin, C. Y. Suen, and A. S. Britto Jr, 2008. Evaluation of incremental learning algorithms for an HMM-based handwritten isolated digits recognizer. *Proceedings of the 11th International Conference on Frontiers in Handwriting Recognition (ICFHR 2008)*, p. 1–6, Montreal, Canada.

Cavalin, P. R., R. Sabourin, C. Y. Suen, and A. S. Britto Jr., 2009. Evaluation of incremental learning algorithms for HMM in the recognition of alphanumeric characters. *Pattern Recognition*, 42(12):3241–3253. New Frontiers in Handwriting Recognition.

Cavalin, P. R., R. Sabourin, and C. Y. Suen, 2010. Dynamic selection of ensembles of classifiers using contextual information. *9th International Workshop on Multiple Classifier Systems (MCS 2010)*, p. 145–154, Cairo, Egypt.

Cavalin, P. R., R. Sabourin, and C. Y. Suen, 2011a. Dynamic selection approaches for multiple classifiers systems (accepted for publication). *Neural Computing and Applications*.

Cavalin, P. R., R. Sabourin, and C. Y. Suen, 2011b. LoGID: An adaptive framework combining local and global incremental learning for dynamic selection of ensembles of HMMs (submitted). *Pattern Recognition*.

Cheriet, M., M. El Yacoubi, H. Fujisawa, D. Lopresti, and G. Lorette, 2009. Handwriting recognition research: Twenty years of achievement... and beyond. *Pattern Recognition*, 42(12):3131–3135. New Frontiers in Handwriting Recognition.

Chien, J.-T., C-H. Lee, and H-C. Wang, Jun 1997. A hybrid algorithm for speaker adaptation using map transformation and adaptation. *Signal Processing Letters, IEEE*, 4(6):167–169.

Ciresan, D. C., U. Meier, L. M. Gambardella, and J. Schmidhuber, 2011. Convolutional neural network committees for handwritten character classification. *Proceedings of the 11th International Conference on Document Analysis and Recognition (ICDAR 2011)*.

Cui, B., B. C. Ooi, J. Su, and K-L. Tan, 2003. Contorting high dimensional data for efficient main memory KNN processing. *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, p. 479–490, San Diego, USA.

Cui, B., H. T. Shen, J. Shen, and K-L. Tan, 2005. Exploring bit-difference for approximate KNN search in high-dimensional databases. *Proceedings of the 16th Australasian database conference*, p. 165–174. Australian Computer Society, Inc.

Davis, R. I. A. and B. C. Lovell, 2003. Comparing and evaluating HMM ensemble training algorithms using train and test and condition number criteria. *Pattern Analysis Application*, 6:327–336.

Digalakis, V. V., 1999. Online adaptation of hidden Markov models using incremental estimation algorithms. *Speech and Audio Processing*, 7(3):253–261.

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

Dos Santos, E. M., R. Sabourin, and P. Maupin, 2006. Single and multi-objective genetic algorithms for the selection of ensemble of classifiers. *Proceedings of International Joint Conference on Neural Networks, 2006*, p. 3070–3077, Vancouver, Canada.

Dos Santos, E. M., R. Sabourin, and P. Maupin, 2008. A dynamic overproduce-and-choose strategy for the selection of classifier ensembles. *Pattern Recognition*, 41:2993–3009.

Duda, R. O., P. E. Hart, and D. G. Stork, 2000. *Pattern Classification*. Wiley-Interscience Publication.

Florez-Larrahondo, G., 2005. *Incremental Learning of Discrete hidden Markov Models.* PhD thesis, Mississipi State University.

Florez-Larrahondo, G., S. Bridges, and E. A. Hansen, 2005. Incremental estimation of discrete hidden Markov models based on a new backward procedure. *National Conference on Artificial Intelligence*, p. 758–763.

Frank, A. and A. Asuncion, 2010. UCI machine learning repository. URL http://archive.ics.uci.edu/ml.

Gangardiwala, A. and R. Polikar, 2005. Dynamically weighted majority voting for incremental learning and comparison of three boosting based approaches. *Proceedings of the International Joint Conference on Neural Networks*, p. 1131–1136.

Giacinto, G. and F. Roli, 2001. Dynamic classifier selection based on multiple classifier behaviour. *Pattern Recognition*, 34:1879–1881.

Gotoh, Y., M. M. Hochberg, and H. F. Silverman, 1998. Efficient training algorithms for HMM's using incremental estimation. *Speech and Audio Processing*, 6(6):539–548.

Gunter, S. and H. Bunke, 2004. Combination of three classifiers with different architectures for handwritten word recognition. *Proceedings of the 9th International Workshop on Frontiers in Handwriting Recognition (IWFHR-9 2004)*.

Hammami, N. and M. Bedda, 2010. Improved tree model for Arabic speech recognition. *3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT)*, vol. 5, p. 521–526.

Hansen, L. K., C. Liisberg, and P. Salamon, 1997. The error-reject tradeoff. *Open Systems & Information Dynamics*, 4(2):159–184.

Ho, T., 1998. The random subspace method for construction decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:832–844.

Kapp, M., R. R. Sabourin, and P. Maupin, 2010. Adaptive incremental learning with an ensemble of support vector machines. *20th International Conference on Pattern Recognition*, p. 4048–4051, Instanbul, Turkey.

Khreich, W., E. Granger, A. Miri, and R. Sabourin, 2012. Adaptive ROC-based ensembles of HMM applied to anomaly detection. *Pattern Recognition*, 45:208–230.

Kim, M.-S., D. Kim, and S-Y. Lee, 2003. Face recognition using the embedded HMM with second-order block-specific observations. *Pattern Recognition*, 36(11):2723–2735.

Ko, A. H., R. Sabourin, and A. S. Britto, 2007. A new HMM-based ensemble generation method for character recognition. *7th International Workshop on Multiple Classifier Systems*, Prague, Czech Republic.

Ko, A. H., R. Sabourin, and A. S. Britto Jr, 2008. From dynamic classifier selection to dynamic ensemble selection. *Pattern Recognition*, 41(5):1718–1731.

Ko, A. H., P. R. Cavalin, R. Sabourin, and A. S. Britto Jr, 2009a. Leave-one-out-training and leave-one-out-testing hidden Markov models for a handwritten numeral recognizer: the implication of a single classifier and multiple classifications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. 2168–2178.

Ko, A. H., R. Sabourin, and A. S. Britto Jr, 2009b. Ensemble of HMM classifiers based on the clustering validity index for a handwritten numeral recognizer. *Pattern Anal. Appl.*, 12: 21–35.

Kuncheva, L. I., 2000. Cluster-and-selection model for classifier combination. *Proceedings of the International Conference on Knowledge Based Intelligent Engineering Systems and Allied Technologies*, p. 185–188, Brighton, UK.

Kuncheva, L. I., J. C. Bezder, and R. P. W. Duin, 2001. Decision templates for multiple classifier fusion: an experimental comparison. *Pattern Recognition*, 34:299–314.

Kuncheva, L. I., C. J. Whitaker, C.A. Shipp, and R.P.W. Duin, 2003. Limits on the majority vote accuracy in classifier fusion. *Pattern Analysis and Applications*, 6(1):22–31.

Lu, Y., H. Wu, L. Zhou, and Z. Wu, 2010. Multi-environment model adaptation based on vector taylor series for robust speech recognition. *Pattern Recognition*, 43(9):3093–3099.

Mackay, D. J. C., 1997. Ensemble learning for hidden Markov models. Technical report, Cavendish Laboratory, Cambridge University, UK.

Milgram, J., 2007. *Contribution à l'intégration des machines à vecteurs de support au sein de systèmes de reconnaissance de formes: application à la lecture automatique de l'écriture manuscrite*. PhD thesis, École de Technologie Supérieure.

Milgram, J., M. Cheriet, and R. Sabourin, 2006. "One against one" or "One against all": Which one is better for handwriting recognition with SVMs? *Proceedings of 10th International Workshop on Frontiers in Handwriting Recognition*, La Baule, France.

Mizuno, J., T. Watanabe, K. Ueki, K. Amano, E. Takimoto, and A. Maruoka, 2000. Online estimation of hidden Markov model parameters. *Third International Conference Discovery Science (DS 2000)*, vol. 1967, p. 155–169.

Mongillo, G. and S. Deneve, 2008. Online learning with hidden Markov models. *Neural Computation*, 20:1706–1716.

Muhlbaier, M. D., A. Topalis, and R. Polikar, 2009. Learn++.NC: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes. *IEEE Transactions on Neural Networks*, 20(1):152–168.

Najkar, N., F. Razzazi, and H. Sameti, 2010. A novel approach to HMM-based speech recognition systems using particle swarm optimization. *Mathematical and Computer Modelling*, 52(11-12):1910–1920. BIC-TA 2009 Special Issue, International Conference on Bio-Inspired Computing: Theory and Applications.

Neal, R. M. and G. E. Hinton, 1993. A new view of the EM algorithm that justifies incremental and other variants. *Learning in Graphical Models*, p. 355–368.

Oliveira, L. E. S. and R. Sabourin, 2004. Support vector machines for handwritten numeral string recognition. *9th International Workshop Frontiers in Handwriting Recognition (IWFHR-9)*, p. 39–44, Kokubunji, Tokyo, Japan.

Oliveira, L. E. S., R. Sabourin, F. Bortolozzi, and C. Y. Suen, 2002. Automatic recognition of handwritten numeral strings: A recognition and verification strategy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(11):1438–1454.

O'Shaughnessy, D., 2008. Invited paper: Automatic speech recognition: History, methods and challenges. *Pattern Recognition*, 41(10):2965–2979.

Park, Y. and J. Sklansky, 1990. Automated design of linear tree classifiers. *Pattern Recognition*, 23(12):1393–1412.

Polikar, R., L. Udpa, S. S. Udpa, and V. Honavar, 2001. Learn++: An incremental learning algorithm for supervised neural networks. *Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 31(4):497–508.

Rabiner, L. R., 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Radtke, P., 2006. *Classification Systems Optimization with Multi-objective Evolutionary Algorithms*. PhD thesis, École de Technologie Supérieure (ETS), Montreal, Canada.

Radtke, P., T. Wong, and R. Sabourin, 2006. An evaluation of over-fit control strategies for multi-objective evolutionary optimization. *International Joint Conference on Neural Networks (IJCNN 2006)*, p. 3327–3334.

Rheaume, F., A.-L. Jousselme, D. Grenier, E. Bosse, and P. Valin, 2002. New initial basic probability assignments for multiple classifiers. I. Kadar, editor, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 4729, p. 319–328.

Rokach, L., 2010. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1):1–39.

Ruta, D. and B. Gabrys, 2002. A theoretical analysis of the limits of majority voting errors for multiple classifier systems. *Pattern Analysis & Applications*, 5:333–350.

Ruta, D. and B. Gabrys, 2005. Classifier selection for majority voting. *Information Fusion*, 1: 63–81.

Serpico, S. B., L. Bruzzone, and F. Roli, 1996. An experimental comparison of neural and statistical non-parametric algorithms for supervised classification of remote-sensing images. *Pattern Recognition Letters*, 17(3):1331–1341.

Shipp, C. A. and L. I. Kuncheva, 2002. Relationships between combination methods and measures of diversity in combining classifiers. *Information Fusion*, 3(2):1351–48.

Singer, Y. and M. K. Warmuth, 1997. Training algorithms for hidden Markov models using entropy based distance functions. *Advances in Neural Information Processing Systems*, vol. 9, p. 641–648. The MIT Press.

Singh, S. and M. Singh, 2005. A dynamic classifier selection and combination approach to image region labelling. *Signal Processing: Image Communication*, 20(3):219–231.

Soares, R. G. F., A. Santana, A. M. P. Canuto, and M. C. P. de Souto, 2006. Using accuracy and diversity to select classifiers to build ensembles. *Proceedings of the 2006 International. Joint Conference on Neural Networks*, p. 1310–1316, Vancouver, Canada.

Stenger, B., V. Ramesh, N. Paragios, F. Coetzee, and J. Buhmann, 2001. Topology free hidden Markov models: Application to background modeling. *International Conference on Computer Vision*, vol. 1, p. 294–301.

Su, T.-H., T-W. Zhang, D-J. Guan, and H-J. Huang, 2009. Off-line recognition of realistic chinese handwriting using segmentation-free strategy. *Pattern Recognition*, 42(1):167–182.

Tan, X., S. Chen, Z-H. Zhou, and F. Zhang, 2006. Face recognition from a single image per person: A survey. *Pattern Recognition*, 39(9):1725–1745.

Ulas, A., M. Semerci, O. T. Yildiz, and E. Alpaydin, 2009. Incremental construction of classifier and discriminant ensembles. *Information Sciences*, 179:1298–1318.

Wang, X., 1994. Durationally constrained training of HMM without explicit state durational PDF. *Institute of Phonetic Sciences, University of Amsterdam, Proceedings 18*, p. 111–130.

Woods, K., W. P. Jr. Kegelmeyer, and K. Bowyer, 1997. Combination of multiple classifiers using local accuracy estimates. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(4):405–410.

Yu-Shu, C. and C. Yi-Ming, 2009. Combining incremental hidden Markov model and AdaBoost algorithm for anomaly intrusion detection. *CSI-KDD '09: Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics*, p. 3–9, New York, NY, USA.

Zhang, X. and Y. Gao, 2009. Face recognition across pose: A review. *Pattern Recognition*, 42 (11):2876–2896.

Zhao, W., R. Chellappa, J. Phillips, and A. Rosenfeld, 2003. Face recognition: A literature survey. *ACM Computing Surveys*, p. 399–458.

Zhu, X., X. Wu, and Y. Yang, 2004. Dynamic classifier selection for effective mining from noisy data streams. *Proceedings of the 4th IEEE International Conference on Data Mining*, p. 305–312, Washington, DC, USA. IEEE Computer Society.