Pattern Recognition 45 (2012) 3544–3556

Contents lists available at SciVerse ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

LoGID: An adaptive framework combining local and global incremental learning for dynamic selection of ensembles of HMMs

Paulo R. Cavalin [a,*], Robert Sabourin [b], Ching Y. Suen [c]

[a] Universidade Federal do Tocantins (UFT), École de Technologie Supérieure (ETS), Quadra 109 Norte Av. NS15 s/n Bl. II sala 21, Palmas (TO) 77001-090, Brazil
[b] École de Technologie Supérieure (ETS), 1100 Notre-dame ouest, Montréal (QC), Canada H3C-1K3
[c] Centre for Pattern Recognition and Machine Intelligence (CENPARMI), Concordia University, 1455 de Maisonneuve Blvd West, Montréal (QC), Canada H3G-1M8

ARTICLE INFO

Article history:
Received 10 August 2011
Received in revised form
28 November 2011
Accepted 23 February 2012
Available online 10 March 2012

Keywords:
Adaptive systems
Ensembles of classifiers
Incremental learning
Dynamic selection
Hidden Markov models

ABSTRACT

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.

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 [1–3].

The use of ensembles of classifiers (EoCs) [4] 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 [1,2,5–7]. 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 [8], the use of EoCs in dynamic selection may provide better performance than static selection in settings involving a high level of uncertainty. The main approach, called

* Corresponding author.
E-mail addresses: cavalin@uft.edu.br (P.R. Cavalin), robert.sabourin@etsmtl.ca (R. Sabourin), suen@cse.concordia.ca (C.Y. Suen).

[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.

0031-3203/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2012.02.034

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:

(1) *Local*: incremental learning of the pool of base classifiers;
(2) *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 [9] 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 [1] to incrementally generate a diverse pool of classifiers. Given the KNOP architecture, global incremental learning is realized 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 [2], 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 2 presents an overview of incremental learning and dynamic selection. In Section 3, the proposed LoGID approach is described in greater detail. The experimental evaluation is presented in Section 4, and the conclusions and future work are discussed in Section 5.

[2] An output profile consists of a vector containing the outputs yielded by the base classifiers [10].

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.

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 [1]:

(1) Incorporation of new knowledge into an existing classification scheme;
(2) No loss of previous knowledge in the process; if there is a loss, the system is said to suffer from *catastrophic forgetting*;
(3) Reduction of the complexity overhead of batch learning (BL), the requirements of which, in terms of memory and time, increase as 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 [3,11,12]. Many authors have proposed one-pass versions of BL counterparts [11,12]. 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) [1,2,5–7]. When new data are available, new members can be appended to an existing pool of classifiers. Since 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 [1,2,6]. 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

[3] Some authors maintain that the algorithm should use no previous data at all [1], 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 [2].

weighting of classifier outputs may result in better recognition rates than static classifier combination [13,14]. 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.

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 $x_{i,test}$ [9,15–19]. 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 [9,15], clustering [20], multiple training datasets [21], or measures considering the outputs produced by the base classifiers [18].

Recently, instance-based DS approaches [8,9,22] 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 multistage 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 [9] 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. 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 $x_{i,test}$.

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 Fig. 1, and its main steps are formalized in Algorithm 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.

Algorithm 1. The main steps of the LoGID framework.

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

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

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 3.2. 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, $DSel'$.

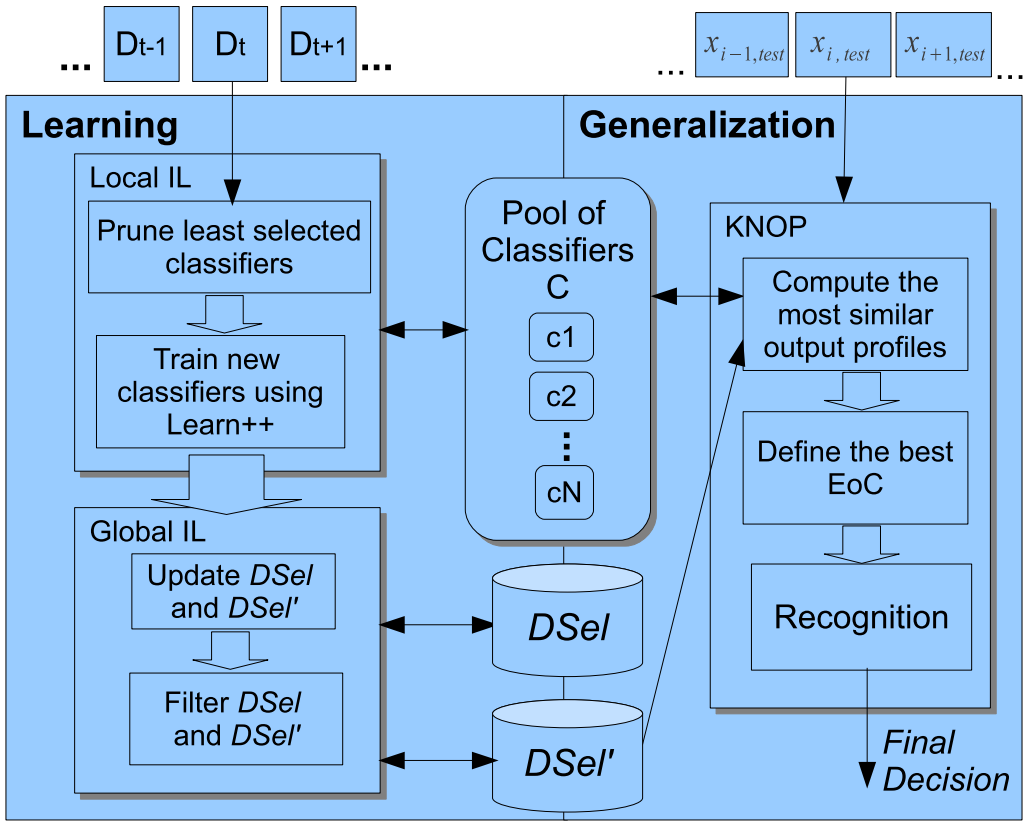

Fig. 1. General overview of the LoGID architecture.

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 [1]. 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.

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 2. First, a predefined threshold, denoted N_{max}, is used to evaluate the size of C (steps 2–4). 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).

Algorithm 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 3.1.1.1, prune the $N - N_{max}$ least used classifiers from C
4: **end if**
5: Call Algorithm 3 to update C
6: **Output:** the updated pool of classifiers C

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 3.2). 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.

3.1.1.2. 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 $1 \le k \le T_k$, 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.

Algorithm 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 ϵ_k on D_t
8: **if** $\epsilon_k > 1/2$ **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

The main steps of Learn++ are formalized in Algorithm 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 ϵ_k of this classifier is computed on D_t. If ϵ_k is above 1/2, c_k is discarded and the algorithm jumps back to step 5 (step 9). If c_k is not discarded during the individual evaluation, this classifier is 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 5. 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 [1].

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 3.2) to dynamically select EoCs.

This phase, formalized in Algorithm 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 2–4). 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 5–9). 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, $x_{j,unseen}$ 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 12–18).

Algorithm 4. The global incremental learning algorithm.

1: **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 $DSel'$
2: **for** each $x_{j,dsel}$ in $DSel$ **do**
3: Update its corresponding output profile in $DSel'$, i.e. $\tilde{x}_{j,dsel}$, to store the outputs of the new classifiers in C
4: **end for**
5: **for** each $x_{j,unseen}$ in D_t **do**
6: Compute $\tilde{x}_{j,unseen}$, i.e. the output profile of $x_{j,unseen}$
7: Put $x_{j,unseen}$ in $DSel_t$.
8: Put $\tilde{x}_{j,unseen}$ in $DSel'_t$.
9: **end for**
10: $DSel = DSel \cup DSel_t$
11: $DSel' = DSel' \cup DSel'_t$
12: **for** each $x_{j,dsel}$ in $DSel$ **do**
13: Compute m_j for $\tilde{x}_{j,dsel}$, using Eq. (1)
14: **if not** $(\vartheta_{min} \le m_j \le \vartheta_{max})$ **then**
15: Remove $x_{j,dsel}$ from $DSel$.
16: Remove $\tilde{x}_{j,dsel}$ from $DSel'$.
17: **end if**
18: **end for**
19: **Output:** the updated sets $DSel$ and $DSel'$

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 Eq. (1). Note that $\nu 1_j$ 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, $\nu 2_j$ corresponds to the number of votes received by the class placing second.

$$m_j = \frac{\nu 1_j - \nu 2_j}{N}, \text{ where } 0 \le m_j \le 1 \quad \text{and} \quad N = |C| \tag{1}$$

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

3.2. Generalization phase—the KNOP algorithm

The generalization phase involves the application of the K-nearest Output Profiles (KNOP) algorithm, depicted in Fig. 2, to recognize each test sample $x_{i,test}$. The main steps of this algorithm are presented as Algorithm 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 Ψ_i. Next, the samples in Ψ_i are used to define the best ensemble C_i^* for recognizing $x_{i,test}$ (steps 5–11), where $|C_i^*| = U_i$. For each sample in Ψ_i, if a classifier c_k correctly recognizes this sample, it is added to C_i^*.

Algorithm 5. Complete KNOP algorithm for HMMs.

1: **for** each data point $x_{i,test}$ in *Test* **do**
2: Compute $\tilde{x}_{i,test}$ using transformation T, as defined in Eq. (2)
3: Considering $DSel'$, find the K $\tilde{x}_{j,dsel}$ most similar to $\tilde{x}_{i,test}$ and put into Ψ_i
4: $C_i^* = \emptyset$
5: **for** each $\tilde{x}_{j,dsel}$ in Ψ_i **do**
6: **for** each c_k in C **do**
7: **if** KNORA-Union's rules are satisfied **then**
8: Insert c_k into C_i^*
9: **end if**
10: **end for**
11: **end for**
12: Compute cl_i from C_i^* using Eq. (3)
13: *# Switch mechanism*
14: **if** $cl_i > \theta$ **then**
15: $d_i =$ most voted class from C_i^*
16: **else**
17: $d_i =$ the label of the most similar $\tilde{x}_{j,dsel}$ from $DSel'$
18: **end if**
19: **end for**

Before computing the final decision d_i, the switch mechanism is used (steps 12–18), 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 θ, 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.

3.2.1. 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 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 [8].

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:

$$T : x_i \Rightarrow \tilde{x}_i, \tag{2}$$

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 Λ_j for all classes, given $x_{i,test}$. Also, consider the set of scores produced by Λ_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.

3.2.2. 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 Ψ_i. Then, a selection algorithm inspired by the KNORA-OP-Union method [23] uses the output profiles in Ψ_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 Ψ_i, suppose each $\tilde{x}_{j,dsel}$ has been correctly classified by a set of classifiers C_j'. Every classifier $c_k \in C_j'$ 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 Ψ_i.

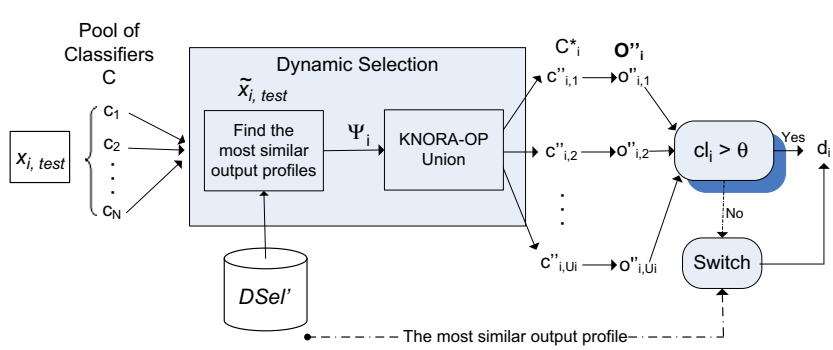

Fig. 2. Overview of the KNOP approach.

After C_i^* is computed, the final decision is evaluated by the switch mechanism.

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 the following equation:

$$cl_i = \frac{v1_i - v2_i}{K \times N} \tag{3}$$

cl_i is based on the margin [24] 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 θ 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 θ, 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 Eq. (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 θ, in the 0–1 range, can be used for pools of classifiers with different cardinalities.

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 1.

The parameters of the proposed approach are presented in Table 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 [26] 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 [27]. The number of states for each left-right HMM is computed with Wang's method [28]. The value of K is set to 30 for the KNOP algorithm, since this value worked well for many applications in [8]. 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).

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 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, 10 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.

Table 1
The databases considered in this work.

Database	Problem	Source
Japanese Vowels	Voice recognition	UCI [25]
Arabic Spoken Digits	Speech recognition	UCI [25]
Isolated Uppercase Letters	Handwriting recognition	NIST SD19
Isolated Digits	Handwriting recognition	NIST SD19

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.

4.1.1. Parameter setting

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

(1) Each parameter is evaluated in this sequence: T_k, θ, ϑ_{min} and ϑ_{max}, then N_{max}.
(2) 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 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 θ, 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 θ is evaluated in the 0.0–1.0 range, with an interval of 0.1 between each evaluation. Owing to

Table 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. θ, T_k, ϑ_{min}, ϑ_{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	θ	ϑ_{min}	ϑ_{max}	N_{max}
Japanese	216	54	370	3	9	12	24	10	0.1	0.2	1.0	15
Arabic	5500	1100	2200	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 *test2* - 58,646	18	10	47	256	5	0.3	0.2	0.8	200

space constraints, only the best values for these parameters are listed in Table 2.

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

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 Figs. 7–10, 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.

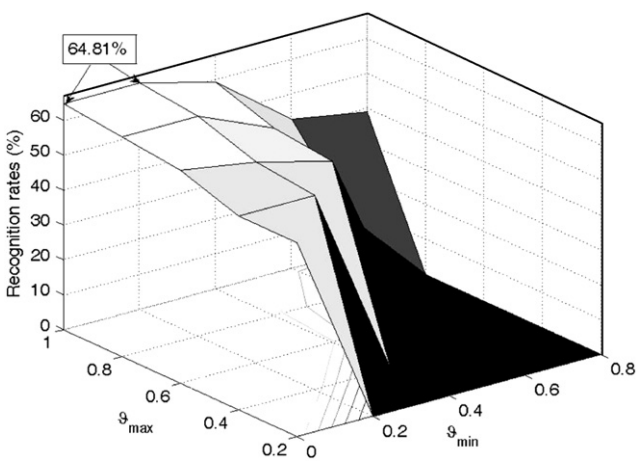

Fig. 3. Evaluation of different values for ϑ_{min} and ϑ_{max} on the dynamic selection set of Japanese Vowels. The best recognition rates are reached with $\vartheta_{min}=0.2$ and $\vartheta_{max}=1.0$.

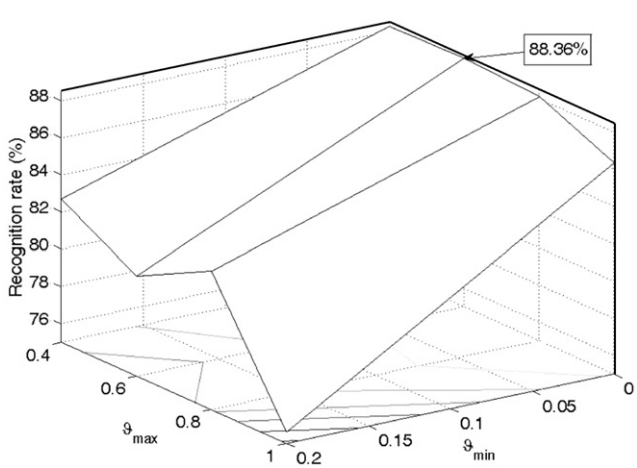

Fig. 4. Evaluation of different values for ϑ_{min} and ϑ_{max} on the dynamic selection set of Arabic Digits. The best recognition rates are reached with $\vartheta_{min}=0$ and $\vartheta_{max}=0.6$.

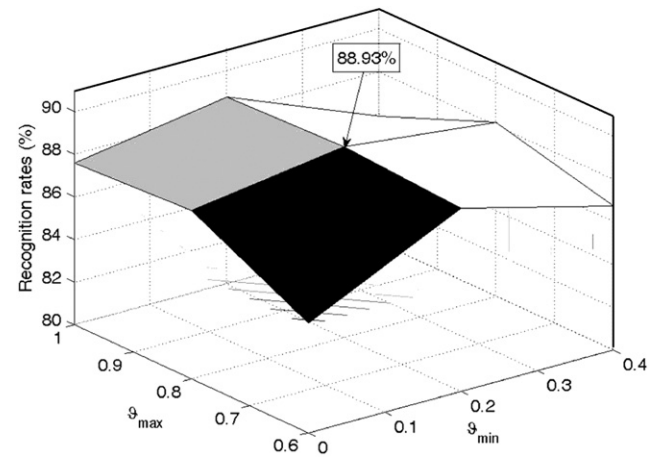

Fig. 5. Evaluation of different values for ϑ_{min} and ϑ_{max} on the dynamic selection set of NIST Letters. The best recognition rates are reached with $\vartheta_{min}=0.2$ and $\vartheta_{max}=0.8$.

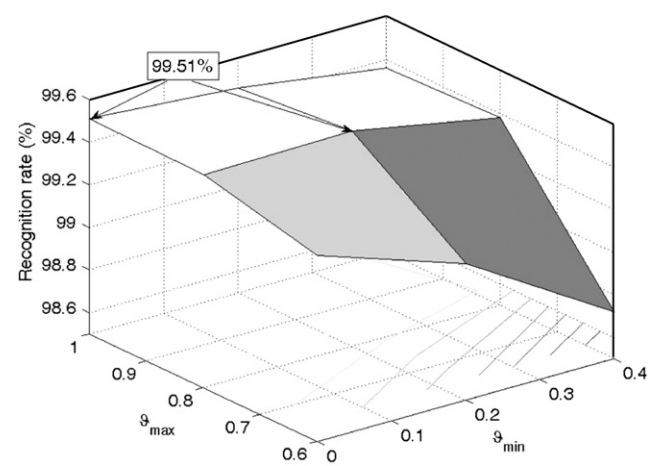

Fig. 6. Evaluation of different values for ϑ_{min} and ϑ_{max} on the dynamic selection set of NIST Digits. The best recognition rates are reached with $\vartheta_{min}=0.2$ and $\vartheta_{max}=0.8$.

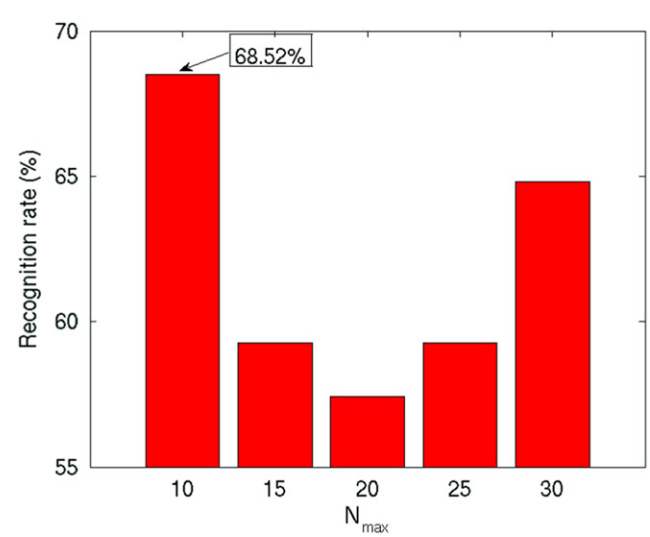

Fig. 7. For Japanese Vowels, $N_{max}=10$ provides the best recognition rates on the dynamic selection set.

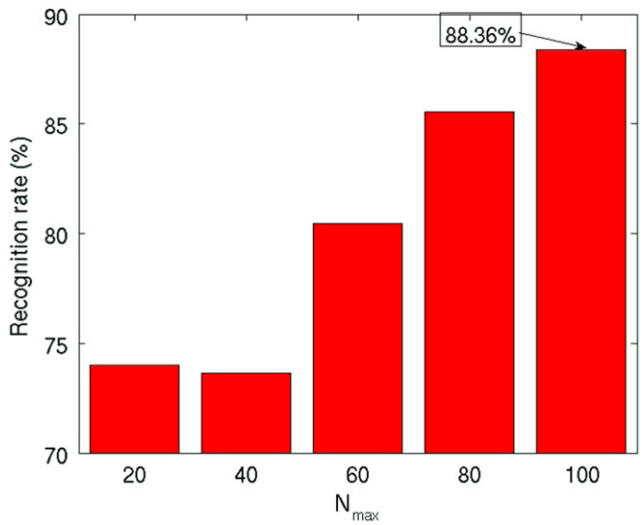

Fig. 8. For Arabic Digits, $N_{max} = 100$ yields the best results on the dynamic selection set.

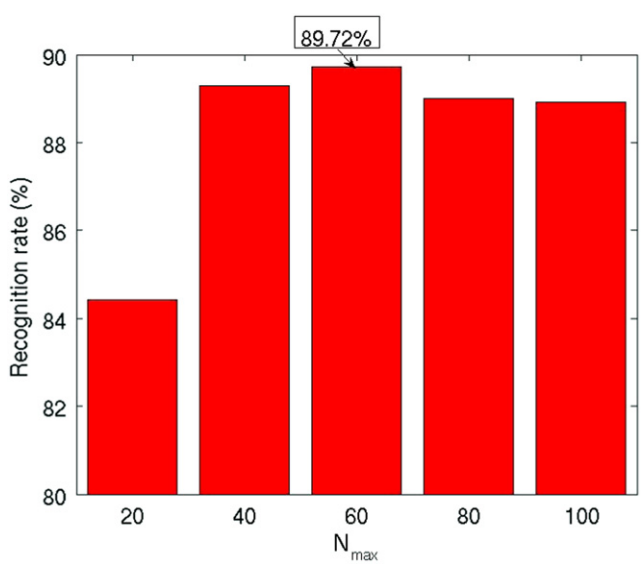

Fig. 9. For NIST Letters, $N_{max} = 60$ yields the best results on the dynamic selection set.

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:

- *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;
- *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 [26] 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 [1,14];

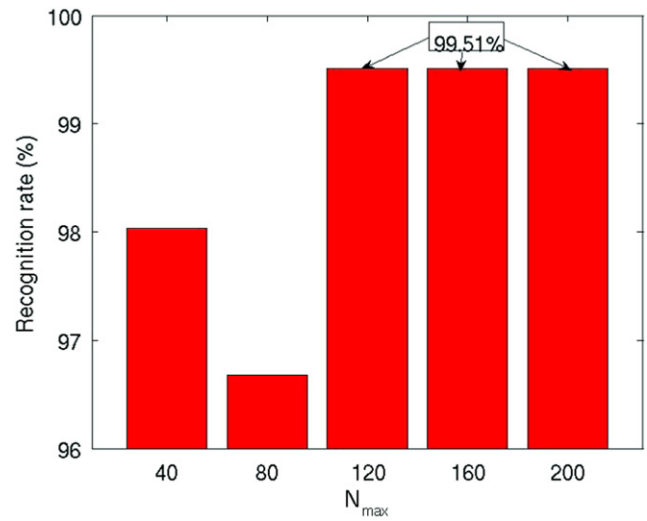

Fig. 10. For NIST Digits, three values for N_{max}: 120, 160, and 200, yield the best results on the dynamic selection set. The smallest value, i.e. $N_{max} = 120$, is the preferred one.

- *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 Fig. 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 Fig. 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.

The performance comparison for NIST Letters is presented in Fig. 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 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 Fig. 14. LoGID achieved the best recognition rates on this database, at about 98.84%. Global IL yielded the second best

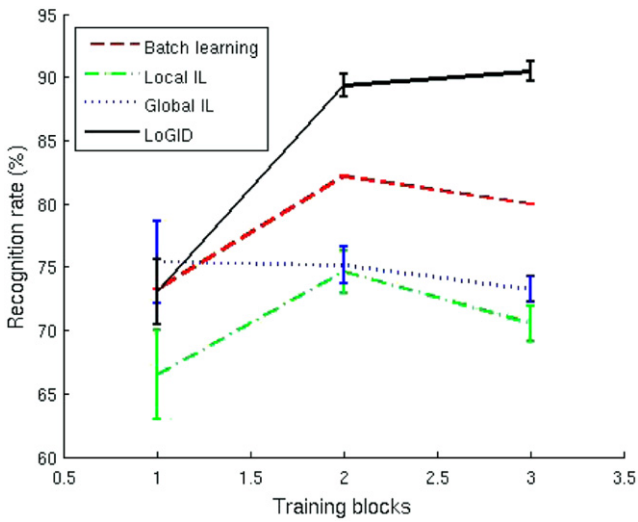

Fig. 11. Performance comparison for Japanese Vowels.

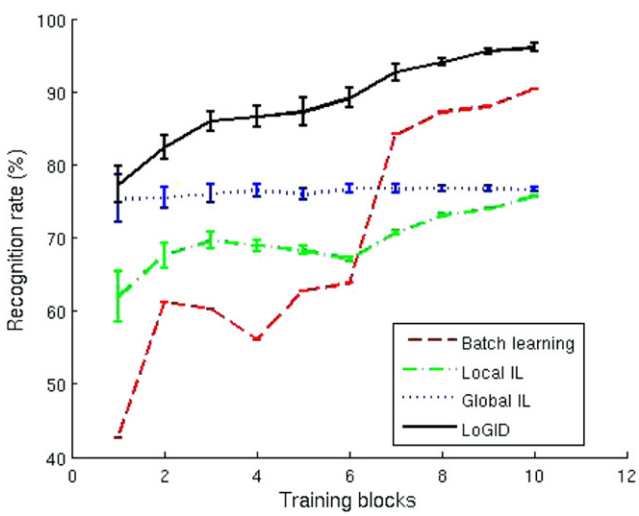

Fig. 12. Performance comparison for Arabic Digits.

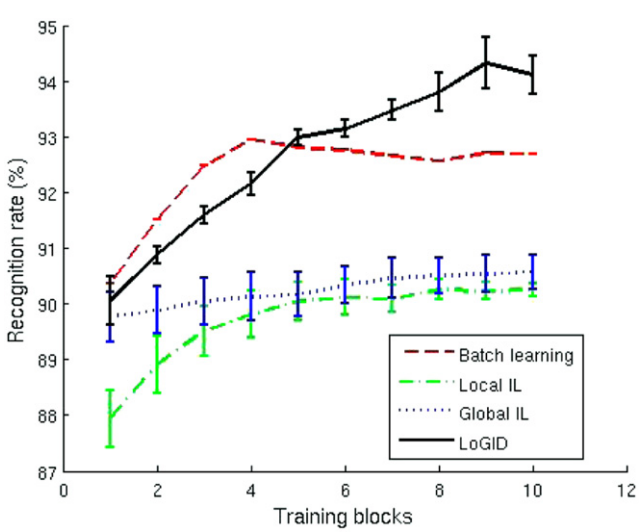

Fig. 13. Performance comparison for NIST Letters.

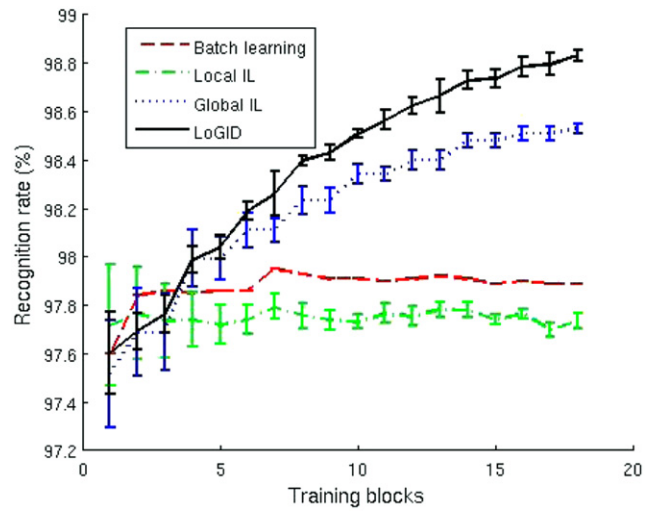

Fig. 14. Performance comparison for NIST Digits in *test1*.

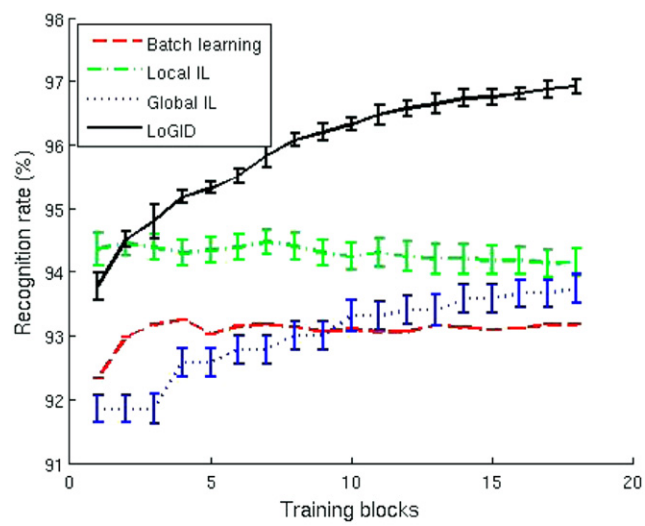

Fig. 15. Performance comparison for NIST Digits in *test2*.

result, at 98.53%. We observe that the performance of both LoGID and Global IL evolve 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 than 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.

Fig. 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.

4.1.3. Impact of the filtering mechanism on the size of $DSel'$

To demonstrate the impact of the filtering mechanism described in Section 3.1.2.1, we compare the size of $DSel'$ that

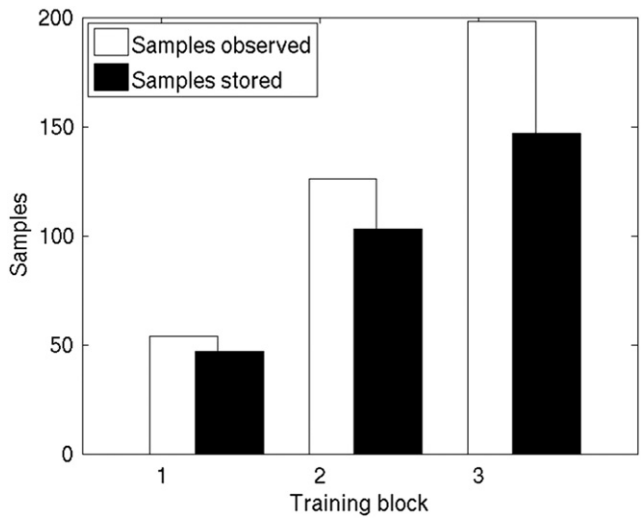

Fig. 16. Comparison of the number of samples held in *DSel′* with all the samples observed, on Japanese Vowels.

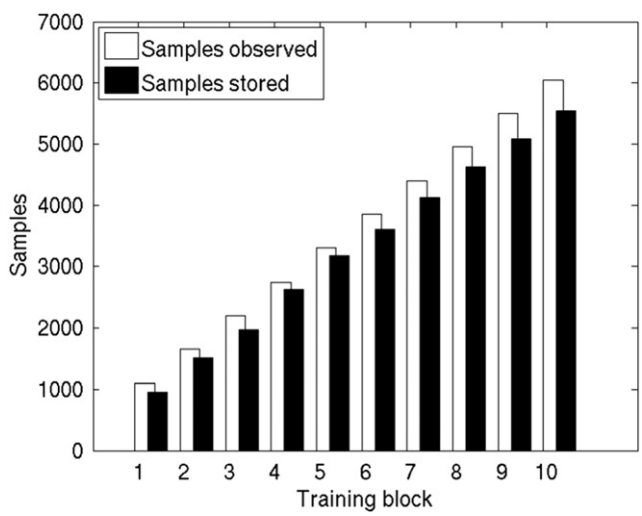

Fig. 17. Comparison of the number of samples held in *DSel′* with all the samples observed, on Arabic Digits.

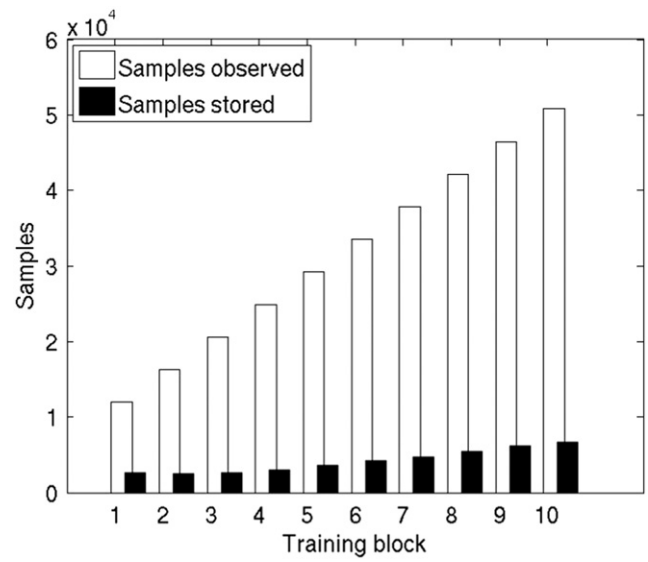

Fig. 18. Comparison of the number of samples held in *DSel′* with all the samples observed, on NIST Letters.

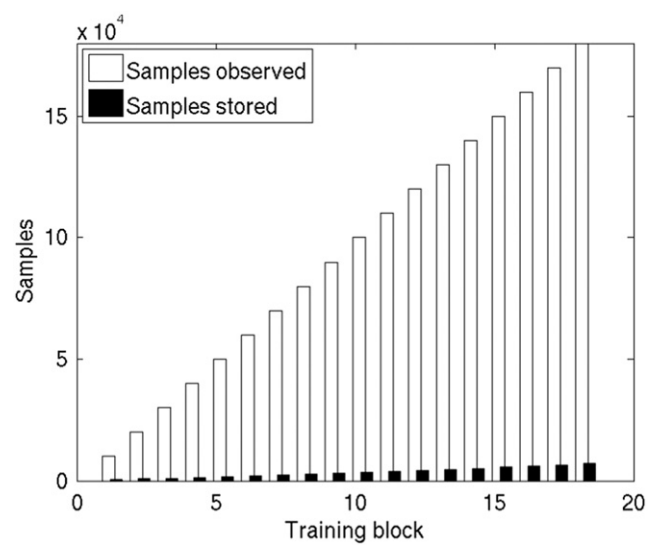

Fig. 19. Comparison of the number of samples held in *DSel′* with all the samples observed, on NIST Digits.

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 Figs. 16–19 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.

It is also interesting to note that the use of this mechanism can also successfully replace samples that are no longer considered useful. In Fig. 18, for example, we see that there is no increase in the size of the dynamic selection set during the learning of blocks 1–3. We see, though, that a considerable number of samples is observed and that the performance of the system improves, as shown in Fig. 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.

4.2. Discussion

In Table 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 [1,14]. 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,

Table 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 [29]	99.37 [30] (SVM)
Digits *test2*	**96.91 (0.11)**	94.15 (0.21)	93.74 (0.22)	93.18 (–)	96.00 (0.08)	NA	97.67 [31] (NN)
Letters	**94.10 (0.34)**	90.25 (0.14)	90.57 (0.31)	92.69 (–)	91.63 (0.40)	93.24 [2]	98.09 [32] (NN)
Arabic	**95.99 (0.57)**	75.68 (0.11)	76.54 (0.31)	90.36 (–)	86.09 (0.60)	NA	93.12 [33] (TM)
Japanese	**90.43 (0.78)**	70.54 (1.40)	73.24 (1.01)	80.00 (–)	80.54 (0.70)	NA	NA

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, we can conclude that the proposed approach can also perform better than both ensembles and single classifiers trained with batch learning.

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 [29,34]. 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.

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 [35].

Acknowledgments

The authors acknowledge CAPES Brazil and NSERC Canada for their financial support.

References

[1] R. Polikar, L. Udpa, S.S. Udpa, V. Honavar, Learn++: an incremental learning algorithm for supervised neural networks, Systems, Man, and Cybernetics-Part C: Applications and Reviews 31 (4) (2001) 497–508.
[2] P.R. Cavalin, R. Sabourin, C.Y. Suen, A.S. Britto Jr., Evaluation of incremental learning algorithms for HMM in the recognition of alphanumeric characters, Pattern Recognition 42 (12) (2009) 3241–3253. (New Frontiers in Handwriting Recognition).
[3] G. Mongillo, S. Deneve, Online learning with hidden Markov models, Neural Computation 20 (2008) 1706–1716.
[4] L. Rokach, Ensemble-based classifiers, Artificial Intelligence Review 33 (1) (2010) 1–39.
[5] C. Yu-Shu, C. Yi-Ming, Combining incremental hidden Markov model and AdaBoost algorithm for anomaly intrusion detection, in: CSI-KDD '09: Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics, New York, NY, USA, 2009, pp. 3–9.
[6] A. Ulas, M. Semerci, O.T. Yildiz, E. Alpaydin, Incremental construction of classifier and discriminant ensembles, Information Sciences 179 (2009) 1298–1318.
[7] M. Kapp, R.R. Sabourin, P. Maupin, Adaptive incremental learning with an ensemble of support vector machines, in: 20th International Conference on Pattern Recognition, Istanbul, Turkey, 2010, pp. 4048–4051.
[8] P.R. Cavalin, R. Sabourin, C.Y. Suen, Dynamic selection approaches for multiple classifiers systems, Neural Computing and Applications, doi: 10.1007/s00521-011-0737-9, in press.

[9] A.H. Ko, R. Sabourin, A.S. Britto Jr., From dynamic classifier selection to dynamic ensemble selection, Pattern Recognition 41 (5) (2008) 1718–1731.
[10] 'L.I. Kuncheva, J.C. Bezder, R.P.W. Duin, Decision templates for multiple classifier fusion: an experimental comparison, Pattern Recognition 34 (2001) 299–314.
[11] J. Mizuno, T. Watanabe, K. Ueki, K. Amano, E. Takimoto, A. Maruoka, On-line estimation of hidden Markov model parameters, in: Third International Conference Discovery Science (DS 2000), vol. 1967, 2000, pp. 155–169.
[12] G. Florez-Larrahondo, S. Bridges, E.A. Hansen, Incremental estimation of discrete hidden Markov models based on a new backward procedure, in: National Conference on Artificial Intelligence, 2005, pp. 758–763.
[13] A. Gangardiwala, R. Polikar, Dynamically weighted majority voting for incremental learning and comparison of three boosting based approaches, in: Proceedings of the International Joint Conference on Neural Networks, 2005, pp. 1131–1136.
[14] M.D. Muhlbaier, A. Topalis, R. Polikar, 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) (2009) 152–168.
[15] K. Woods, W.P.J. Kegelmeyer, K. Bowyer, Combination of multiple classifiers using local accuracy estimates, IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (4) (1997) 405–410.
[16] G. Giacinto, F. Roli, Dynamic classifier selection based on multiple classifier behaviour, Pattern Recognition 34 (2001) 1879–1881.
[17] X. Zhu, X. Wu, Y. Yang, Dynamic classifier selection for effective mining from noisy data streams, in: Proceedings of the 4th IEEE International Conference on Data Mining, IEEE Computer Society, Washington, DC, USA, 2004, pp. 305–312.
[18] E.M. Dos Santos, R. Sabourin, P. Maupin, A dynamic overproduce-and-choose strategy for the selection of classifier ensembles, Pattern Recognition 41 (2008) 2993–3009.
[19] R.G.F. Soares, A. Santana, A.M.P. Canuto, M.C.P. de Souto, Using accuracy and diversity to select classifiers to build ensembles, in: Proceedings of the 2006 International Joint Conference on Neural Networks, Vancouver, Canada, 2006, pp. 1310–1316.
[20] L.I. Kuncheva, Cluster-and-selection model for classifier combination, in: Proceedings of the International Conference on Knowledge Based Intelligent Engineering Systems and Allied Technologies, Brighton, UK, 2000, pp. 185–188.
[21] S. Singh, M. Singh, A dynamic classifier selection and combination approach to image region labelling, Signal Processing: Image Communication 20 (3) (2005) 219–231.
[22] P.R. Cavalin, R. Sabourin, C.Y. Suen, Dynamic selection of ensembles of classifiers using contextual information, in: 9th International Workshop on Multiple Classifier Systems (MCS 2010), Cairo, Egypt, 2010, pp. 145–154.
[23] L. Batista, E. Granger, R. Sabourin, Dynamic ensemble selection for off-line signature verification, in: 10th International Workshop on Multiple Classifier Systems, 2011.
[24] L.K. Hansen, C. Liisberg, P. Salamon, The error-reject tradeoff, Open Systems & Information Dynamics 4 (2) (1997) 159–184.
[25] A. Frank, A. Asuncion, UCI machine learning repository 〈http://archive.ics.uci.edu/ml〉, 2010.
[26] A.S. Britto Jr., R. Sabourin, F. Bortolozzi, C.Y. Suen, Recognition of numeral strings using a two-stage HMM-based method, International Journal on Document Analysis and Recognition 5 (2) (2003) 102–117.
[27] L.R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proceedings of the IEEE 77 (2) (1989) 257–286.
[28] X. Wang, Durationally constrained training of HMM without explicit state durational PDF, in: Proceedings of the Institute of Phonetic Sciences, vol. 18, University of Amsterdam, 1994, pp. 111–130.
[29] A.H. Ko, P.R. Cavalin, R. Sabourin, A.S. Britto Jr., 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 1 (2009) 2168–2178.
[30] J. Milgram, M. Cheriet, R. Sabourin, "One against one" or "One against all": which one is better for handwriting recognition with SVMs? in: Proceedings of 10th International Workshop on Frontiers in Handwriting Recognition, La Baule, France, 2006.
[31] P. Radtke, T. Wong, R. Sabourin, An evaluation of over-fit control strategies for multi-objective evolutionary optimization, in: International Joint Conference on Neural Networks (IJCNN 2006), 2006, pp. 3327–3334.
[32] D.C. Ciresan, U. Meier, L.M. Gambardella, J. Schmidhuber, Convolutional neural network committees for handwritten character classification, in: Proceedings of the 11th International Conference on Document Analysis and Recognition (ICDAR 2011), 2011.
[33] N. Hammami, M. Bedda, Improved tree model for Arabic speech recognition, in: 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), vol. 5, 2010, pp. 521–526.
[34] A.H. Ko, R. Sabourin, A.S. Britto Jr., Ensemble of HMM classifiers based on the clustering validity index for a handwritten numeral recognizer, Pattern Analysis and Applications 12 (2009) 21–35.
[35] B. Cui, H.T. Shen, J. Shen, K.-L. Tan, Exploring bit-difference for approximate KNN search in high-dimensional databases, in: Proceedings of the 16th Australasian Database Conference, Australian Computer Society, Inc., 2005, pp. 165–174.

Paulo Rodrigo Cavalin received M.Sc.A. and Ph.D. degrees in Computer Science from the Universidade Catolica do Parana (PUC-PR, Brazil), in 2005, and the Ecole de Technologie Superieure, Universite du Quebec, in 2011, respectively. In 2011 he joined the Computer Science department, at Universidade Federal do Tocantins (UFT, Brazil), where he is currently a professor. His research interests include Handwriting Recognition, Hidden Markov Models, Ensemble Classification Methods, Incremental Learning, and Dynamic Selection of Classifiers.

Robert Sabourin received B.ing., M.Sc.A., Ph.D. degrees in electrical Engineering from the Ecole Polytechnique de Montreal in 1977, 1980 and 1991, respectively. In 1977, he joined the Physics Department of the Universite de Montreal where he was responsible for the design and development of scientific instrumentation for the Observatoire du Mont Megantic. In 1983, he joined the staff of the Ecole de Technologie Superieure, Universite du Quebec, Montreal, P.Q, Canada, where he is currently a professeur titulaire in the Departement de Genie de la Production Automatisee. In 1995, he joined also the Computer Science Department of the Pontificia Universidade Catolica do Parana (PUC-PR, Curitiba, Brazil) where he was coresponsible since 1998 for the implementation of a Ph.D. program in Applied Informatics. Since 1996, he is a senior member of the Centre for Pattern Recognition and Machine Intelligence (CENPARMI). His research interests are in the areas of handwriting recognition and signature verification for banking and postal applications.

Ching Y. Suen received an M.Sc.(Eng.) degree from the University of Hong Kong and a Ph.D. degree from the University of British Columbia, Canada. In 1972, he joined the Department of Computer Science at Concordia University where he became Professor in 1979 and served as Chairman from 1980 to 1984, and as Associate Dean for Research of the Faculty of Engineering and Computer Science from 1993 to 1997. He has guided/hosted 70 visiting scientists and professors, and supervised 65 doctoral and master's graduates. Currently he holds the distinguished Concordia Research Chair in Artificial Intelligence and Pattern Recognition, and is the Director of CENPARMI, the Centre for PR & MI. Prof. Suen is the author/editor of 11 books and more than 400 papers on subjects ranging from computer vision and handwriting recognition, to expert systems and computational linguistics. A Google search of "Ching Y. Suen" will show some of his publications. He is the founder of "The International Journal of Computer Processing of Oriental Languages" and served as its first Editor-in-Chief for 10 years. Presently he is the Deputy Editor of Pattern Recognition, a member of the Advisory Board of Pattern Recognition Letters, and an Associate Editor of the International Journal of Pattern Recognition and Artificial Intelligence, Signal Processing, Expert Systems with Applications, and the International Journal of Document Analysis and Recognition. He was also an Associate Editor of the IEEE Transactions on Pattern Analysis and Machine Intelligence and Pattern Analysis and Applications. A Fellow of the IEEE, IAPR, and the Academy of Sciences of the Royal Society of Canada, he has served several professional societies as President, Vice-President, or Governor. He is also the Founder and chair of several conference series including ICDAR, IWFHR, and VI. He was the General Chair of numerous international conferences, including the International Conference on Computer Processing of Chinese and Oriental Languages held in August 1988 in Toronto, International Workshop on Frontiers in Handwriting Recognition in April 1990 in Montreal, International Conference on Document Analysis and Recognition held in Montreal in August 1995, and the International Conference on Pattern Recognition, held in Quebec City in August 2002. Dr. Suen has given 150 seminars at major computer industries and various government and academic institutions around the world. He has been the principal investigator of 25 industrial/government research contracts, and has received many research grants from national and provincial funding agencies. He is a recipient of prestigious awards, including the ITAC/NSERC National Award from the Information Technology Association of Canada and the Natural Sciences and Engineering Research Council of Canada in 1992, the Concordia "Research Fellow" award in 1998, and the IAPR ICDAR award in 2005.