Pattern Recognition 42 (2009) 3241 – 3253

Contents lists available at ScienceDirect

Pattern Recognition

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

Evaluation of incremental learning algorithms for HMM in the recognition of alphanumeric characters

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

[a]École de Technologie Supérieure, 1100 Notre-dame ouest, Montréal, QC, Canada H3C-1K3
[b]Centre for Pattern Recognition and Machine Intelligence (CENPARMI), Concordia University, 1455 de Maisonneuve Blvd West, Montréal, QC, Canada H3G-1M8
[c]Pontifícia Universidade Católica do Paraná, Rua Imaculada Conceição, 1155 - Curitiba, PR 80.215-901, Brazil

ARTICLE INFO

Article history:
Received 9 August 2008
Accepted 15 October 2008

Keywords:
Incremental learning
Hidden Markov models
Ensembles of classifiers
Handwriting recognition
Isolated digits
Uppercase letters

ABSTRACT

We present an evaluation of incremental learning algorithms for the estimation of hidden Markov model (HMM) parameters. The main goal is to investigate incremental learning algorithms that can provide as good performances as traditional batch learning techniques, but incorporating 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, employing ensembles of HMMs, can lead to very promising performances. Furthermore, the use of a validation dataset demonstrated that it is possible to reach better performances than the ones presented by batch learning.

1. Introduction

The design of complex pattern recognition systems for recognizing handwriting or speech, involves the search for classifiers that produce high generalization performances [1]. 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 [2–5]. 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 [6,7].

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 [5].

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 [2–5]. Since the performance of a learning algorithm, as stated by the no free-lunch theorem [1], 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 hidden Markov model (HMM)-based handwriting recognition system [8–10]. An HMM-based framework has been selected due to the potential of HMMs for the handwriting recognition problem in general, and the application field of IL algorithms for this kind of system is vast. For instance, IL algorithms can be used to adapt previously

* Corresponding author. Tel.: +1 514 678 1367.
E-mail addresses: cavalin@livia.etsmtl.ca (P.R. Cavalin), robert.sabourin@etsmtl.ca (R. Sabourin), suen@cse.concordia.ca (C.Y. Suen), alceu@ppgia.pucpr.br (A.S. Britto Jr.).

0031-3203/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2008.10.012

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 Ref. [9]. 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 paper is organized as follows. In Section 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 3, we describe the methodology employed in this work, which includes the baseline system and the IL algorithms evaluated experimentally. Next, in Section 4, we report the experimental evaluation and analyze the corresponding results. Conclusions drawn from this work are described in Section 5.

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.

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 that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observations [11].

Mathematically, an HMM is characterized by the following components:

(1) 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 t is defined as q_t.
(2) 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.
(3) The state probability distribution $A=\{a_{ij}\}$, where

$$a_{ij}=P[q_{t+1}=S_j|q_t=S_i],\quad 1\leqslant i,j\leqslant N \tag{1}$$

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

$$b_j(k)=P[v_k \text{ in } t|q_t=S_j],\quad 1\leqslant j\leqslant N,\ 1\leqslant k\leqslant M \tag{2}$$

(5) The initial state distribution $\pi=\{\pi_i\}$,

$$\pi_i=P[q_1=S_i],\quad 1\leqslant i\leqslant N \tag{3}$$

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

$$\lambda=(A,B,\pi) \tag{4}$$

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, λ is used to process the observation sequence O.

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 λ_i, where $1\leqslant i\leqslant C$. Suppose that $L(O|\lambda_i)$ is the likelihood of O given λ_i (the likelihood can be easily computed by the forward–backward procedure or by the Viterbi algorithm [11]). In order to find c, the following equation must be used:

$$c=arg\ \max L(O|\lambda_i),\quad 1\leqslant i\leqslant C \tag{5}$$

2.3. IL algorithms for HMMs

IL 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 $t\geqslant 1$, given the current HMM λ_{t-1}. IL, in this case, consists of computing the parameters of the new HMM λ_t, where:

$$\lambda_t=\lambda'_{t-1} \tag{6}$$

In this case, λ'_{t-1} corresponds to a mathematical transformation involving both λ_{t-1} and ϕ_t, where the latter represents the sufficient statistics computed from the observation D_t and λ_{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 λ_{t-1} and ϕ_t is performed. Each aspect is discussed in the remainder of this section.

2.3.1. The amount of new data

In considering that D_t can be composed of a single observation sequence, as in Refs. [2,4], or can be composed of a block of S_t samples, as in Ref. [3], 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 λ_{t-1}, and consequently may converge very quickly. One drawback, however, is that after learning an enough 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.

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, these data are 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 are 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 [12,13].

Generally, the use of a learning rate η on D_t [4,14] 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 η, the more adaptive the algorithm to new data. Consequently, old data are 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 Ref. [2] all the samples have the same weight since ϕ_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.

2.3.3. Combining old and new information

The third issue involving IL algorithms for HMMs lies in the way λ_{t-1} and ϕ_t are combined to generate λ'_{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 λ'_{t-1} directly from λ_{t-1}. For instance, some methods compute λ'_{t-1} by performing a partial expectation step (E-step) of the Baum–Welch algorithm on λ_{t-1} and ϕ_t [2–4,12,15,16], and a maximization step (M-step) after each time t. One shortcoming of this approach is that once one parameter in λ_{t-1} is set to 0, there is no way to re-estimate this parameter again. To work around that problem, a small constant ε is added to each parameter in λ_{t-1} [2], but this solution results in an imprecise estimate of parameters and additional evaluations are required for finding the best value of ε. In Ref. [17], 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 ϕ_t, without taking into account the knowledge stored in λ_{t-1}. Then, $\bar{\lambda}_{t-1}$ and λ_{t-1} are combined to generate λ'_{t-1}. Such a combination can be done by merging both $\bar{\lambda}_{t-1}$ and λ_{t-1} [18,19], 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 [5]. This kind of approach avoids the aforementioned zero-probability problem by processing unseen data independent of the previous knowledge, which allows for performing BL 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 $\bar{\lambda}_{t-1}$.

3. Methodology

Here we present the entire methodology employed in this work. Section 3.1 describes the baseline isolated characters recognizer. In Section 3.2, we present the IL algorithms evaluated in this work. And the methodologies proposed for complexity analysis is presented in Section 3.3.

3.1. The baseline system

The baseline system is the isolated character recognizer presented in Refs. [8–10]. This recognizer is divided into three modules: Pre-processing, Feature Extraction, and Recognition (see Fig. 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).

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 two 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 of the 26 classes of uppercase letters. Note that each class is represented by two HMMs λ_c and λ_r, being λ_c an HMM trained from column observation sequences, and λ_r another HMM trained from row observation sequences. In this case, $\log L(O|\lambda)$ is represented by the following equation:

$$\log L(O|\lambda) = \log L(O|\lambda_c) + \log L(O|\lambda_r) \tag{7}$$

The system is described in Refs. [8–10] in great detail.

3.2. IL algorithms

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

3.2.1. The incremental Baum–Welch algorithm (IBW)

The incremental Baum–Welch (IBW) algorithm is a straightforward adaptation of the original BL Baum–Welch algorithm

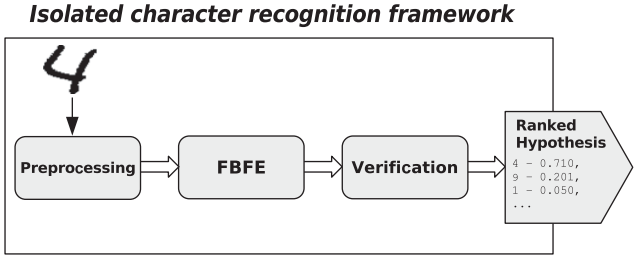

Fig. 1. An overview of the isolated character recognition framework.

to IL. First proposed in Ref. [14] for continuous HMMs, it was later adapted to discrete models in Ref. [20].

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 matrices A and B of an HMM, are updated at each time step t, given λ_{t-1} and D_t, where D_t is composed of a single observation sequence. See Algorithm 1 for detail.

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

This algorithm presents some worth noting aspects. In spite of considering only one observation sequence to update λ_{t-1}, sufficient statistics used in this algorithm represent information computed from all the observed data, meaning that each sample has the same weight. Moreover, the addition of a constant ε in matrices A and B, after doing step 8, was proposed to avoid that some parameters receive a null value.

3.2.2. The incremental maximum-likelihood algorithm (IML)

In Ref. [3], 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 λ_0, and the blocks of data $D_t, 1\leqslant t\leqslant T$ drawn from the training set, this algorithm works according to Algorithm 2.

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

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, 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 ε to matrices A and B after λ_t is computed. Otherwise, as stated before, the information cannot be completely learned by this algorithm if unseen observations are present within the new observation sequences.

3.2.3. Ensemble training (ET)

Another interesting approach for IL, namely ensemble training (ET), was presented in Refs. [18,19]. 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, matrices A, B, and π, for the final HMM, are computed in the following way:

$$\overline{a}_{ij}=\frac{\sum_k W_k a_{ij}^k}{\sum_k W_k} \tag{8}$$

$$\overline{b}_{ij}=\frac{\sum_k W_k b_{ij}^k}{\sum_k W_k} \tag{9}$$

$$\overline{\pi}_i=\frac{\sum_k W_k \pi_i^k}{\sum_k W_k} \tag{10}$$

where W_k is the weighting factor for each sequence. In this work we empirically defined that $W_k=1$, 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 λ_{t-1}, which corresponds to all data up to the time step $t-1$. The re-estimation of A, B, and π (the new current model λ_t), when new data are available, considers only λ_{t-1} and the model generated at time t (λ'_t). One important aspect to assure that the older information is kept in λ_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 λ'_t, by using the following equations:

$$\overline{a}_{ij}^t=\frac{W_{t-1}a_{ij}^{t-1}+W'_t a'^{\,t}_{ij}}{W_{t-1}+W'_t} \tag{11}$$

$$\overline{b}_{ij}^t=\frac{W_{t-1}b_{ij}^{t-1}+W'_t b'^{\,t}_{ij}}{W_{t-1}+W'_t} \tag{12}$$

$$\overline{\pi}_i^t=\frac{W_{t-1}\pi_i^{t-1}+W'_t \pi'^{\,t}_i}{W_{t-1}+W'_t} \tag{13}$$

$$W_t=W_{t-1}+W'_t \tag{14}$$

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.

3.2.4. ET 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 Ref. [5], 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 are 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 λ_{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 λ'_t, this is included in ζ, 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 ζ can be easily computed by the sum rule, by considering the logarithmic likelihood of each HMM contained in ζ. For example, after learning t blocks of data, the likelihood of ζ, given an observation sequence O, can be computed by the following equation:

$$L(O|\zeta) = \sum_{i=1}^{t} \log L(O|\lambda_i) \tag{15}$$

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.

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 B is the number of blocks of data, S_i corresponds to the number of samples in block i, where $1 \leqslant i \leqslant B$, 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 are defined by

$$F_{tr} = \sum_{i}^{B} (S_i \times I_i) \tag{16}$$

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.

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

$$F_{rec} = \sum_{i=1}^{C} N_i^2 T \tag{17}$$

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

$$F_{rec} = \sum_{i=1}^{C} N_i^2 \tag{18}$$

Eq. (18) can be easily adapted to compute the complexity of EoHMMs, which is useful for the algorithm presented in Section 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.

4. Experimental evaluation

The experimental evaluation consisted of evaluating the IL algorithms described in Section 3.2, for the recognition of handwritten isolated digits and uppercase letters. The isolated character recognition system presented in Section 3.1 was the baseline system. The algorithms are compared to a BL setting, using the traditional Baum–Welch algorithm (see Ref. [11] 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, by setting the number of states to the minimum possible value found in the training set. These parameters were defined in the previous research [9].

The isolated digits were organized in 195,000 samples for training (equally distributed into 19,500 samples for each of the 10 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 the number of states for each HMM representing a class of digit.

The uppercase letters were organized in 43,160 samples for training (equally distributed into 1660 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 2 presents the state configuration of the HMMs representing uppercase letters.

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 (1000 samples per class), and one block of 5000 samples (500 samples per class). For uppercase letters, the training set was divided into 10 blocks of 4316 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 ε, 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

Table 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			

Table 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			

Table 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

of iterations for computing λ'_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, ε 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 3 (note that the EN_stop is presented and justified further in this section).

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 are 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.

Figs. 2 and 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 Figs. 6 and 7, on the validation and test sets, respectively. In Figs. 4 and 5 we present a complexity analysis of training and recognition for digits, and in Figs. 8 and 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, 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 set. For example, the use of a stop criterion, where new HMMs are added to ζ only when the error on the validation set decreases, is able to set a good performance-complexity trade-off for EN. By observing Fig. 2, we could stop including new HMMs after learning six blocks, and the performances remain at the same level of learning 20 blocks. But as we can see in Fig. 5, the recognition complexity of learning only six blocks is significantly lower than the one for learning the whole training set.

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

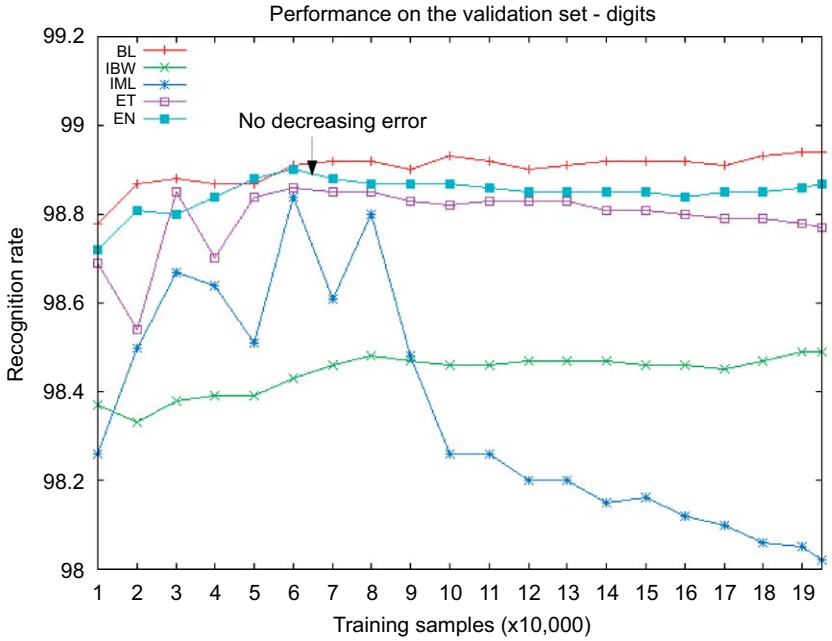

Fig. 2. The recognition results of all the algorithms on the validation set, for digits.

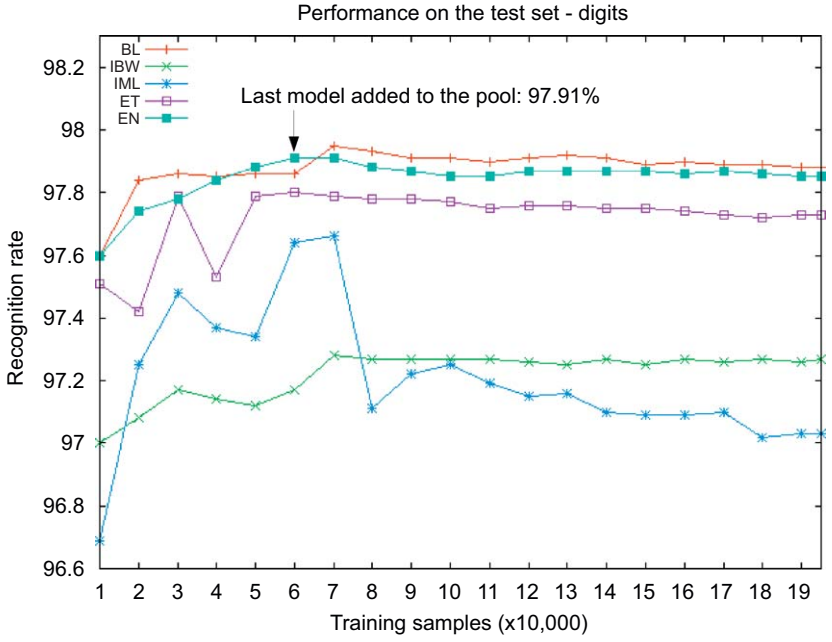

Fig. 3. The recognition results of all the algorithms on the test set, for digits.

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

In Table 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.

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

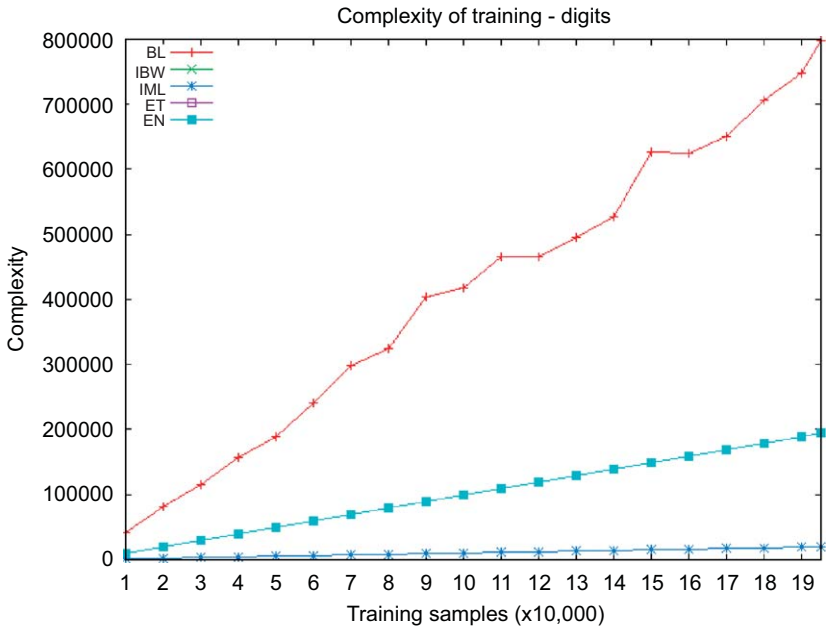

Fig. 4. Training complexity for digits.

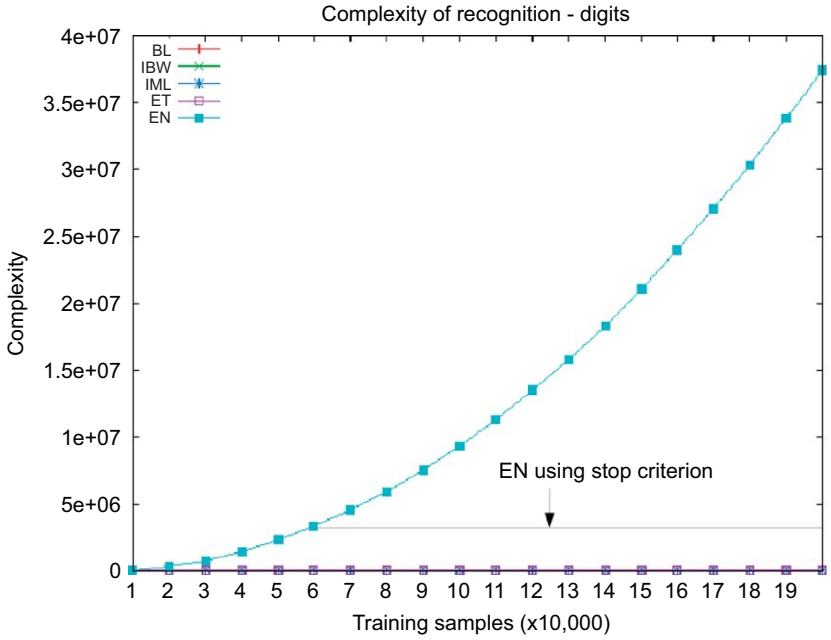

Fig. 5. Recognition complexity for digits.

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.

Fig. 10 shows the error-reject evaluation for isolated digits. We can see that both BL and ET presented very similar reject rates,

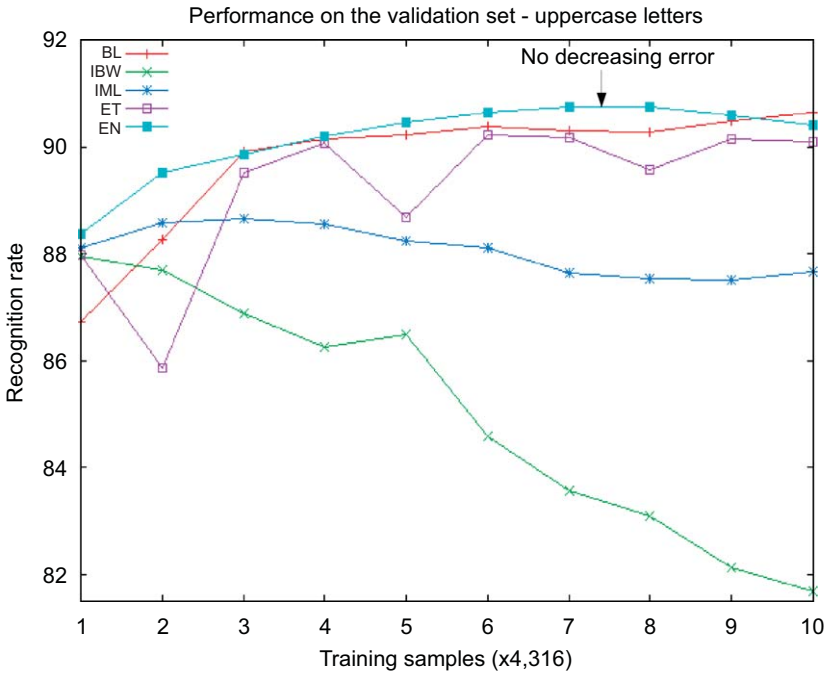

Fig. 6. The recognition results of all the algorithms on the validation set, for uppercase letters.

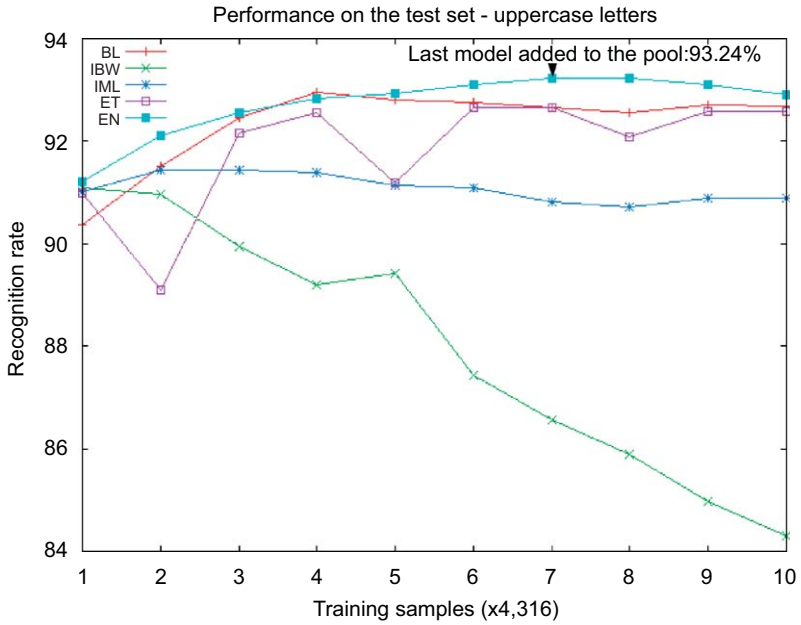

Fig. 7. The recognition results of all the algorithms on the test set, for uppercase letters.

meaning that both algorithms presents a similar reliability. Also, both EN and its modified version presented similar reject curves too, being the algorithms with the highest reliability in this problem.

Fig. 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% [6,21–23], 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% [8] to 98.86% [24]. By considering the learning approaches presented in this work, we believe that 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.

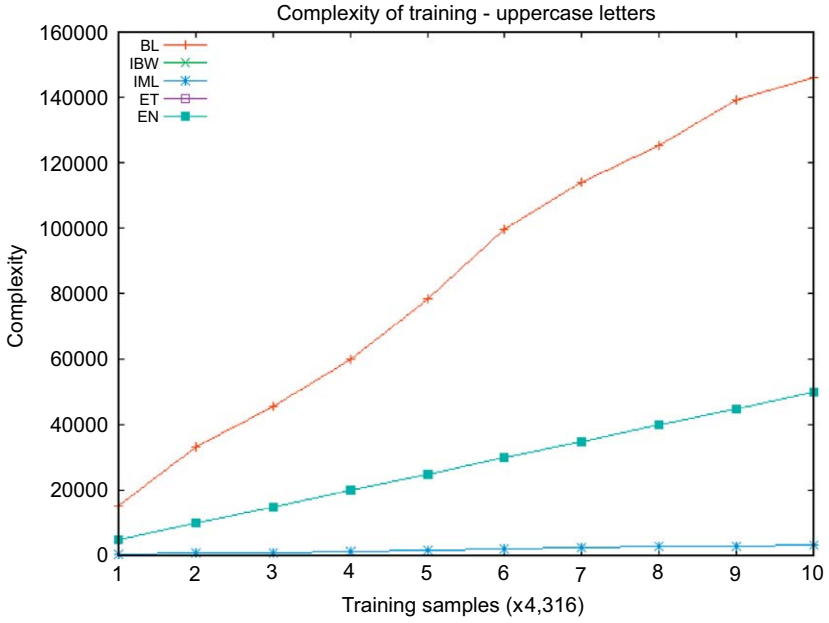

Fig. 8. Training complexity for uppercase letters.

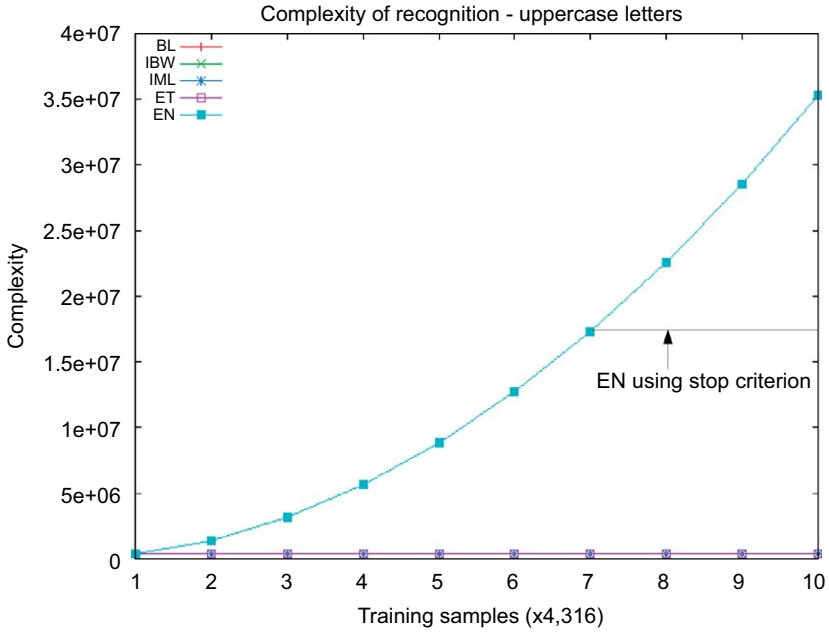

Fig. 9. Recognition complexity for uppercase letters.

Table 4
A summary of the performances of the classifiers, after learning the whole training set

Algorithm	Digits			Uppercase letters		
	Val	Test No rejection	Reject With er.:0.5%	Val	Test No rejection	Reject With 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**

The bold highlights the best results in evaluating each dataset.

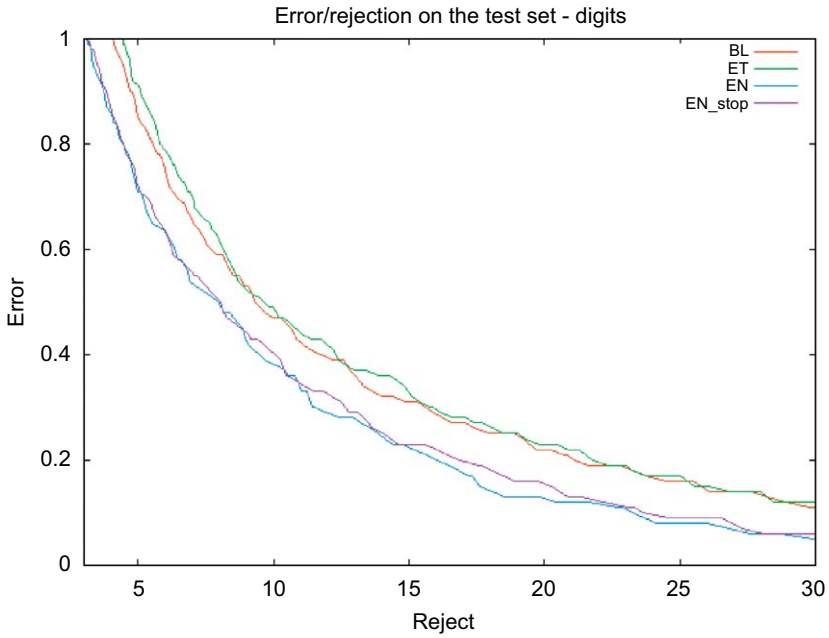

Fig. 10. Error-reject analysis of batch learning (BL), ensemble training (ET), ensemble training using EoHMMs (EN), and EN with a stop criterion, for digits.

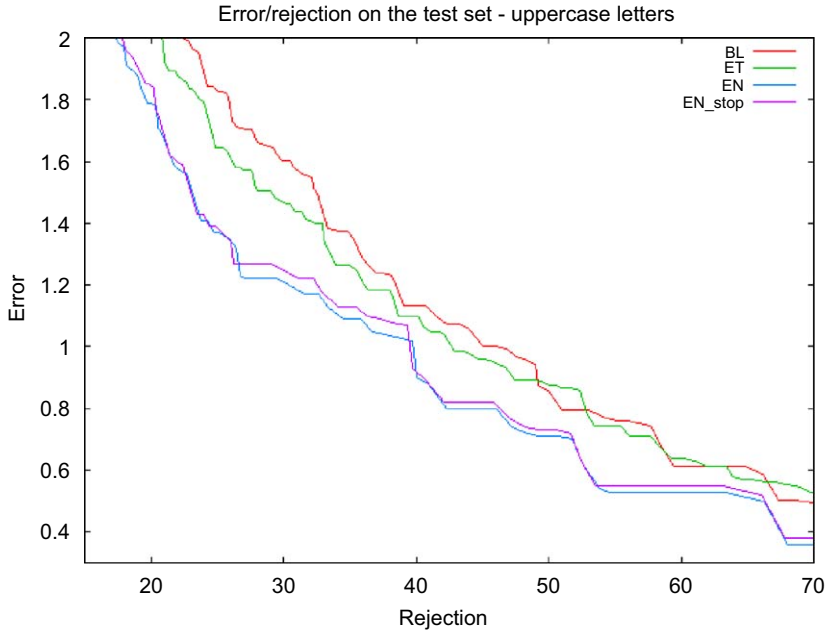

Fig. 11. Error-reject analysis of batch learning (BL), ensemble training (ET), ensemble training using EoHMMs (EN), and EN with a stop criterion, for uppercase letters.

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 [25], 95.00% and 95.98% using MLP and ensemble of MLP [26], respectively, and 96.82% using SVM [22]. 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 Ref. [24], 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.

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 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 [2]. In addition, we can also investigate techniques to optimize the codebooks' code-words from each block.

Acknowledgments

The authors would like to acknowledge the CAPES—Brazil and NSERC—Canada for the financial support.

References

[1] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, Wiley-Interscience, New York, 2000.
[2] G. Florez-Larrahondo, Incremental learning of discrete hidden Markov models, Ph.D. Thesis, Mississipi State University, 2005.
[3] Y. Gotoh, M.M. Hochberg, H.F. Silverman, Efficient training algorithms for HMM's using incremental estimation, Speech Audio Process. 6 (6) (1998) 539–548.
[4] 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.
[5] R. Polikar, L. Udpa, S.S. Udpa, V. Honavar, Learn + +: an incremental learning algorithm for supervised neural networks, Syst. Man Cybern. Part C Appl. Rev. 31 (4) (2001) 497–508.
[6] L.E.S. Oliveira, R. Sabourin, F. Bortolozzi, C.Y. Suen, Automatic recognition of handwritten numeral strings: a recognition and verification strategy, IEEE Trans. Pattern Anal. Mach. Intell. 24 (11) (2002) 1438–1454.
[7] J.-X. Dong, A. Krzyzak, C.Y. Suen, Fast SVM training algorithm with decomposition on very large data sets, Pattern Anal. Mach. Intell. 27 (4) (2005) 603–618.
[8] A.S. Britto, A two-stage HMM-based method for recognizing handwritten numeral strings, Ph.D. Thesis, Pontifícia Universidade Católica do Paraná, 2001.
[9] A.S. Britto, R. Sabourin, F. Bortolozzi, C.Y. Suen, Recognition of numeral strings using a two-stage HMM-based method, Int. J. Doc. Anal. Recognition 5 (2) (2003) 102–117.
[10] P.R. Cavalin, A.S. Britto, F. Bortolozzi, R. Sabourin, L.E.S. Oliveira, An implicit segmentation-based method for recognition of handwritten strings of characters, in: The 21st Annual ACM Symposium on Applied Computing, Dijon, France, 2006, pp. 836–840.
[11] L.R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proc. IEEE 77 (2) (1989) 257–286.
[12] V.V. Digalakis, Online adaptation of hidden Markov models using incremental estimation algorithms, Speech Audio Process. 7 (3) (1999) 253–261.
[13] J.-T. Chien, C.-H. Lee, H.-C. Wang, A hybrid algorithm for speaker adaptation using map transformation and adaptation, IEEE Signal Process. Lett. 4 (6) (1997) 167–169.
[14] B. Stenger, V. Ramesh, N. Paragios, F. Coetzee, J. Buhmann, Topology free hidden Markov models: application to background modeling, in: International Conference on Computer Vision, vol. 1, 2001, pp. 294–301.
[15] R.M. Neal, G.E. Hinton, A new view of the EM algorithm that justifies incremental and other variants, Learn. Graphical Models (1993) 355–368.
[16] Y. Singer, M.K. Warmuth, Training algorithms for hidden Markov models using entropy based distance functions, Advances in Neural Information Processing Systems, vol. 9, MIT Press, Cambridge, MA, 1997, p. 641.
[17] P. Baldi, Y. Chauvin, Smooth on-line learning algorithms for hidden Markov models, Neural Comput. 6 (2) (1994) 179–190.
[18] D.J.C. Mackay, Ensemble learning for hidden Markov models, Technical report, Cavendish Laboratory, University of Cambridge, UK, 1997.
[19] R.I.A. Davis, B.C. Lovell, Comparing and evaluating HMM ensemble training algorithms using train and test and condition number criteria, Pattern Anal. Appl. 6 (2003) 327–336.
[20] 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.
[21] L.E.S. Oliveira, R. Sabourin, Support vector machines for handwritten numeral string recognition (IWFHR-9), in: 9th International Workshop Frontiers in Handwriting Recognition, Kokubunji, Tokyo, Japan, 2004, pp. 39–44.
[22] J. Milgram, M. Cheriet, R. Sabourin, "One against one" or "One against all": which one is better for handwriting recognition with SVMs?, in: 10th International Workshop on Frontiers in Handwriting Recognition (IWFHR 2006), 2006.
[23] 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.
[24] A. Ko, R. Sabourin, A.S. Britto, A new HMM-based ensemble generation method for character recognition, in: 7th International Workshop on Multiple Classifier Systems, Prague, Czech Republic, 2007.
[25] 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.
[26] P. Radtke, Classification systems optimization with multi-objective evolutionary algorithms, Ph.D. Thesis, École de Technologie Supérieure, 2006.

About the Author—PAULO RODRIGO CAVALIN received M.Sc.A. degree in Computer Science from the Pontifical Catholic University of Parana, PUC-PR, Brazil, in 2005. Since 2006, he started his Ph.D. degree in Pattern Recognition at Ecole de Technologie Superieure, Universite du Quebec. His research interests include handwriting recognition, hidden Markov models, ensemble classification methods, and incremental learning.

About the Author—ROBERT SABOURIN received B.ing., M.Sc.A., and 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 as 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 also joined the Computer Science Department of the Pontificia Universidade Catolica do Parana, PUC-PR, Curitiba, Brazil, where he was co-responsible 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.

About the Author—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 a Professor in 1979 and served as a Chairman from 1980 to 1984, and as an 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.

About the Author—ALCEU DE SOUZA BRITTO, Jr. received M.Sc. degree in Industrial Informatics from the Federal Center for Technological Education of Parana, Brazil, in 1996, and Ph.D. degree in Computer Science from Pontifical Catholic University of Parana, PUC-PR, Brazil. In 1989, he joined the Computer Science Department of the Ponta Grossa University, Brazil. In 1995, he also joined the Computer Science Department of the PUC-PR. His research interests are in the areas of document analysis and handwriting recognition.