Leave-One-Out-Training and Leave-One-Out-Testing Hidden Markov Models for a Handwritten Numeral Recognizer: The Implications of a Single Classifier and Multiple Classifications

Albert Hung-Ren Ko, *Student Member*, *IEEE*, Paulo Rodrigo Cavalin, Robert Sabourin, *Member*, *IEEE*, and Alceu de Souza Britto Jr.

Abstract—Hidden Markov Models (HMMs) have been shown to be useful in handwritten pattern recognition. However, owing to their fundamental structure, they have little resistance to unexpected noise among observation sequences. In other words, unexpected noise in a sequence might "break" the normal transmission of states for this sequence, making it unrecognizable to trained models. To resolve this problem, we propose a leave-one-out-training strategy, which will make the models more robust. We also propose a leave-one-out-testing method, which will compensate for some of the negative effects of this noise. The latter is actually an example of a system with a single classifier and multiple classifications. Compared with the 98.00 percent accuracy of the benchmark HMMs, the new system achieves a 98.88 percent accuracy rate on handwritten digits.

Index Terms—Hidden Markov Models, ensemble of classifiers, sequence, noise, leave one out, pattern recognition.

1 Introduction

The purpose of handwritten pattern recognition systems is to achieve the best possible recognition rate, and the Hidden Markov Model (HMM) is one of the most popular classification methods for sequential pattern recognition problems [3], [7], [24], [25], [28]. The objective of the HMM is to model a series of observable signals, and it is this signal-modeling ability that makes HMMs a better choice than other classification methods for recognition problems. As a stochastic process, an HMM is constructed with a finite number of states and a set of transition functions between two states, or over the same state [3], [24], [28]. Each state emits some observation(s), according to a codebook setting out corresponding emission probabilities. Such observations may be either discrete symbols or continuous signals.

One important result in research on HMMs is the emergence of ensemble of HMMs (EoHMM) as a promising scheme for improving HMMs [1], [9], [10], [11], [12], [13], [14]. This is because an ensemble of classifiers (EoC) is known to be capable of performing better than its best single classifier [6], [8], [19], [27]. These classifiers can be generated by changing the training set, the input features, or the parameters and architecture of the base classifiers [13]. The applicable ensemble creation methods include Bagging, Boosting, and Random Subspace. There may be other methods for creating HMM classifiers, based on the choice of features [10] for isolated handwritten images, and both column HMM classifiers and row HMM classifiers can be applied to enhance performance [4], [5]. The use of various topologies, such as the left-right HMM, and the semijump-in HMM, semi-jump-out HMM [11], and circular HMM [1], can also be applied.

However, even with EoHMMs, we see that the performance of HMMs is somehow more limited than in other types of classifiers [17], [18], such as Multilayer Perceptrons (MLPs) [21] or Support Vector Machines (SVMs) [20]. We believe that such a limited performance in some aspects might be attributed to the fundamental structure of HMMs, especially first-order HMMs. For first-order HMMs, all observations are supposed to stay at the same state or to move from current state to next state. This mathematical method allows us to match a given observation sequence with a model by examining the relationship between each current observation and its next observation.

- *A.H.-R. Ko is with the Joseph L. Rotman School of Management, University of Toronto, 105 St. George Street, Toronto, ON M5S 3E6, Canada. E-mail: drinkblue@gmail.com.*
- *P.R. Cavalin is with the Laboratoire LIVIA-Département de Génie de la Production Automatisée (GPA), 1100 rue Notre-Dame Ouest Room A-3600, Montréal, QC H3C 1K3, Canada. E-mail: cavalin@livia.etsmtl.ca.*
- *R. Sabourin is with the Département de Génie de la Production Automatisée, École de Technologie Supérieure, 1100 rue Notre-Dame Ouest, Montréal, QC H3C 1K3, Canada. E-mail: robert.sabourin@etsmtl.ca.*
- *A. de Souza Britto Jr. is with the Programa de Pós-Graduação em Informática Aplicada (PPGIa-PUCPR), Rua Imaculada Conceição, 1155 (Prado Velho), Curitiba (Paraná) CEP: 80215-901, Brazil. E-mail: alceu@ppgia.pucpr.br.*

Manuscript received 27 Aug. 2007; revised 24 Mar. 2008; accepted 2 Oct. 2008; published online 9 Oct. 2008.
Recommended for acceptance by D. Lopresti.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2007-08-0535.
Digital Object Identifier no. 10.1109/TPAMI.2008.254.

Nevertheless, when there is unexpected noise in a sequence, the mathematical structure of an HMM will not enable us to discover the relationship between each current observation and its next observation. Thus, this sequence will become unrecognizable to trained models. We explain this phenomenon as noise which "breaks" the transmission of states for a sequence of observations. In fact, it is possible that more than one observation in a sequence might be noise.

We face two types of problems: 1) unexpected noise occurring in a sequence of observations used for training; and 2) unexpected noise occurring in a sequence of observations used for testing. The former might bring about negative effects in a trained model, and the latter will make a sequence unrecognizable. It is difficult to handle these problems, since we have no information on whether or not an observation of a sequence is actually noise.

To resolve these problems, we presume that each observation has the same chance to be noisy. We thus propose a leave-one-out-training (LOOT-training) strategy: for a sequence of M observations, we leave an observation out of the original sequence and make a new sequence without this observation. We then apply the same process on all observations in this sequence and create M new sequences. We use all M sequences and the original sequence for training. For testing, we carry out the same process for each observed sequence, and create M new sequences from one original sequence for leave-one-out-testing (LOOT-testing). We sum up the likelihood of all new M sequences to make a final decision. Note that this is actually an example of a system with a single classifier and multiple classifications. The key questions that need to be addressed are the following:

1. Will leave-one-out-training HMMs perform better than the original HMMs?
2. Will leave-one-out-testing HMMs perform better than the original HMMs?

To answer these questions, we constructed leave-one-out-training and leave-one-out-testing HMMs (LOOT-HMM), and carried out an experiment on the handwritten digit recognition problem. We used the HMM-based handwritten numeral recognizer in [4], [5], which includes a numeral string segmentation stage and a single character verification stage (Fig. 1). In this paper, we focus on improving the verification stage to recognize isolated handwritten digits. At this stage, column and row HMM classifiers are used to enhance classification accuracy, and the sum of the outputs from the single best column HMM and the single best row HMM constitutes the final decision. With this system, we were able to improve verification by constructing a LOOT-HMM on both column HMM classifiers and row HMM classifiers.

It is important to note that the LOOT-HMM is different from the EoHMM, in that it does not need to train more than one classifier. However, the nature of leave-one-out sequences can generate multiple subsamples for testing, and thus the LOOT-HMM is similar to the EoHMM in the sense that it also relies on multiple classifications. Nevertheless, since the LOOT-HMM requires only one classifier, there is no cost associated with classifier selection, as in the multiple classifier system. Furthermore, the cost of classifier training for the LOOT-HMM would be much lower than that for the EoHMM if only LOOT-HMM testing is implemented. We will see later that, even though the LOOT-HMM has a lower time cost, it actually has a better performance than the EoHMM.

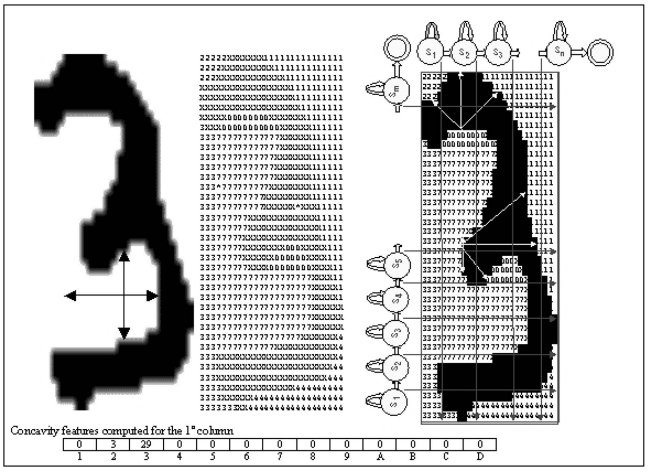

Fig. 1. The benchmark HMM classifiers in [4], [5]: Features are extracted from each column and each row, a column HMM classifier and a row HMM classifier are thus constructed for isolated handwritten numeral recognition.

The paper is organized as follows: first, we present the main concepts underlying HMMs, and briefly discuss the main techniques; then, we introduce the basic concepts of LOOT-HMMs in Section 3; in Section 4, we report on experiments we carried out on the NIST SD19 handwritten numeral database. A discussion and our conclusion are presented in the final sections.

2 HMM Structure, State, and Transition Models

The HMM is one of the most useful techniques for pattern recognition, especially for speech recognition and handwritten recognition [4], [5], [7], [9], [24], [25]. The objective of the HMM is to model a real-world process containing a series of observable signals. The signal might be discrete or continuous, pure or noisy. The HMM characterizes such real-world signals via a statistical approach: The signal can be well characterized as a parametric random process, and the parameters of the stochastic process can be determined in a precise, well-defined manner [24], [25].

As a stochastic process, the HMM is a construct with a finite number of states $S_1, S_2, \ldots, S_N$ and a set of transition functions aij between two states or over the same state. At each point in time, only a state exists, but states can change from one to another via a transition probability. Each state i emits some observation according to a codebook with a corresponding emission probability b_i. Given a series of observations $O = O_1, O_2, \ldots, O_T$, the function of the HMM is to determine the original states or models.

To carry out this task, an HMM must first be constructed to describe observed phenomena well, according to a certain class. In other words, each class would have its own HMMs. An observation might consist of either

discrete symbols or continuous signals. Discrete symbols can be characterized as quantized vectors in its codebook, and so vector quantization is needed. In contrast, continuous signals are represented by samples from a continuous waveform. More precisely, five kinds of parameters can alter the structure of an HMM:

1. the number of states in the model N;
2. the number of distinct observation symbols per state M;
3. the state transition probability distribution $a_{ij} = P[q_{t+1} = S_j \mid q_t = S_i]$;
4. the observation symbol probability distribution $b_i(k) = P[v_k \mid q_t = S_i]$ for codebook k; and
5. the initial state distribution $\pi_i = P[q1 = S_i]$. Note that we implement discrete HMMs in our experiment.

To train and recognize signals using HMMs, there are three fundamental problems of HMM design: 1) evaluation of the probability of a sequence of observations given a specific HMM; 2) determination of a best sequence of model states; and 3) adjustment of model parameters so as to best account for observed signals, including the observation symbol probability distribution $b_i(k)$ and the state transition probability distribution a_{ij}. Meanwhile, the number of states and the number of distinct observation symbols per state are also important issues for HMM, but they are often heuristically defined.

To evaluate the probability of a sequence of observations, the HMM must consider how well a given model matches a given observation sequence. This can be achieved by a forward-backward procedure, a calculation step which can also contribute to solving the following two problems: determination of the best sequence of model states, and the adjustment of model parameters.

To determine the best sequence of model states, the solution is to find an optimal state sequence with given observations. The most widely used criterion is to find the single best state sequence, i.e., to maximize the path probability. This can be achieved by dynamic programming methods, such as the Viterbi algorithm [24], [25], which will determine the probability of the observation for each model. However, when the length of the observation is unknown, the issue might become how many models should fit into the sequence. The Level Building Algorithm (LBA) solves this problem by finding the sequence of HMMs that best matches the patterns of unknown lengths [24], [25]. In our experiment, we implement the Viterbi algorithm for HMM testing.

For the adjustment of model parameters, there is no known way to analytically solve for a model which maximizes the probability of the observation sequence, since an optimization surface is complex and has many local maxima. However, it can be re-estimated to obtain a better approximation using the Expectation-Maximization (EM) algorithm. The HMM implements the Baum-Welch (BW) algorithm, which is essentially the same as the EM, to calculate transition probabilities [24], [25]. This algorithm is an implementation of EM, and can guarantee that a model will converge to a local maximum of the observation probability according to the maximum likelihood criterion. We use the Baum-Welch algorithm for HMM training in our experiments.

Concerning the problem of the length of each HMM, it is shown that the possible number of HMM states can be calculated by duration statistics from training data, which take into account the mean length and variance of all observation sequences. Another argument might be that an observation is better produced by the transition of states than by the states themselves. It is also argued that an empty transition in which no observation can be obtained might help the stochastic process, as well as the space state and the end state [4], [5]. This is the mechanism we have chosen to implement.

Following a brief overview of HMMs, we introduce the three new schemes for the implementation of the LOOT-HMM in the next sections.

3 Generation of LOOT-HMMs

The LOOT-HMM has three components: codebook optimization, LOOT-training, and LOOT-testing. We first explain the codebook optimization process using a clustering validity index, and then we explain the leave-one-out method for creating sequences of observations in HMM training and testing.

3.1 Codebook Optimization Using Xie-Beni (XB) Index

The first step in LOOT-HMM is codebook optimization. In general, an HMM codebook is generated from a vector quantization procedure, and each code word can be regarded as a centroid of a cluster in the feature space. The fitness of clustering depends on a number of different factors, such as clustering methods and the number of clusters. In this work, we focus on optimizing codebooks by finding the optimal number of clusters, i.e., by finding the optimal codebook size. In order to select the optimal number of clusters, we need to perform clustering with different numbers of clusters and evaluate them with using a clustering validity index, and then select the clustering with the best clustering validity index. For this purpose, a clustering validity index is designed to evaluate clustering results, as well as to assign a level of fitness to those results. Note that a clustering validity index is not a clustering algorithm in and of itself, but a measure with which to evaluate the results of clustering algorithms and give an indication of the partitioning that best fits a data set. Note, then, that a clustering validity index is independent of clustering algorithms and data sets.

We use the XB index as the clustering validity index for codebook optimization. This index [2], [15], [16], [23] was originally a fuzzy clustering validity index. For a fuzzy clustering scheme, suppose we have the data set $X = \{x_i, 1 \le i \le N\}$, where N is the number of samples with centroids v_j of clusters c_j, $1 \le j \le nc$, where nc is the total number of clusters. We seek to define the matrix of membership $U = [u_{ij}]$, where u_{ij} denotes the degree of membership of the sample x_i in the cluster c_j. To define the XB index, the sum of the squared errors must first be defined for fuzzy clustering, which is achieved as follows:

$$J_m(U, V) = \sum_{i=1}^{N} \sum_{j=1}^{nc} (u_{ij})^m \|x_i - v_j\|^2, \quad (1)$$

where $1 \le m \le \infty$. In general, we use J_1 for the calculation. U is a partition matrix of fuzzy membership $U = [u_{ij}]$, and V

is the set of cluster centroids $V = [v_i]$. In addition, the minimum intercluster distance d_{min} must also be defined, which is achieved as follows:

$$d_{min} = min_{i,j}\|v_i - v_j\|. \quad (2)$$

If we generate M clusters from the data, then the XB index can be defined as

$$XB = \frac{J_m}{M \times (d_{min})^2}. \quad (3)$$

The XB index is designed to measure the fitness of fuzzy clustering, but it is also suitable for crisp clustering. The XB index has been mathematically justified in [30]. The lower the value of the XB index, the better the clustering should be. We can use the XB index to scan the data and determine the number of clusters with the smallest XB index. This number is then chosen to be the codebook size, and codewords are generated using simple K-Means clustering.

We illustrate more details of codebook optimization below. Given a data set of $X = \{x_i, 1 \leq i \leq N\}$, where N is the number of samples, and defining a possible range M for the numbers of clusters j, $1 \leq j \leq M$, the XB index should give the fitness $F_t(j)$ for these M clusterings, with $1 \leq j \leq M$. Because of the large size of the data set in question, we used a smaller data set with N_s samples extracted from N samples for clustering goodness evaluation, $N_s = \eta\dot{N}$, where η is the proportion of samples used. The selection of the N_s sample is conducted just to make the machine work, and at the same time to have much data as possible.

Assuming that we intend to select the best clustering to determine the optimal codebook size, then this clustering could be selected in accordance with the minimum XB index $F_{\hat{t}}$, where $F_{\hat{t}} = minF_t(j), 1 \leq j \leq M$. The selected number of clusterings then serves as the size of the codebook of our HMM classifier. The selected codebook size is used again for the clustering on all N samples, with the result that the respective codebooks are generated.

Note that the XB index is one of a few clustering indices which can be implemented in codebook optimization, the others being the R-squared (RS) index, the Root-Mean-Square Standard Deviation (RMSSTD) index, Dunn's Index, the PBM index, and the Davies-Bouldin (DB) index [17], [18]. We chose the XB index for theoretical reasons as well as for practical ones. From a theoretical perspective, Dunn's Index, the DB index, and the XB index have been proved mathematically, but the use of Dunn's index could be relatively time consuming for a large data set. From a practical perspective, a clustering validity index must have several optima which can depict a data set at multiple levels of granularity, and the XB index is found to have this desirable property in our problem context [17], [18]. Consequently, we chose the XB index for codebook selection.

After the codebooks have been created, we apply the leave-one-out process for HMM training.

3.2 Leave-One-Out Training for HMMs

We use the Viterbi algorithm for standard HMM testing [25], but also implement a LOOT-testing process as a metatesting. The leave-one-out process for HMM training is a simple one, which generates several sequences of observations from one sequence of observations. Suppose that, for any sequence S_i, we have the observations $o_1, o_2, \ldots, o_{M_i}$, where M_i is the number of observations for the sequence S_i.

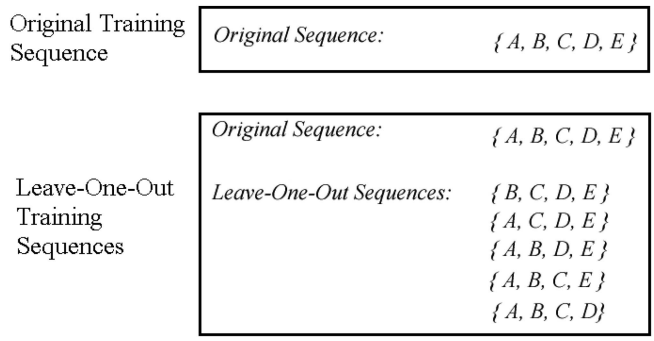

Fig. 2. The process of LOOT-training for HMMs: The sequence leaves out each observation once and forms a new sequence for training.

From this original sequence of observations, we then generate new sequences of observations by simply removing one observation at a time (Fig. 2). The longer the original sequence, the more new sequences it will generate.

Given a set of $S = \{S_i, 1 \leq i \leq N\}$, where N is the number of sequences and each sequence has M_i observations, an original sequence can generate M_i new sequences, and so, for all N sequences, the total number of sequences is

$$\sum_{1 \leq i}^{N} M_i. \quad (4)$$

For HMM training, all new sequences, as well as the original sequences, are used for training, giving us a total of

$$\left(\sum_{1 \leq i}^{N} M_i\right) + N$$

sequences for training.

LOOT-training gives us more training samples. Furthermore, it may reduce noise from, or induce noise into, a sequence. If it reduces the noise in a sequence, then HMMs have a more reliable sample for training; if it induces noise in a sequence, then HMMs can be trained to recognize noisy sequences in testing. LOOT-training is thus meant to make HMMs more robust. As a by-product, it also partially solves the problem of insufficient samples.

After LOOT-training has been completed, the HMMs are created and ready to classify any sequence tested. However, before doing this, we need to apply the LOOT-testing process on the sequences to be tested.

3.3 Leave-One-Out Testing for HMMs

LOOT-testing is similar to LOOT-training. The only difference is that, for testing, the leave-one-out process is applied on test sequences instead of on training sequences.

In a test sequence $S_t i$, we have the observations $o_1, o_2, \ldots, o_{m_i}$, where m_i is the number of observations for the sequence $S_t i$. Again, we generate new sequences of observations from the original sequence of observations by simply eliminating one observation at a time (Fig. 3).

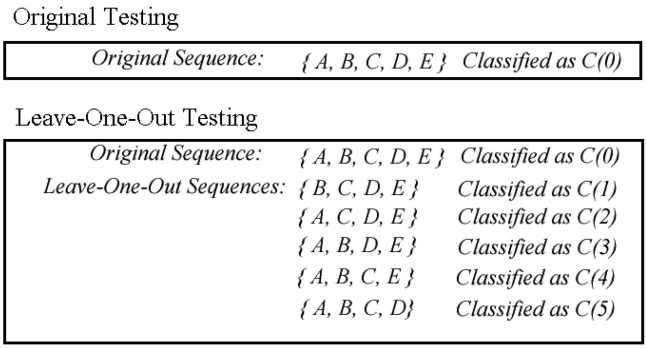

Fig. 3. The process of LOOT-testing for HMMs: The sequence leaves out each observation once and forms a new sequence for testing, and HMMs will assign a likelihood of each class for each new sequence. The final decision is made based on the sum of the likelihoods of each class.

After new sequences have been generated, all new m_i sequences and the original test sequence are submitted for testing. There are two schemes for generating final outputs based on these sequences. The first scheme (a) involves simply combining the likelihood of all sequences (all new m_i sequences and the original test sequence) for each class, and selects the class with the highest likelihood. This scheme uses a mechanism similar to the SUM rule in multiclassifier systems, and relies on the fact that most classifications will make diverse errors and that such errors can be reduced through the combination of probability, likelihood, or votes from different classifications. Method (a) forces a sequence of observations to ignore one single observation at a time, and then combines the likelihood of each sequence. We regard this method as a hard scheme for LOOT-testing, because all new m_i sequences and the original test sequence will affect the final output. The second scheme (b) involves directly selecting the class with the highest likelihood among all the classifications from all sequences, i.e., among all new m_i sequences and the original test sequence. To illustrate, if there are C classes, then the overall sequences produce $(m_i + 1) \times C$ likelihoods $L_j(k)$, $1 \leq j \leq C$ and $1 \leq k \leq m_i + 1$. By directly selecting the class C_j as $\arg\max L_j(k)$, we can decide on the most possible class, taking into account that one of the observations might be noise. Although this scheme seems similar to the MAX rule in multiclassifier systems, there is another interpretation of it if we remember that the purpose of LOOT-testing is to reduce the adverse effect of possible noise in test patterns. In the traditional HMM classification scheme, the objective is to determine the class C_j with the highest likelihood among all classes C_j, $1 \leq j \leq C$ from one sequence of observations. LOOT-testing generates m_i new sequences for classification, and, since all observations have the same probability of being noise, all new sequences will have the same importance. By selecting the class with the highest likelihood for each class, method (b) allows a sequence of observations to skip any single observation in order to achieve the highest likelihood. Then, by comparing the highest likelihood of each class, this method can assign the best class to the sequence by taking into consideration that one observation might be noise. As a result, method (b) suggests that there is no need to constrain the comparison of likelihoods to one class, and it is justifiable to compare the likelihoods among all classes from all sequences at the same time. We regard method (b) as a soft scheme for LOOT-testing, because only the best sequence among all new m_i sequences and the original test sequence will affect the final output. We denote method (a) as LOOT-testing(SUM) and method (b) as LOOT-testing(MAX).

Also note that there are two premises in LOOT-testing. First, we presume that we are unable to distinguish noise from a normal signal in a sequence of observations. Thus, each observation is treated with an equal chance of being noisy. Second, we presume that a sequence is, in most cases, still recognizable without one of its observations. Based on these two premises, we can remove a single observation from a sequence of observations and submit it for classification. When we repeat this process, we submit a sequence of observations for classification multiple times by removing different observations.

As in LOOT-training, LOOT-testing generates several test sequences from a single test sequence. If the leave-one-out process takes away a true noise, then the new sequence has a better chance of being correctly classified. However, if the leave-one-out process takes away a normal observation, then all we can hope is that the new sequence will still be recognizable without the removed observation.

The latter can be true when an omitted observation is not the only observation emitted by the current state. In other words, if the current state emits more than one observation, then the new sequence has a chance of being correctly recognized. In contrast, if the current state emits only one observation, i.e., the current state just moves from the previous state and will then move to the next state right away, then the new sequence might be unrecognizable.

Nevertheless, if there are more recognizable new sequences than unrecognizable new sequences, then the final classification result is likely to be correct. This might be the case when states are likely to undergo self-transition and emit more than one observation in a sequence. Therefore, if the number of observations in a sequence is significantly higher than the number of states, then the leave-one-out process might work well.

To verify whether or not the leave-one-out process will work for HMMs, we perform some experiments on a benchmark database.

4 Experiments with HMMs

The experimental data were extracted from $NISTSD19$ as a 10-class handwritten numeral recognition problem. As a result, there is an HMM model for each class, and 10 HMM models for 1 HMM classifier. Five databases were used: The training set with 150,000 samples $(hsf\_\{0-3\})$ was used to create 2 HMM classifiers, one being the column HMM classifier and another being the row HMM classifier. The large size of a data set for training can lead to a better recognition rate for each individual classifier. For codebook size selection evaluated by clustering validity indices, due to the extremely large data set (150,000 images are equivalent to 5,048,907 columns and 6,825,152 rows, with 47 features per column or per row), we use only the first 10,000 images from the training data set to evaluate the quality of the clustering, and they are equal to 342,910 columns and 461,146 rows. However,

TABLE 1
The Number of States for Base HMM Classifiers with the Codebook Size of 256 (Column HMMs (Col-S) and Row HMMs (Row-S))

Digit	0	1	2	3	4	5	6	7	8	9
Col-S	13	5	14	14	15	13	15	14	14	15
Row-S	15	16	17	20	19	20	19	19	21	22

codebooks were generated with the whole training set. The training validation set of 15,000 samples was used to stop HMM training once the optimum had been achieved. Using the best solution from this archive, the test set containing 60,089 samples ($hsf\_\{7\}$) was used to evaluate the accuracy of EoCs.

Each column HMM used 47 features obtained from each column, and each row HMM used 47 features obtained from each row (see Fig. 1). The features were extracted by the means described in [4], [5], and K-Means was used for vector quantization to generate codebooks for HMMs. The number of HMM states was optimized by the method described in [29]. The HMMs were trained by the Baum-Welch algorithm [24], [25]. The benchmark HMM classifiers used 47 features, with a codebook size of 256 [4], [5]. Note that optimization of the number of states was carried out using the method in [29] (see Table 1). For benchmark column HMM, we have a recognition rate of 97.60 percent, and for benchmark row HMM, the classification accuracy was about 96.76 percent, and the combination of benchmark column HMM and benchmark row HMM achieved a rate of 98.00 percent. Note that the combination of column HMM and row HMM is simply the combination of likelihoods of column HMM and row HMM for each class (e.g., weighted sum of log likelihoods), and the class with the highest combined likelihood is chosen as the final output.

We describe the process of generating LOOT-HMMs in the sections below (Fig. 4).

4.1 Optimization for HMMs

To decide on suitable codebook sizes for HMMs, we carried out clustering analyses on HMM features. Because of the large data set, it is clear that we could not use the entire training set to perform the clustering. As a result, the first 10,000 images in the training set were used for clustering, these images containing 342,910 columns and 461,146 rows. We scanned the data with the number of clusters from 1 to 10,000.

We believe that 10,000 is a reasonable upper limit for the size of a codebook with 47 features. If all the features are binary, then we will have a dimension of 2^{47} in the clustering space. Moreover, most features are not binary, but real-value features. Since $2^{47} \gg 10{,}000$, we believe that the number of clusters is not overextended. Note that the optimal cluster number is highly dependent on the number of feature vectors per state.

For column HMMs, we determined that the optimal codebook size is 9,525; for row HMMs, the best codebook size is 9,783. Following the codebook optimization process, we optimized the number of states [29] (see Table 2). Note

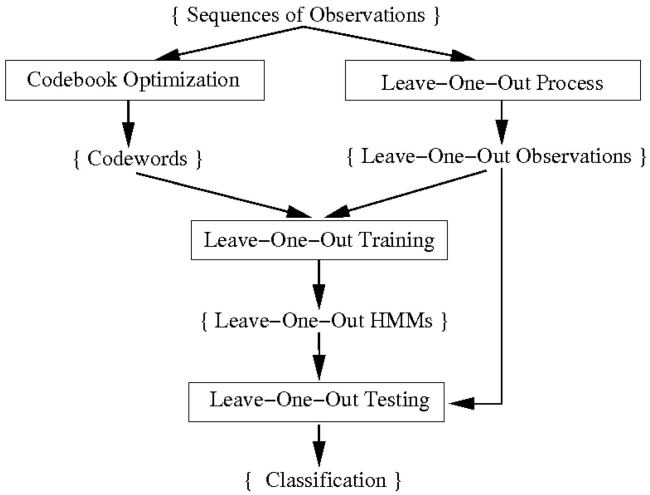

Fig. 4. The process of LOOT-training and LOOT-testing HMMs (note that this process applies to both column HMMs and row HMMs).

that the difference in the numbers of states between benchmark HMMs and LOOT-HMMs is relatively small, where most HMMs have a nearly identical number of states.

Note that the number of states is important, especially for LOOT-testing. Even though it is difficult to obtain direct proof, we suspect that the leave-one-out process might perform well only if the number of observations in a sequence is significantly greater than the number of states. We shall compare these two numbers later in our experiment.

4.2 Leave-One-Out Sequences

For sequences of column and row observations, the leave-one-out process significantly increases the number of sequences in both training samples and testing samples (Table 3).

Once these sequences have been generated, they can be used for HMM training and testing. However, it is interesting to see the number of observations generated for LOOT-testing (Table 4).

We note two things: first, row sequences have a larger number of observations than column sequences; second, the digit 1 has a relatively smaller number of observations of column sequences. Nevertheless, all these digits have a larger number of observations than their number of states (Table 2), which means each state generally emits more than one observation.

We then carried out LOOT-training and LOOT-testing for HMMs.

4.3 Column LOOT-HMMs and Row LOOT-HMMs

The optimization of codebooks and the numbers of states can improve the accuracy of column HMMs from 97.60 percent

TABLE 2
Number of States for Column HMMs (Col-S) and Row HMMs (Row-S) with Codebook Optimization

Digit	0	1	2	3	4	5	6	7	8	9
Col-S	14	5	14	14	15	13	15	14	14	16
Row-S	15	16	18	21	19	20	20	20	22	23

TABLE 3
The Number of Leave-One-Out Sequences for Column Training Samples and Row Training Samples

Digit	Column LOOT-HMMs Training Sequences	Column LOOT-HMMs Testing Sequences	Row LOOT-HMMs Training Sequences	Row LOOT-HMMs Testing Sequences
0	416,545 (15,000)	207,292 (5,893)	594,200 (15,000)	247,056 (5,893)
1	183,184 (15,000)	78,618 (6,567)	651,658 (15,000)	299,688 (6,567)
2	641,222 (15,000)	270,024 (5,967)	645,570 (15,000)	265,864 (5,967)
3	554,929 (15,000)	229,903 (6,036)	692,479 (15,000)	291,346 (6,036)
4	567,505 (15,000)	227,439 (5,873)	762,825 (15,000)	312,440 (5,873)
5	670,698 (15,000)	255,979 (5,684)	684,897 (15,000)	272,056 (5,684)
6	525,266 (15,000)	211,252 (5,900)	714,223 (15,000)	288,911 (5,900)
7	536,137 (15,000)	239,045 (6,254)	706,824 (15,000)	305,751 (6,254)
8	509,330 (15,000)	202,818 (5,889)	739,169 (15,000)	302,188 (5,889)
9	503,675 (15,000)	206,308 (6,026)	783,307 (15,000)	324,951 (6,026)

The number of original sequences is indicated in parentheses.

TABLE 4
The Average Number of Observations for Column HMMs (Col-Obs) and Row HMMs (Row-Obs)

Digit	0	1	2	3	4	5	6	7	8	9
Col-Obs	35.18	11.97	45.25	38.09	38.73	45.04	35.81	38.22	34.44	34.24
Row-Obs	41.92	45.64	44.56	48.27	53.20	47.86	48.97	48.89	51.31	53.92

TABLE 5
The Performance of Column LOOT-HMMs and Row LOOT-HMMs

	Original	Optimization	train	LT-ts (SUM)	LT-ts (MAX)	LT-ts(SUM)+ LT-tr	LT-ts(MAX)+ LT-tr
Column	97.60%	98.66%	98.69%	98.68%	98.74%	98.72%	98.76%
Row	96.76%	97.78%	98.22%	98.15%	98.15%	98.14%	98.23%
Combination	98.00%	98.79%	98.86%	98.86%	98.85%	98.84%	98.88%

LT-tr Stands for LOOT-Training; and LT-ts Stands for LOOT-Testing.

to 98.66 percent, and that of row HMMs from 96.76 percent to 97.78 percent. However, the use of LOOT-training does not greatly improve the accuracy of HMMs for column HMMs. The improvement is considered significant only on row HMMs. Note that, while increasing the number of training sequences might make HMMs more robust, it does not significantly enhance recognition rates for column HMMs (Table 6). It does, however, improve row HMMs (Table 7). Combined HMMs have a similar statistical significance to column HMMs. This contrast between column HMMs and row HMMs also exists in LOOT-testing. After optimization, the improvement as a result of LOOT-testing is significant on row HMMs, but not on column HMMs.

Furthermore, LOOT-training and LOOT-testing seem to result in improvement on similar data points. When we apply LOOT-testing(MAX) after LOOT-training, we see that performances change from 98.69 percent to 98.76 percent for column HMMs, and from 98.22 percent to 98.23 percent for row HMMs (see Table 5). These are insignificant improvements, and it seems that robust HMM classifiers with LOOT-training can achieve similar performance with HMM classifiers with only LOOT-testing.

For the combination of column and row HMMs with LOOT-testing, note that, since row HMMs have a longer observation sequence than column HMMs, we have to weigh the likelihood output from HMMs with the length of the observation sequence for LOOT-testing(SUM), i.e., the number of added new sequences in LOOT-testing, so that column HMMs and row HMMs can have a similar weight in the final decision output. For LOOT-testing(MAX), since only the class with the highest likelihood is selected, no weighting is used.

Even without LOOT-testing, LOOT-training still brings about some improvement over the original optimized HMMs. Note, however, that these improvements could be regarded as insignificant for column HMMs, since LOOT-HMM does not statistically outperform HMM optimization (Table 6). There are three reasons for this: 1) It seems that there is not much improvement space for reducing noise in sequences; 2) there is too much noise, and the elimination of any single noise signal will not bring about significant improvement; and 3) the column HMM observations are too short, and thus the number of new sequences is not sufficient to make a significant improvement. For row HMMs, however, both LOOT-training and LOOT-testing show some significant improvement over the optimization set (Table 7). As we noted previously, row HMMs have longer observations and thus more sequences for classification and training, and this might be the reason for its significant ability to result in improvement. Nevertheless, note that LOOT-testing does not bring about further improvement after LOOT-training,

TABLE 6
The p-Value of Different Column LOOT-HMM Schemes

	Original	Optimization	LT-tr	LT-ts (SUM)	LT-ts (MAX)	LT-ts(SUM) +LT-tr
Optimization	$p < 0.0001$					
LT-tr	$p < 0.0001$	$p > 0.1$				
LT-ts(SUM)	$p < 0.0001$	$p > 0.1$	$p > 0.1$			
LT-ts(SUM) + LT-tr	$p < 0.0001$	$p > 0.1$	$p > 0.1$	$p > 0.1$		
LT-ts(MAX)	$p < 0.0001$	$p > 0.1$	$p > 0.1$	$p > 0.1$	$p > 0.1$	
LT-ts(MAX) + LT-tr	$p < 0.0001$	$p < 0.1$	$p > 0.1$	$p > 0.1$	$p > 0.1$	$p > 0.1$

LT-tr Stands for LOOT-Training; and LT-ts Stands for LOOT-Testing.

TABLE 7
The p-Value of Different Row LOOT-HMM Schemes

	Original	Optimization	LT-tr	LT-ts (SUM)	LT-ts (MAX)	LT-ts(SUM) +LT-tr
Optimization	$p < 0.0001$					
LOOT-train	$p < 0.0001$	$p < 0.0001$				
LT-ts(SUM)	$p < 0.0001$	$p < 0.0001$	$p > 0.1$			
LT-ts(SUM) + LT-tr	$p < 0.0001$	$p < 0.0001$	$p > 0.1$	$p > 0.1$		
LT-ts(MAX)	$p < 0.0001$	$p < 0.0001$	$p > 0.1$	$p > 0.1$	$p > 0.1$	
LT-ts(MAX) + LT-tr	$p < 0.0001$	$p < 0.0001$	$p > 0.1$	$p > 0.1$	$p > 0.1$	$p > 0.1$

LT-tr Stands for LOOT-Training; and LT-ts Stands for LOOT-Testing.

TABLE 8
The Recognition Rate of Each Numeral for Combined LOOT-Testing HMMs, Expressed as a Percentage

Digit	0	1	2	3	4	5	6	7	8	9
LT-tr + LT-ts(SUM)	98.00	98.26	99.73	98.92	99.46	98.50	99.07	99.06	98.93	98.54
LT-tr + LT-ts(MAX)	98.12	98.22	99.68	98.96	99.37	98.61	99.29	99.26	98.76	98.61

LT-tr + LT-ts Stands for LOOT-Training + LOOT-Testing; SUM Stands for SUM Combination Rule; and MAX Stands for MAX Combination Rule.

which means that the LOOT-testing process probably eliminates similar noise which could be reduced by LOOT-training. When we combine LOOT-training-testing column HMMs and LOOT-training-testing row HMMs, we achieve a 98.88 percent recognition rate on handwritten numerals. The performance is good compared with the recognition rate of 98.00 percent of the original combination of column and row HMMs. Note, however, that if we skip the LOOT-training process and only carry out LOOT-testing after optimization, we can still achieve a 98.86 percent recognition rate, while LOOT-training without LOOT-testing(MAX) can also achieve a recognition rate of 98.86 percent. Again, this confirms an overlapping effect from LOOT-testing and LOOT-training. Also note that the best result, 98.88 percent, is only slightly better than the best results that can be achieved with EoHMMs for the data set in question. This might hint that a metalearning approach—either with multiple classifiers or with multiple classifications—has its own limits. Further improvement should focus on the learning structure of the HMM, and not on metalearning methodologies.

Furthermore, since different digits have different average numbers of observations, it is worth looking at their performances separately (see Table 8). We note that the digit recognition rate varies from 99.73 percent to 98.00 percent for LOOT-training + LOOT-testing(SUM), and from 99.68 percent to 98.12 percent for LOOT-training + LOOT-testing(MAX). It seems that different digits face different challenges, and that the difficulties might be more than just noise in the sequences that we are aiming to reduce.

5 Discussion

The proposed LOOT-HMM is an easy and relatively effective scheme. The column HMMs achieve a 98.76 percent recognition rate, while the row HMMs have a 98.23 percent accuracy. More importantly, the combination of column and row HMMs improves performance, which reaches 98.88 percent.

The LOOT-HMM has three components: codebook optimization, LOOT-training, and LOOT-testing. While LOOT-training shows very little improvement, codebook optimization and LOOT-testing noticeably raise the recognition rate.

Although LOOT-training does not always lead to significant improvement, it has an advantage when there is an insufficient number of training sequences. The use of LOOT-training not only increases the robustness of HMMs, but it also resolves the problem of insufficient samples known as the curse of dimensionality. Note, however, that LOOT-training does not necessarily accelerate the classifier training process. In a case where the training sample has N sequences, for example, and where each sequence has an

TABLE 9
The Recognition Rate of Different Classification Schemes on NIST SD19

Year	Authors	Method	Result
2002	Oliveira et al. [21]	Multi-Layer Perceptrons	99.16%
2003	Britto et al. [5]	Hidden Markov Models	98.00%
2004	Oliveira et al. [22]	Support Vector Machines	99.20%
2006	Radtke et al. [26]	Multi-Layer Perceptrons	99.29%
2006	Milgram et al. [20]	Support Vector Machines	99.37%
2007	Ko et al. [17], [18]	Hidden Markov Models	98.86%
2008	Proposed Method	Hidden Markov Models	98.88%

average of X observations, traditional multiple classifier methods, such as EoHMM, could require training M classifiers with N sequences, while LOOT-training will train one classifier with $X \times N$ sequences. Since training a classifier with $X \times N$ sequences is as fast as training X classifiers with N sequences, if $M \leq X$, then LOOT-training will not necessarily be faster than EoHMM in terms of classifier training time. Nevertheless, it does offer advantages in terms of ensemble selection and ensemble optimization, since only a single classifier is used.

LOOT-testing offers improvement similar to LOOT-training. The LOOT-testing process uses a single classifier, but can generate multiple classifications. By using these multiple classifications, LOOT-testing might offer similar advantages to those of an EoC. For example, an EoHMM achieves 98.86 percent [17] on the same data set using a pool of 40 classifiers, i.e., 400 HMMs. Compared with the EoHMM scheme, our LOOT scheme uses only two classifiers, i.e., 20 HMMs, and thus costs less and has a similar performance. It might, therefore, be of great interest to further pursue this direction. Note that the use of LOOT-training alone seems to have effects similar to those of LOOT-testing (Tables 6 and 7). Without LOOT-testing, LOOT-training can still achieve 98.86 percent recognition rate, and vice versa. Note that if only LOOT-testing is implemented, then LOOT-HMM will have a time-saving advantage over EoHMM, because only one HMM classifier training is required.

LOOT-testing uses equal-weight artificial sequences (with one observation skipped) and original sequences. The reason behind this is that we attribute to each observation the same probability of error, and thus there is no reason to assign more weight to original sequences. However, this does not mean that such a weighting mechanism is optimal for the LOOT-HMM, and it will be in our interest to explore further weighting optimization on this classification scheme.

Note that there are still some potential opportunities for the acceleration of the LOOT-HMM. One reviewer suggests a time reduction scheme, stemming from the fact that many of the calculations are repeated because there are so many nearly identical sequences in the LOOT-HMM scheme. Instead of skipping one of the n observations in a sequence in order to obtain the likelihood of an leave-one-out sequence, it is plausible to store the likelihoods of being in the states without a skip, and use them to update the likelihoods of being in the states with a skip. It should be interesting to look at this in the future to further speed up the LOOT-HMM.

Considering other classification methods applied to the same data set, the KNN with 150,000 samples achieves a 98.57 percent accuracy [21], the MLP achieves 99.16-99.29 percent accuracy [21], [26], and the use of the SVM achieves a 99.30 percent recognition rate with a pairwise coupling strategy and a 99.37 percent recognition rate with the one-against-all strategy [20] (Table 9). Note that both the MLP and the SVM are relatively time consuming, while the KNN is a faster method for classifier training but less accurate. HMMs also require substantial training and are thus time consuming, but they could hardly perform better than at 99.00 percent accuracy, and thus are less accurate than the MLP or the SVM for the data set in question. In practice, it is well known that the accuracy of the HMM is inferior to other sophisticated classification methods, partly due to its weak resistance to noise. However, HMMs have a unique advantage in sequential signal recognition, because they can recognize and segment continuous signals at the same time, while other classification methods must rely on other well-designed and fine-tuned empirical signal segmentation schemes. Such signal segmentation schemes are often obtained through trial and error, and are not optimal. As a result, this disadvantage of other classification schemes makes HMMs an attractive choice for sequential pattern recognition.

6 Conclusion

In general, HMMs are constructed using a structure in which each state emits some observable symbols, and one state moves on to another state with a certain probability. From this, we obtain the probability of emission of each symbol from each state and the probabilities of transition among states, and then establish models. Finally, observation sequences are matched to these trained models. However, the fundamental structure of an HMM implies that the models have little resistance to unexpected noise in observation sequences.

To take into account the effect of unexpected noise occurring in a sequence of observations, we propose a leave-one-out methodology for HMM training and HMM testing. We find that LOOT-training could make HMMs slightly more robust, and LOOT-testing could enhance the accuracy of HMMs. This is especially significant when there are longer sequences of observations, such as in row HMMs.

LOOT-testing is actually an example of a system with a single classifier and multiple classifications. In our experiment, we found that these multiple classifications

are likely to perform better than a single classification. Compared with an EoC, the LOOT-testing strategy has several advantages: first, it does not require multiple classifiers, and this reduces the cost of classifier training considerably; second, it has no need for ensemble selection, and this eliminates the time required for the ensemble search.

We show that LOOT-HMMs can, in our experiment, perform as well as an EoHMM. Moreover, if only LOOT-testing is implemented, then LOOT-HMMs will have time-saving advantage over EoHMM. Yet, our method is limited by the uncertainty of the nature of observations, since the recognition rates achieved are still not close enough to our objective.

Our experimental study indicates that the LOOT-HMM has a positive impact. However, we still face the problem of requiring a theoretical proof on the effectiveness of the proposed approaches. Since the issue of noise in observation sequences is a complex problem, and it is generally impossible to distinguish noise from normal signals in such observations, this makes a theoretical proof quite challenging. Overall effectiveness might depend on density, spectrum, and type of noise, and this is a rather difficult issue to tackle. Nevertheless, it is possible that some related theoretical work might emerge in the future.

Our study also raises more questions, such as how to guarantee the optimality of the leave-out process, since the number of possible scenarios for this process is very large. The leave-one-out strategy presented in this paper is straightforward, however, further improvements can be made. We believe that this methodology can be greatly enhanced with theoretical studies on the distinguished nature between the observations and the potential noise, the influence of geometrical and topological constraints on the HMMs, better statistical studies to quantify the uncertainty of the nature of observations, and empirical studies on more real-world problems.

ACKNOWLEDGMENTS

The authors thank anonymous reviewers for valuable comments. Special thanks to Dr. Daniel Lopresti, who provided insightful feedbacks on the structure of this paper. This work was supported in part by grant OGP0106456 to Robert Sabourin from the NSERC of Canada.

REFERENCES

[1] N. Arica and F.T.Y. Vural, "A Shape Descriptor Based on Circular Hidden Markov Model," *Proc. 15th Int'l Conf. Pattern Recognition,* 2000.

[2] S. Bandyopadhyay and U. Maulik, "Non-Parametric Genetic Clustering: Comparison of Validity Indices," *IEEE Trans. Systems, Man, and Cybernetics, Part C,* vol. 31, no. 1, pp. 120-125, Feb. 2001.

[3] Y. Bengio, "Markovian Models for Sequential Data," *Neural Computing Surveys,* vol. 2, pp. 129-162, 1999.

[4] A. Britto Jr., "A Two-Stage HMM-Based Method for Recognizing Handwritten Numeral Strings," PhD thesis, Pontifical Catholic Univ. of Paraná, 2001.

[5] A. Britto, R. Sabourin, F. Bortolozzi, and C.Y. Suen, "Recognition of Handwritten Numeral Strings Using a Two-Stage HMM-Based Method," *Int'l J. Document Analysis and Recognition,* vol. 5, no. 2, pp. 102-117, 2003.

[6] G. Brown, J. Wyatt, R. Harris, and X. Yao, "Diversity Creation Methods: A Survey and Categorisation," *Int'l J. Information Fusion,* vol. 6, no. 1, pp. 5-20, 2005.

[7] T.G. Dietterich, "Machine Learning for Sequential Data: A Review," *Structural, Syntactic, and Statistical Pattern Recognition,* pp. 15-30, Springer-Verlag, 2002.

[8] D. Eppstein, "Fast Hierarchical Clustering and Other Applications of Dynamic Closest Pairs," *Proc. Ninth ACM-SIAM Symp. Discrete Algorithms,* pp. 619-628, 1998.

[9] S. Gunter and H. Bunke, "Creation of Classifier Ensembles for Handwritten Word Recognition Using Feature Selection Algorithms," *Proc. Eighth Int'l Workshop Frontiers in Handwriting Recognition,* pp. 183-188, 2002.

[10] S. Gunter and H. Bunke, "Generating Classifier Ensembles from Multiple Prototypes and Its Application to Handwriting Recognition," *Proc. the Third Int'l Workshop Multiple Classifier Systems,* pp. 179-188, 2002.

[11] S. Gunter and H. Bunke, "A New Combination Scheme for HMM-Based Classifiers and Its Application to Handwriting Recognition," *Proc. 16th Int'l Conf. Pattern Recognition,* vol. 2, pp. 332-337, 2002.

[12] S. Gunter and H. Bunke, "Ensembles of Classifiers for Handwritten Word Recognition," *Int'l J. Document Analysis and Recognition,* vol. 5, no. 4, pp. 224-232, 2003.

[13] S. Gunter and H. Bunke, "New Boosting Algorithms for Classification Problems with Large Number of Classes Applied to a Handwritten Word Recognition Task," *Proc. Fourth Int'l Workshop Multiple Classifier Systems,* pp. 326-335, 2003.

[14] S. Gunter and H. Bunke, "Offline Cursive Handwriting Recognition—On the Influence of Training Set and Vocabulary Size in Multiple Classifier Systems," *Proc. 11th Conf. Int'l Graphonomics Soc.,* 2003.

[15] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On Clustering Validation Techniques," *J. Intelligent Information Systems,* vol. 17, pp. 2-3, 2001.

[16] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "Clustering Validity Checking Methods: Part II," *ACM SIGMOD Record,* vol. 3, no. 3, pp. 19-27, 2002.

[17] A.H.R. Ko, R. Sabourin, and A. Britto Jr., "A New HMM-Based Ensemble Generation Method for Character Recognition," *Proc. Int'l Workshop Multiple Classifier Systems,* pp. 52-61, 2007.

[18] A.H.R. Ko, R. Sabourin, and A. Britto Jr., "Ensemble of HMM Classifiers Based on the Clustering Validity Index for a Handwritten Numeral Recognizer," *Pattern Analysis and Applications J.,* 2008, doi 10.1007/s10044-007-0094-6.

[19] L.I. Kuncheva, M. Skurichina, and R.P.W. Duin, "An Experimental Study on Diversity for Bagging and Boosting with Linear Classifiers," *Int'l J. Information Fusion,* vol. 3, no. 2, pp. 245-258, 2002.

[20] J. Milgram, M. Cheriet, and R. Sabourin, "Estimating Accurate Multi-Class Probabilities with Support Vector Machines," *Proc. Int'l Joint Conf. Neural Networks,* pp. 1906-1911, 2005.

[21] L.E.S. Oliveira, R. Sabourin, F. Bortolozzi, and C.Y. Suen, "Automatic Recognition of Handwritten Numerical Strings: A Recognition and Verification Strategy," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 11, pp. 1438-1454, Nov. 2002.

[22] L.E.S. Oliveira and R. Sabourin, "Support Vector Machines for Handwritten Numeral String Recognition," *Proc. Ninth Int'l Workshop Frontiers in Handwriting Recognition,* pp. 39-44, Oct. 2004.

[23] M.K. Pakhira, S. Bandyopadhyay, and U. Maulik, "Validity Index for Crisp and Fuzzy Clusters," *Pattern Recognition,* vol. 37, no. 3, pp. 487-501, 2004.

[24] L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. IEEE,* vol. 77, no. 2, pp. 257-286, 1989.

[25] L.R. Rabiner and B.H. Juang, *Fundamentals of Speech Recognition.* Prentice-Hall, 1993.

[26] P. Radtke, T. Wong, and R. Sabourin, "An Evaluation of Over-Fit Control Strategies for Multi-Objective Evolutionary Optimization," *Proc. IEEE World Congress on Computational Intelligence—Int'l Joint Conf. Neural Networks,* 2006.

[27] D. Ruta and B. Gabrys, "Classifier Selection for Majority Voting," *Int'l J. Information Fusion,* pp. 63-81, 2005.

[28] P. Smyth, D. Heckerman, and M.I. Jordan, "Probabilistic Independence Networks for Hidden Markov Probability Models," *Neural Computation,* vol. 9, pp. 227-269, 1997.

[29] X. Wang, "Durationally Constrained Training of HMM without Explicit State Durational," *Proc. Inst. of Phonetic Sciences,* vol. 18, pp. 111-130, 1994.
[30] X.L. Xie and G. Beni, "A Validity Measure for Fuzzy Clustering," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 8, pp. 841-847, Aug. 1991.

Albert Hung-Ren Ko received the MSc degree in artificial intelligence and pattern recognition from the Université Pierre et Marie Curie (Paris, France), in 2002, the MSc degree in advanced calculation of structure from the École Normale Supérieure de Cachan (France), in 2003, and the PhD degree in pattern recognition from the École de Technologie Supérieure, Université du Québec (Montreal), in 2007. He joined C.u.O. Vogt Institute of Brain Research (Düsseldorf, Germany) in 2002 in an attempt to model schizophrenics' reinforcement behavior in Wisconsin Card Sorting Test with a neural network, then joined the Image, Vision, and Artificial Intelligence Lab (Montreal, Québec) and was responsible for a distributed computing system. His research interests are ensemble classification methods, small world structure, and neural networks. He is a student member of the IEEE and the IEEE Computer Society.

Paulo Rodrigo Cavalin received the MSc degree in computer science from the Pontifícia Universidade Católica do Paraná (PUC-PR, Brazil) in 2005. In 2006, he began studies toward the PhD degree in pattern recognition at the École de Technologie Supérieure, Université du Québec. His research interests include handwriting recognition, Hidden Markov Models, ensemble classification methods, and incremental learning.

Robert Sabourin joined the Physics Department of Montreal University in 1977, where he was responsible for the design, experimentation, and development of scientific instrumen-tation for the Mont Mégantic Astronomical Observatory. His main contribution was the design and the implementation of a microprocessor-based fine tracking system combined with a low-light level CCD detector. In 1983, he joined the staff of the École de Technologie Supérieure, Université du Québec, in Montréal, where he cofounded the Department of Automated Manufacturing Engineering, where he is currently a full professor and teaches pattern recognition, evolutionary algorithms, neural networks, and fuzzy systems. In 1992, he also joined the Computer Science Department of the Pontifícia Universidade Católica do Paraná (Curitiba, Brazil), where he was coresponsible for the implementation, in 1995, of a master's program and, in 1998, of a PhD program in applied computer science. Since 1996, he has been a senior member of the Centre for Pattern Recognition and Machine Intelligence (CENPARMI, Concordia University). Dr. Sabourin is the author (and coauthor) of more than 240 scientific publications, including journals and conference proceedings. He was cochair of the program committee of the Conférence Internationale Francophone sur l'Écrit et le Document, Québec, Canada (CIFED '98) and the Ninth International Workshop on Frontiers in Handwriting Recognition, Tokyo, Japan (IWFHR '04). He was nominated as conference cochair of the Ninth International Conference on Document Analysis and Recognition (ICDAR '07) that was held in Curitiba, Brazil, in 2007. His research interests are the areas of handwriting recognition, signature verification, watermark, and bio-cryptography. He is a member of the IEEE and the IEEE Computer Society.

Alceu de Souza Britto Jr. received the MSc degree in industrial informatics from the Federal Center for Technological Education of Paraná, Brazil, in 1996, and the PhD degree in computer science from the Pontifícia Universidade Católica do Paraná (PUCPR, Brazil). In 1989, he joined the Computer Science Department of the Ponta Grossa University, Brazil. In 1995, he also joined the Computer Science Department at PUCPR. From July 1998 to July 2000, he worked on handwriting recognition area in Montreal (Canada) at the laboratories of the Centre for Pattern Recognition and Machine Intelligence (CENPARMI-Concordia University) and École de Technologie Supérieure, Université du Québec. From 2004 to 2005, he was the director of the Computer Science Undergraduate Course at PUCPR. In 2001, he joined the Post-Graduate Program in Applied Informatics of PUCPR. His research interests are in the areas of pattern recognition, computer vision, document analysis, and handwriting recognition.