Online active supervision of an evolving classifier for customized-gesture-command learning

Manuel Bouillon, Eric Anquetil

Online Active Supervision of an Evolving Classifier for Customized-Gesture-Command Learning

Manuel Bouillon[a,b,*], Eric Anquetil[a,b]

[a] *INSA de Rennes, Avenue des Buttes de Coesmes, 35043 Rennes, France*
[b] *IRISA, CNRS UMR 6074, Campus de Beaulieu, 35042 Rennes, France*

Abstract

Touch sensitive interfaces enable new interaction methods, like gesture commands. For users to easily memorize more than a dozen of gesture commands, it is important to enable gesture set customization. The classifier used to recognize drawn symbols must hence be customizable – able to learn from very few data – and evolving – able to learn new classes on-the-fly and improve during its use. The objective of this work is to obtain a gesture command system that cooperates as best as possible with the user: that learns from its mistakes without soliciting the user too often. This paper presents a novel approach for the online active learning of gesture commands, with three contributions. The *IntuiSup* supervisor monitors the learning process and user interactions. The Evolving Sampling by Uncertainty (ESU) algorithm enables to maintain the error/interaction compromise over time. The Boosted-ESU (B-ESU) method optimizes interaction impact to fasten system learning speed. The efficiency of our approach is evaluated on the publicly available ILG Data Base of gesture commands. Experimentation shows the effectiveness of the supervision strategy and improvements both in term of accuracy and learning speed.

Keywords: Handwritten Gesture Command, User Interaction, Online Active Supervision

1. Introduction

With the increasing use of touch sensitive screens, human-computer interactions are evolving. New interaction methods have been designed to take advantage of the new potential of interaction offered by those interfaces. Among them, a new concept has recently appeared: to associate commands to gestures (see Figure 1). Those gesture commands [1, 2] enable users to execute various actions simply by drawing the associated symbols. Previous studies [3, 4] have shown that enabling customization is essential to help user memorization of gestures. In order to use such gesture commands, a handwritten gesture recognition system is required. Moreover, if gestures are customized, the classifier has to be flexible and able to learn with very few data samples.

Gesture commands give rise to a cross-learning situation where the user has to learn and memorize the gesture set, and the classifier has to learn and recognize drawn gestures. Enabling customization of the gesture commands is essential for user memorization. On the other hand, enabling users to choose their own gestures may lead to commands with similar or strange gestures that are hard to recognize by the classifier. We must assist the user to avoid similar gestures during this definition step, by providing a dynamic feedback on potential confusion risks [5]. Moreover, we can't expect users to draw much more than a few gesture samples per class, so the classifier must be able to learn with very few data. A lot of very powerful classifiers exist, like Support Vector Machines or Neural Networks for example. However, such offline systems don't improve during their use and don't evolve with the user writing style. Indeed, novice users usually draw gestures slowly and carefully, but as they become more and more expert, users draw their gestures more fluidly and rapidly. In that case, we want the classifier to adapt to the user, and not the other way round! More flexibility in a classifier requires an online system, a system that learns from the run-time data flow and adapt to its changes.

Evolving classification systems have explored in the last decade to meet the need for classifiers that work in changing environments [6, 7]. They use online learning algorithms to adapt to the data flow and cope with class adding (or removal) at run-time. Some evolving template matching classifiers exist, like the $1 classifier [8] for instance; and some simple systems, such as nearest neighbors classifiers, or Bayesian classifiers, can be made incremental easily, but their performances are quite limited on complex tasks. This work is based on a much more powerful classifier – namely *Evolve* [9] – which is an evolving first order fuzzy inference system [7]. It can start learning from scratch, and then learns incrementally in real time from the run-time data flow, to adapt its model and to improve its performance during its use.

The online learning algorithm is a supervised algorithm that requires labeled data. However, labeling data without errors require user interactions, and soliciting the user has a cost. Labeling interactions are annoying for the user and reduces gesture-commands fluidity. In this context, the optimization of user-

*Corresponding author
Email addresses: manuel.bouillon@irisa.fr (Manuel Bouillon), eric.anquetil@irisa.fr (Eric Anquetil)

system interaction is complex (user and system cross-learning), but essential for gesture command recognition. This paper focuses on the online active supervision of the evolving classifier learning, to train the classifier during its use at minimal cost.

This paper presents a novel online active learning supervisor – namely *IntuiSup* – which optimizes user-system interactions in online learning situations. The *IntuiSup* supervisor combines two supervision strategies:

- an implicit supervision strategy,
- an explicit supervision strategy.

Two explicit supervision strategies are presented in this paper:

- ESU: Evolving Sampling by Uncertainty,
- B-ESU: Boosted Evolving Sampling by Uncertainty.

First contribution of this paper is the online active supervisor *IntuiSup* that combines implicit and explicit supervision mechanisms, to optimize user interactions and system learning. In the context of gesture command recognition, the only way of knowing the true label of a gesture is to interact with the user (explicitly or implicitly). However, soliciting the user after each command cancel the very interest of gesture commands! The implicit supervision strategy consists of taking advantage of implicit validations of system recognitions by the user: if he continues his action without canceling or undoing the executed command, he implicitly validates the recognition. The explicit supervision consists of asking the user to explicitly validate the recognized command, to obtain data label and be able to train the classifier.

Second contribution is the new algorithm for explicit supervision: the Evolving Sampling by Uncertainty (ESU) algorithm. The ESU algorithm uses the classifier confidence to make the sampling decision (in a similar way than a reject option). The user is solicited to obtain data true label when the confidence of the recognition is low. A low confidence means that the data sample is complex to recognize, and it is very important to be able to learn from complex data. The ESU sampling algorithm makes the sampling evolve dynamically to adapt the supervision to system learning and environment changes. Selecting data samples that have a low recognition confidence allows to reduce recognition errors. Although, a selected gesture may be better than a wrong command, it is not desirable to have a lot of user interactions either. There is a trade-off between the number of errors and the number of user interactions.

Third contribution is the boosting method that enhance the previous sampling algorithm to make B-ESU (Boosted - Evolving Sampling by Uncertainty). We designed a learning boosting method that improves system learning speed, during both initial learning and concept drifts (environment changes). In our evolving context, each selected data sample represents an additional labeled data sample available for the classifier learning. As a consequence, it is important to select enough data to improve the classifier efficiently, at least before the learning process has converged. More precisely, it is interesting to select more data at the beginning of system learning, to quickly improve system performance, and then to reduce the sampling rate. In the same way, when concept drifts appear, the sampling rate is raised to fasten system evolution and adaptation to the changes.

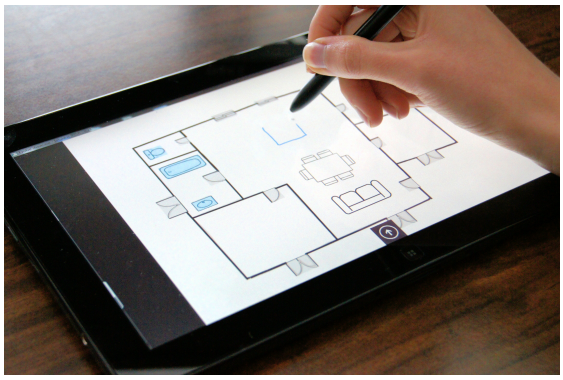

Figure 1: Gesture command used to insert furniture in an architectural plan in the *Varchitect* application (freely available on Windows store).

We evaluate our approach, and our three contributions, on the publicly available ILG Data Base [10] of customized gesture commands. This database is very interesting for two main reasons. First, gestures are chronologically ordered (in their drawing order) which enables to see the evolution of user writing styles with time. Second, for the majority of the database, gesture classes were freely chosen by the writers themselves (see Figure 6). These reasons make this database unique and representative of the real use of an online classifier used for gesture command recognition. The experimentation we conducted shows the effectiveness of our approach for the online active supervision of an evolving classifier of customized gesture commands.

This paper is organized as follows. We present in Section 2 the state-of-the-art in the field of online classification, as well as in the field of active learning. Section 3 describes the architecture of our evolving fuzzy classifier, and its incremental learning algorithm. We detail in Section 4 the new *IntuiSup* supervisor which combines implicit and explicit supervision. Section 5 presents the new *Evolving Sampling by Uncertainty* (ESU) algorithm to make the explicit supervision evolve. Next, Section 6 details the new boosted method B-ESU that we designed to optimize user interactions. Finally, we present the results of our realistic experimentation, and show the improvements induced by our approach with our three contributions, in Section 7. Section 8 concludes and discusses future work.

2. Online Classification State-of-the-Art

In this work, we use an evolving classification system, with an incremental learning algorithm, that smoothly follows all changes in the data stream, but different approaches also exist. This section presents an overview of the state-of-the-art in the fields of online classification, as well as in the field of active learning. It will show the interest of an evolving system for online classification, present the main active learning problems and discuss the different architectures that can be used.

2.1. Online Classification and Evolving Systems

Several approaches can be considered to tackle the problem of changing data stream classification (online classification): re-learning the system, using a sliding window, having an online learning classifier or using an evolving ensemble of weak classifiers [11]. If all approaches are interesting and have different advantages, not all are suitable for customized gesture command recognition.

A first and simple approach is to train a new classifier each time new data samples are available. As a consequence, this approach can be quite effective on a stream where data arrives in chunks, or when few recognition are made [12], but it cannot be considered for fast data streams that require quick and continuous adaptation. Obviously, this approach is very expensive, both in term of computational cost and memory requirement, which makes it unsuitable for gesture commands recognition. Additionally, keeping all data will not be relevant when changes happen in the data stream, and changes will happen as the user get use to the gesture command system.

A method to adapt to concept drifts in the data stream is to use a sliding window that only contains most recent data samples [13][14]. The use of a sliding window also has the advantage of limiting both the computational cost and the memory requirement. When using a sliding window of data, there is a trade-off between the reactiveness and the precision of the system. A short window guarantees a quick adaptation to concept drifts, but will limit recognition performance; and a long window ensures enough learning data to minimize the error rate, but will lengthen system adaptation to changes. Using a sliding window is possible, but relearning the classifier each time a new gesture is drawn is definitely not the most efficient way of recognizing gesture commands.

A second approach is to monitor the data stream to detect any change – the so-called concept drifts – and trigger an adaptation mechanism to modify the classifier accordingly. This method is efficient when changes are occasional, and stable periods quite long between those concept drifts. Some drift detectors are based on changes in the probability distribution of the data samples [15], and others on changes in the classification accuracy [16]. In our case, this method isn't appropriate since we not only want our system to adapt to abrupt concept drifts, but also to smoothly follow slow concept shifts.

We have very few initial learning data, so we need an online learning system – an evolving classifier – that will incrementally learn during its use. Moreover, an evolving system, with an incremental learning algorithm, will smoothly follow any changes in the data stream. Most acknowledged requirements for online evolving systems are:

- each data sample must be processed only once,
- memory and computing time must be limited,
- system learning can be interrupted and its quality shouldn't be altered.

Some simple and well-known classifiers are incremental by nature, like the Bayes classifier and the Nearest Neighbor classifier [17], even if the later require a forgetting mechanism to limit the complexity growth. Some more complex systems are designed to learn online, to adapt their model in a fast and efficient manner to be able to learn incrementally and in real-time. Among those system, we can note the Fuzzy Rule Based Systems [6, 7], Learning Vector Quantization [18], Neural Networks [19] and some Decision Trees [20][21]. Such evolving system perfectly correspond to the requirements induced by the context of gesture command recognition.

Another approach that should be mentioned is the use of an ensemble of weak classifiers, whose predictions are combined to make the final decision [22]. Ensemble of classifiers can be evolving at different levels. First, the combination can evolve, to adapt the weights assigned to the different individual classifiers [23]. Second, the individual classifiers can be evolving systems themselves, and adapt to the data stream [24]. Third, the ensemble itself can evolve, some individuals can be removed from the ensemble, and new ones can be added, trained on most recent data [25]. Even if ensemble classifiers are powerful, they aren't as efficient as evolving system to learn from very few data.

Customized gesture command recognition requires an evolving classifier. Moreover, the supervision of the online learning process requires an active learning strategy.

2.2. Active Learning

The idea of active learning is that the learning system itself will choose which data samples will be used for the learning [26]. A classifier can achieve equivalent performance with only part of the learning data, if those data have been correctly chosen. Active learning is motivated by the fact that obtaining labeled data for learning is costly, which is definitely the case in the context of customized gesture command recognition.

Active learning problems can be divided in three categories:

- query synthesis problems,
- pool sampling problems,
- stream sampling problems.

In the case of query synthesis problems, it is the learner that generates the data sample that will be labeled, from the input space [27]. The generated sample can be the juxtaposition of any values within the feature ranges. However, this way of proceeding isn't appropriated for symbols recognition, as generated samples will often be unrecognizable characters with no semantic meaning.

Pool sampling are the active learning problems where the learner has to choose a subset of samples to be labeled from a pool [28]. In pool sampling problems, the learner can only get the labels of a small number of samples, but it can also use the unlabeled samples with some semi-supervised learning algorithm. The pool sampling scenario applies to batch problems, but not to online problems.

A online classification task implies the last scenario, so-called stream sampling. It means that data samples arrive in a stream and the learner has to decide, on a sample per sample basis, if it wants the current sample to be labeled [29] [30] [31].

This is definitely the problem we face in gesture command learning: to decide after each gesture command if it will be interesting to learn from this gesture, and then ask the user for its true label.

For any of the above mentioned active learning problems, the learner has to make the so-called sampling decision, to decide which are the samples that will be used for learning. The most common way of making this decision, called uncertainty sampling, is to query the label of the sample that the learner is the least certain of how to label [30][32]. However, different methods also exists to make the sampling decision such as querying by committee [33], expected model change [34], expected error reduction [35], variance reduction [36] and density weight [34].

In our case, the classifier makes this sampling decision with the simple but powerful uncertainty sampling method [26]. In order to build an uncertainty sampling method we need a method to evaluate the classifier confidence during the recognition process. It enable to ask the user to label data samples for which the system is likely to make a recognition error, and which will be very interesting for the evolving classifier learning.

This paper focuses on the online active supervision of a recognition system, in the context of customized gesture command. Optimizing user-system interaction in this cross-learning context is actually an active learning problem called stream sampling. We use an evolving fuzzy classifier to cope with this online learning situation. Next section present the evolving classifier we use, which is a fuzzy inference system. Its architecture will be presented, as well as the confidence measure that is used to evaluate recognition confidence and make the active learning sampling decision.

3. Evolving Fuzzy System Overview

This work is based on the evolving classifier Evolve [9, 37], which is a first order Fuzzy Inference System (FIS). It is able to start learning from scratch, add new classes on-the-fly and learn incrementally, which makes it very suitable for customized gesture command learning. However, it is interesting to notice that the *Intuisup* supervisor, combining the supervision strategies, the sampling algorithm and the boosting method, is independent from the classifier. Any other evolving classifier could be used with our approach, provided that a confidence measure is available to make the active learning sampling decision.

This Section presents the evolving classifier on which this work is based to recognize gesture commands: Evolve [9], which is a first order Fuzzy Inference System (FIS). We start by presenting some related works and describing the architecture of a first order FIS. Next, we present the incremental learning algorithm for the online learning of Evolve. Then, we present the confidence measure that is used by the *IntuiSup* supervisor to trigger user interactions.

3.1. Related Works

Evolving fuzzy systems are not new [38], but have really started to be used in the last decade [6, 7] and then intensively studied [39, 40, 41].

DENFIS [42], eTS [6, 43] and FLEXFIS [44] are among the first evolving fuzzy systems. A lot of similar fuzzy systems have been presented since then. Evolve [9] uses generalized fuzzy rules in arbitrarily rotated position for increased precision, which was then re-used on regression problems [45] Another simplified alternative to define the antecedent part of fuzzy evolving systems has been presented, using data Clouds and density distribution [46]. GENEFIS [47] delivers a sensible trade-off between high predictive accuracy and parsimonious rule base while reckoning tractable rule.

Evolving fuzzy system can handle concept drifts [41], mainly with the integration of gradual forgetting factors. PANFIS [48] can split, merge or remove fuzzy rules to improve concept drifts adaptation. pClass [49] can also recall old rules to better follow reoccurring concepts.

Most of the fuzzy models cited above are of type 1 [38]. Recently, type 2 model have been presented [50, 51]. They use non-linear Chebyshev polynomials as rule conclusions to obtain better performances.

Some evolving fuzzy systems also integrate active learning strategies. Most of them are based on rejection principles, like the single-pass active learning approach of Lughofer [52], employing reliability concepts in the form of confusion and distance rejection (so-called conflict and ignorance models); on confusion measure [53]. Active learning modules can also be incorporated into meta-cognitive learning algorithm [54, 51]. The problem of active learning under concept drifts has also been studied by Lughofer et al. [55], who monitors the classifier behavior with a modified version of the Page-Hinkley statistical test.

Our approach, namely the *IntuiSup* supervisor, is also based on rejection principles. However, it extends previous approaches in the sense that it combines implicit and explicit supervision in its active learning strategy. This strategy make it possible to take advantage of the applicative context of gesture command recognition to further reduce user interaction requirements. The *IntuiSup* supervisor is also designed to handle smoothly gradual and abrupt concept drifts, with its boosted learning strategy (See Section 6).

3.2. System Architecture

We focus here on Fuzzy Inference Systems (FIS) [56], with first order conclusion structure – called Takagi-Sugeno FIS [38]. FIS have demonstrated their good performance for incremental classification of changing data flows [7]. Moreover, they can easily be trained online – in real time – and have a good behavior with new classes. In this section, we present the architecture of the evolving FIS *Evolve* [9] on which this work is based.

A Fuzzy Inference System consists of a set of fuzzy inference rules like the following example [40][57].

$$\textbf{Rule}^{(i)} : \textbf{IF } \mathbf{x} \text{ is close to } C^{(i)} \tag{1}$$

$$\textbf{THEN } \hat{\mathbf{y}}^{(i)} = (\hat{\mathbf{y}}_1^{(i)}\,;\,\ldots\,;\,\hat{\mathbf{y}}_c^{(i)})^\top \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the feature vector, $C^{(i)}$ the fuzzy prototype associated to the i-th rule and $\hat{\mathbf{y}}^{(i)\top} \in \mathbb{R}^c$ the output vector of

the i-th rule. Rule premises are the fuzzy membership to rule prototypes, which are clusters in the input space. Rule conclusions are fuzzy membership to all classes, that are combined to produce system output.

3.2.1. Premise Structure

Our model uses rotated hyper-elliptical prototypes that are each defined by a center $\boldsymbol{\mu}^{(i)} \in \mathbb{R}^n$ and a co-variance matrix $\Sigma^{(i)} \in \mathbb{R}^{n \times n}$ (where n is the number of features) [9].

$$\Sigma^{(i)} = \begin{bmatrix} \sigma_1^2 & \dots & c_{1,n} \\ \vdots & \ddots & \vdots \\ c_{n,1} & \dots & \sigma_n^2 \end{bmatrix} \tag{3}$$

By taking into consideration the covariance between features, we allow our fuzzy prototypes to have a rotated hyper-elliptical form.

The activation degree $\alpha^{(i)}(\mathbf{x})$ of each fuzzy prototype is computed as follows [6]:

$$\alpha^{(i)}(\mathbf{x}) = \frac{1}{1 + (\mathbf{x} - \boldsymbol{\mu}^{(i)})^\top (\Sigma^{(i)})^{-1} (\mathbf{x} - \boldsymbol{\mu}^{(i)})^\top)^{1/2}} \tag{4}$$

3.2.2. Conclusion Structure

In a first order FIS, rule conclusions are linear functions of the input [6]:

$$\hat{\mathbf{y}}^{(i)\top} = (l_1^{(i)}(\mathbf{x}) \,;\, \dots \,;\, l_c^{(i)}(\mathbf{x})) \tag{5}$$

$$l_k^{(i)}(\mathbf{x}) = \mathbf{x}^\top \cdot \boldsymbol{\theta}_k^{(i)} = \theta_{0,k}^{(i)} + \theta_{1,k}^{(i)} \cdot x_1 + \dots + \theta_{n,k}^{(i)} \cdot x_n \tag{6}$$

The i-th rule conclusion can be reformulated as:

$$\hat{\mathbf{y}}^{(i)\top} = \mathbf{x}^\top \cdot \Theta^{(i)} \tag{7}$$

with $\Theta^{(i)} \in \mathbb{R}^{n \times c}$ the matrix of the linear functions coefficients of the i-th rule:

$$\Theta^{(i)} = (\boldsymbol{\theta}_1^{(i)} \,;\, \dots \,;\, \boldsymbol{\theta}_c^{(i)}) = \begin{bmatrix} \theta_{1,1}^{(i)} & \dots & \theta_{1,c}^{(i)} \\ \vdots & \ddots & \vdots \\ \theta_{n,1}^{(i)} & \dots & \theta_{n,c}^{(i)} \end{bmatrix} \tag{8}$$

3.2.3. Inference Process

The inference process consists of three steps [6]:

1. Activation degree is computed for every rule and then normalized as follows:

$$\alpha^{(i)}(\mathbf{x}) = \frac{\alpha^{(i)}(\mathbf{x})}{\sum_{k=1}^{r} \alpha^{(k)}(\mathbf{x})} \tag{9}$$

   where r is the number of rules.

2. Rules outputs are computed using Equation 7 and system output is obtained by sum-product inference:

$$\hat{\mathbf{y}} = \sum_{k=1}^{r} \alpha^{(k)}(\mathbf{x}) \cdot \hat{\mathbf{y}}^{(k)} \tag{10}$$

3. Predicted class is the one corresponding to the highest output:

$$\text{class}(\mathbf{x}) = \arg\max_{k=1}^{c} (\hat{y}_k) \tag{11}$$

Figure 2 represents a FIS with first order conclusion structure as a radial basis function (RBF) neural network [58].

3.3. Rule Creation

In Evolve, rule creation is triggered by the incremental clustering algorithm *eClustering* [6].

When a new sample arrives, it will either integrate an existing cluster, of create a new one. New clusters shall only be created for new data samples containing enough new information, meaning that they should be significantly different from existing clusters. To evaluate the importance of new samples regarding the existing clustering, we use their potential values. The potential of a data sample is defined as the inverse of the sum of all its distances to other data samples [59]:

$$P(\mathbf{x}_t) = \frac{1}{1 + \sum_{i=1}^{t-1} \|\mathbf{x}_t - \mathbf{x}_i\|^2} \tag{12}$$

This potential can also be computed recursively [6] :

$$P(\mathbf{x}(t)) = \frac{t-1}{(t-1) \cdot \beta(t) + \gamma(t) - 2 \cdot \zeta(t) + t - 1} \tag{13}$$

Where

$$\beta(t) = \sum_{j=1}^{n} x_j^2(t) \tag{14}$$

$$\gamma(t) = \gamma(t-1) + \beta(t-1), \quad \gamma(1) = 0 \tag{15}$$

$$\zeta(t) = \sum_{j=1}^{n} x_j(t) \cdot \eta_j(t) \tag{16}$$

$$\eta_j(t) = \eta_j(t-1) + x_j(t-1), \quad \eta_j(1) = 0 \tag{17}$$

When the potential of the new sample is greater than the potentials of every existing clusters, then a new cluster is created, based on this sample.

The potentials of existing clusters $C^{(i)}$ are also updated on the arrival of a new sample:

$$P(C^{(i)}) = \frac{(t-1) \cdot P(C^{(i)})}{t - 2 + P(C^{(i)}) + P(C^{(i)}) \cdot \sum_{j=1}^{n} \|\mu_i - x \cdot (t-1)\|_j^2} \tag{18}$$

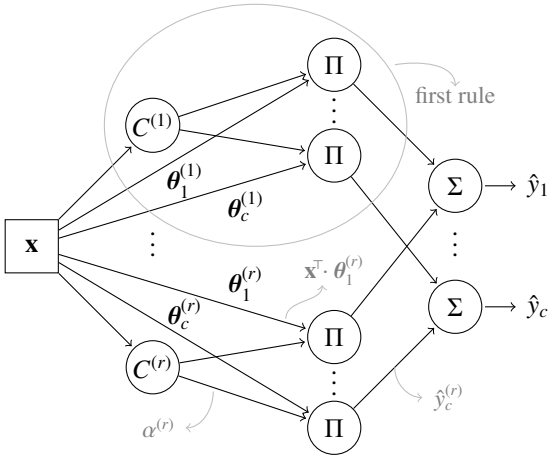

Figure 2: First order FIS as a radial basis function (RBF) neural network

3.4. Incremental Learning Process

Let $\mathbf{x}_i$ ($i = 1..t$) be the i-th data sample, M_i the model at time i, and f the learning algorithm. The incremental learning process can be defined as follows:

$$M_i = f(M_{i-1}, \mathbf{x}_i) \tag{19}$$

whereas a batch learning process would be:

$$M_i = f(\mathbf{x}_1, \ldots, \mathbf{x}_i) \tag{20}$$

In our classifier *Evolve* [9], both rule premises and conclusions are incrementally adapted:

1. Rule prototypes are statistically updated to model the runtime data:

$$\boldsymbol{\mu}_t^{(i)} = \frac{(t-1)\cdot\boldsymbol{\mu}_{t-1}^{(i)} + \mathbf{x}_t}{t} \tag{21}$$

$$\Sigma_t^{(i)} = \frac{(t-1)\cdot\Sigma_{t-1}^{(i)} + (\mathbf{x}_t - \boldsymbol{\mu}_{t-1}^{(i)})(\mathbf{x}_t - \boldsymbol{\mu}_t^{(i)})}{t} \tag{22}$$

2. Rule conclusions parameters are optimized on the data flow, using the Recursive Least Squares (RLS) algorithm:

$$\Theta_t^{(i)} = \Theta_{t-1}^{(i)} + \alpha^{(i)}\mathrm{C}_t^{(i)}\mathbf{x}_t(\mathbf{y}_t^\top - \mathbf{x}_t^\top\Theta_{t-1}^{(i)}) \tag{23}$$

$$\mathrm{C}_t^{(i)} = \mathrm{C}_{t-1}^{(i)} - \frac{\mathrm{C}_{t-1}^{(i)}\mathbf{x}_t\mathbf{x}_t^\top\mathrm{C}_{t-1}^{(i)}}{\frac{1}{\alpha^{(i)}} + \mathbf{x}_t^\top\mathrm{C}_{t-1}^{(i)}\mathbf{x}_t} \tag{24}$$

3.5. Confidence Measure

This section presents the confidence measure that is used by the *IntuiSup* supervisor to make the active learning sampling decision.

For the explicit supervision strategy presented in this article, we use an inner confidence measure evaluating the classifier confusion degree. This measure allows the sampling of data that they are between the classifier models of two classes, and are hence very interesting for the classifier training. Usually, confidence measures, like reject options, are based on system output (membership to all classes). However, we try to evaluate our model quality from a very early stage of the online learning process. As a result, rules conclusions are still rough and unsettled, and not very representative of the system confidence. Instead of using rule conclusions, which are the discriminative part of our system, we base our confidence measure on rule premises, which are the generative part of our classifier. This makes it possible to detect confusion when gestures activate different prototypes at similar levels, which means that those samples are between the different models of our system. We evaluate system confidence using the principles of confusion reject options.

The Mahalanobis distance is used to compute the distance of a data sample $\mathbf{x}$ to the prototypes $C^{(i)}$ (defined by their center $\boldsymbol{\mu}^{(i)}$ and co-variance matrix $\Sigma^{(i)}$).

$$dist(C^{(i)}, \mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu}^{(i)})^\top(\Sigma^{(i)})^{-1}(\mathbf{x} - \boldsymbol{\mu}^{(i)})^\top \tag{25}$$

From this distances, we compute similarity measures in the same way as prototype activation.

$$sim(C^{(i)}, \mathbf{x}) = \frac{1}{1 + dist(C^{(i)}, \mathbf{x})} \tag{26}$$

With this similarity measures, we compute system confidence as:

$$confidence = \frac{sim_{first} - sim_{second}}{sim_{first}} \tag{27}$$

Where sim_{first} and sim_{second} are the first and the second highest similarity values. A data sample is then flagged as confusing, and rejected, when its confidence is below a certain threshold.

The sampling of samples is subject to the error/interaction trade-off (which is quite similar to the error/rejection trade-off). As the threshold increases, the number of selected gestures raises and the number of classification errors reduces. A high threshold will yield many selections, which will reduce system error rate, whereas a low threshold will yield only a few selections, which will reduce user interaction. There is a trade-off between the classifier error rate and the user interaction rate.

To choose a compromise, one must define the cost of an error of classification, and the cost of a user interaction (necessary or not). On the one hand, a selection will make the system ask the user to validate or correct the recognized label. On the other hand, an error of classification will force the user to cancel/undo his command and try to do it again. Our goal is to select data that doesn't fit well into the classifier model to improve its model, but also permit to reduce classification errors. However, we don't want to select too many data samples and solicit the user too often.

We did a survey among the 62 users that participated in our testing campaign [60]. A selection was rated annoying with an average score of 3.96, on a scale from 0 to 10, whereas a recognition error was rated annoying with an average score of 6.71. It seems reasonable to assume that an error is twice more annoying than a selection since an error force the user to cancel/undo the executed command and to retry the intended command, whereas a selection only require a validation or a correction. As a result, we will try, as much as possible, to have twice as much selection as errors, what we can call the E2SR: 1 Error for 2 Selection Rate. As the classifier is evolving, the sampling method will have to evolve as well to maintain this compromise of E2SR.

Following Section presents the *IntuiSup* supervisor, with its combined implicit and explicit supervision strategies (that we developed with the classifier confidence measure). Next section will then present in detaids the Evolving Sampling by Uncertainty (ESU) algorithm that maintains the desired error/interaction compromise over time.

4. The IntuiSup Supervisor

This section presents the *IntuiSup* supervisor and details the online active supervision strategy we developed, combining implicit and explicit supervision. The implicit supervision mechanism takes advantage of user next action to implicitly label the

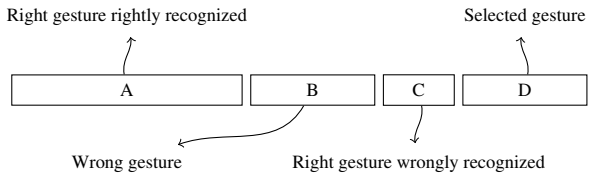

Figure 3: Data partitioning as a result of the user and system cross-learning.

majority of correctly classified data, to strengthen the classifier model. The explicit supervision mechanism makes it possible to learn from complex data samples that are hard to recognize, and from which it is very beneficial to learn. It uses the Evolving Sampling by Uncertainty (ESU) algorithm to trigger user interactions using the classifier confidence measure as input.

In the context of gesture commands, users initialize the system with a few gestures per class (three in our experimentation). To improve gesture command recognition, the classifier learns incrementally during its use. At the same time that the classifier is learning, so is the user: he has to memorize which gesture is associated with which command [4]. In this cross-learning situation, different cases can happen:

Case A The user draws the right gesture which is rightly recognized: the intended command is executed;

Case B The user makes a mistake and draws a wrong gesture;

Case C The classifier makes a mistake and recognizes a wrong command;

Case D The supervisor select the gesture and asks the user to confirm or correct its recognition.

When either the user or the system makes a mistake (case B or C), the command which is executed is not the one that was intended. The user has to cancel/undo that command and try again to do the one he intended. According to this, data can be divided into four categories like shown in Figure 3.

The classifier learning algorithm is a supervised algorithm, it is hence necessary to label run-time data. The supervision strategy is two folded: it combines implicit supervision, without interacting with the user, and explicit supervision, that solicits the user to obtain data true labels. Figure 4 show the online active learning supervision process of the *IntuiDoc* supervisor. The implicit and explicit supervision mechanism are detailed below.

4.1. Implicit supervision: without user interaction

Self supervision consists of labeling run-time data with the labels recognized by the classifier, without interacting with the user. However, this labeling will contain mistakes each time gestures are wrongly recognized. To avoid deteriorating the classifier model by learning on mislabeled data, we take advantage of user next action, to learn only when he implicitly validates the recognized label. The implicit supervision strategy is a form of semi-supervised learning, that allow run-time data labeling without soliciting the user.

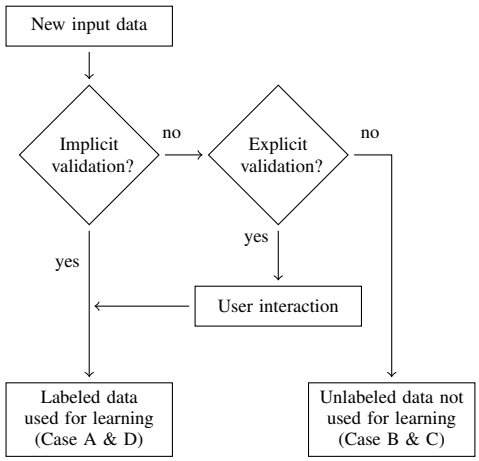

Figure 4: Online active learning supervision process.

In practice, when the user draws a gesture, it is recognized by the classifier and the corresponding command is executed. Two cases are then possible.

- The user cancels or undo this command, either because it doesn't correspond to the gesture he has drawn (classification error, category C of Figure 3), either because he has drawn a wrong gesture (memorization error, category B), or just because he has changed his mind.
- The user continues his actions, which is likely to indicate that the executed command suits his needs, he implicitly validates the recognition (category A).

The implicit supervision is to use the recognized label, but only when it is implicitly validated by the user (by doing another command than cancel/undo). The classifier will learn from the data samples it has correctly recognized (category A), but it will not learn from his mistakes (category C), nor from the user mistakes (category B), rather than risking to learn with a wrong label. This strategy allows being sure not to deteriorate the classifier model, but reduces the number of data samples that can be used for the online learning. The drawback of the implicit supervision is that it only allows learning from correctly recognized data. Learning from incorrectly classified data is much more interesting, but requires user interactions.

4.2. Explicit supervision: labeling by user interactions

Learning from incorrectly recognized data samples requires user interactions to obtain the true labels of the unknown gestures. It is obvious that soliciting the user after each command would be very tedious. We must carefully select the data samples we ask the user to label: it's an active learning problem. To do so, the supervisor uses the ESU algorithm that we developed to select the data samples the system is the least certain of how to classify: it's the uncertainty sampling principle. Those data samples aren't well described by the classifier model, and it will be very beneficial to learn from them. By doing so, the evolving classifier can learn from the gestures that are complex to recognize (category D of Figure 3), and for which it would

have probably made a mistake, thus reducing the error rate at the same time.

Overall, data from categories B and C of Figure 3 isn't used because labels haven't been validated, neither explicitly nor implicitly. Only data from categories A (implicitly validated) and D (explicitly validated) are used by the *IntuiSup* supervisor to learn the classifier. This choice takes out a few data samples that aren't used for learning the classifier, but makes it possible to be sure not to deteriorate its model by learning on potentially mis-labeled data. As a consequence, the supervisor sampling algorithm has a great influence on the classifier learning process. The more data is selected, the more data is available to train the classifier and learning from complex data is very beneficial for the classifier model.

The efficiency of this *IntuiSup* supervisor will be demonstrated by the experimentation presented in Section 7. Following section details the ESU sampling method that is used by the *IntuiSup* supervisor to trigger user interactions.

5. Evolving Sampling by Uncertainty (ESU)

This section present the Evolving Sampling by Uncertainty (ESU) algorithm, which takes the classifier confidence measure as input, and use it to sample data for the classifier learning. The ESU algorithm makes the sampling evolve to maintain the desired error/interaction compromise over time. The context of customized gesture commands implies a cross-learning problem: the user has to memorize the gesture set, and the classifier has to learn the class models. In this cross-learning situation, we try to make user and system cooperate as best as possible, with sampling based user interactions. As a result, we need to make the sampling capacity change and adapt to this dynamic environment in order to maintain the desired error/interaction compromise.

To optimize system learning, and hence system performance, we can tune the sampling capacity to increase data labeling, but we need to keep in mind its impact on user interactions. It is necessary to find a good compromise between the number of recognition errors, and the number of user interactions. The most simple strategy is to choose a constant sampling threshold that optimizes the error/interaction trade-off [61][62] in a similar way than a reject option. The data samples with the highest probability of being mis-recognized are selected, and for an offline system, selected data is homogeneously distributed over time. However, our system is evolving and will improve with time; as a consequence, the number of selected data will decrease over time. In the survey we conducted, users rated errors twice more annoying than selections (sampling and user interaction), which leds us to choose the E2SR compromise: 1 Error for 2 Selections Rate. It is then necessary to make the sampling capacity evolve to maintain this E2SR compromise, as classifier performance will change with time (improve, or worsen in case of concept drift).

The strategy that we follow is to adapt the sampling threshold to maintain the error/interaction compromise as system learns and the environment changes. To make the sampling capacity evolve and follow system evolution, we have developed an online algorithm that updates the sampling threshold as system learns: the Evolving Sampling by Uncertainty (ESU) algorithm.

In this Section, we describe and compare three algorithms:

- Kalman filter based algorithm,
- Automatic Multiple Threshold Learning (AMTL) algorithm,
- the new ESU algorithm.

The two first are state-of-the-art algorithms that we adapted to our online learning situation, but whose performance are limited on our problem (as we will see in Section 7). We developed a third one specifically for this problem: the Evolving Sampling by Uncertainty (ESU) algorithm. The ESU algorithm updates the sampling threshold in a preventive way, which makes it more efficient, whereas the two firsts only update the threshold when some unrecognized data is not selected (mis-acceptation), or when some data are selected and correctly recognized (mis-selection).

5.1. Automatic Multiple Thresholds Learning (AMTL) Algorithm

The Automatic Multiple Thresholds Learning (AMTL) algorithm [63][64] is a generic algorithm to learn multiple rejection thresholds. It is a greedy algorithm based on empiric heuristics. AMTL uses both examples and counter examples to set the threshold according to the desired error/reject compromise.

The AMTL algorithm starts with a high rejection threshold, that rejects all learning samples. Then, it decreases the rejection threshold accepting more and more target examples, and inevitably some counter examples. The algorithm stops when the desired error/reject compromise is achieved on the learning data set. AMTL isn't an online algorithm, it requires a learning set to optimize the threshold. It can be used in an incremental way with the use of a sliding window. However, using a window of past data is costly in term of memory requirements.

The AMTL algorithm can easily be used to optimize the selection threshold, which is similar to a reject option. The error/reject trade-off is then replaced by the error/selection trade-off.

The main interest and advantage of the AMTL algorithm is that it optimizes the threshold to reach the desired error/selection compromise directly. We just have to choose the desired compromise value, the E2SR according to user survey results for instance, and the algorithm does the optimization.

However, the AMTL algorithm only updates the threshold(s) when mis-selections or a mis-acceptations happen. Mis-selections are correctly classified examples that were selected, whereas they would have been used by the implicit supervision strategy. Mis-acceptations are incorrectly classified examples that were accepted, and are hence not used for the classifier learning whereas it would have been beneficial. Threshold adaptation is only corrective, which lacks of reactivity in an online environment.

5.2. Kalman Filter Based Algorithm

This algorithm uses the Kalman filter principle [65] to estimate the separation between correctly and incorrectly recognized samples. The Kalman filter is a linear quadratic estimation algorithm that produces statistically optimal estimates of unknown variables from noisy data. In our case, we feed the Kalman filter with all the mis-selections and mis-acceptations, in order to estimate the separation between examples that should be accepted and (counter) examples that should be selected. The Kalman filter based algorithm starts from an initial separation value and corrects it each time a data sample is either mis-selected or mis-accepted. The selection threshold is then set according to this separation to obtain the desired error/interaction compromise.

The Kalman algorithm is a two-step process. In the prediction step, the variable x_t representing the separation between examples and counter examples, and its uncertainty p_t, are predicted from past values x_{t-1} and p_{t-1}, and where q and r are the process and the measurement noises.

$$x_t = x_{t-1} \tag{28}$$

$$p_t = p_{t-1} + q \tag{29}$$

In the update step, predicted values are corrected with current observation y_t to obtain more accurate predictions.

$$k_t = p_t/(p_t + r) \tag{30}$$

$$x_t = x_t + k_t * (y_t - x_t) \tag{31}$$

$$p_t = (1 - k_t) * p_{t-1} \tag{32}$$

From this separation, we set a ratio θ to obtain the threshold yielding the desired error/interaction trade-off.

$$t_t = \theta \cdot x_t \tag{33}$$

The Kalman filter based algorithm is a very efficient way of tracking the separation between examples and counter examples from the very noisy measurements we have. The drawback of this approach is that the threshold is updated only when the selection process fails, *i.e.* when some example is either mis-selected or mis-accepted. In other words, like for the AMTL algorithm, we only update the threshold in a corrective manner, not a preventive manner.

5.3. Evolving Sampling by Uncertainty (ESU) Algorithm

We designed a new algorithm to adapt the sampling to the classifier evolution in a online way. The Evolving Sampling by Uncertainty (ESU) algorithm adapts the selection threshold not in a corrective manner but on a continuous preventive basis instead. We designed an indicator ζ_t of the system confidence evolution, and we use that indicator to make the sampling threshold evolve accordingly. Estimating system confidence progression is done by following the confidence measure evolution, that we estimate from the mean and standard deviation changes.

The threshold is computed as follows. We estimate the moving average μ_t and moving standard deviation σ_t over the last nb samples:

$$\mu_t = \mu_{t-1} * \frac{nb-1}{nb} + y_t * \frac{1}{nb} \tag{34}$$

$$\sigma_t = \sqrt{\frac{S_t}{t}} \tag{35}$$

$$S_t = S_{t-1}\frac{nb-1}{nb} + (y_t - \mu_t) * (y_t - \mu_{t-1})\frac{1}{nb} \tag{36}$$

Where $nb = \max(t, nb_{max})$ and nb_{max} is the maximum number of samples taken into account. From μ_t and σ_t we compute our system confidence indicator as:

$$\zeta_t = \mu_t - \sigma_t \tag{37}$$

Then, we set a ratio θ to obtain the threshold t_t yielding the desired error/interaction compromise.

$$t_t = \theta \cdot \zeta_t = \theta \cdot (\mu_t - \sigma_t) \tag{38}$$

The major interest of this statistical algorithm is that it is preventive, not corrective. It updates the selection threshold for every data sample, even if it is a correct selection or a correct acceptation. On the contrary, the two previous algorithms (Kalman and AMTL) only update the threshold for the data samples that aren't correctly positioned with respect to the threshold: a false acceptation or a false selection. The effectiveness of the ESU method will be shown in Section 7.

The Evolving Sampling by Uncertainty (ESU) algorithm makes it possible to efficiently update the selection threshold to maintain the desired error/interaction compromise. It makes the active learning sampling capacity, which is used by the *IntuiSup* supervisor, evolve with the classifier and its environment. In addition to this evolving sampling mechanism, we created an boosted method B-ESU that fasten system learning, both at the beginning of system use and in case of concept drift.

6. Boosted - Evolving Sampling by Uncertainty (B-ESU)

This section presents the boosted method B-ESU we designed to optimize user interaction impact and improve system learning speed. The idea is to redistribute learning data, to concentrate more samples at judicious times, instead of having an uniform distribution over time. This method makes it possible to fasten system initial convergence, and adaptation to concept drifts, by increasing the quantity of learning data at judicious times, and then reducing user interactions when the learning process has converged.

6.1. Concept Drift Adaptation with the Learning Boost

The error/interaction trade-off is quite complex in this online learning situation. Selecting data that would have been mis-recognized not only represents avoided mistakes, selected data also represents additional learning data. Additional learning data means an opportunity to improve the system, and an

improved system will make fewer mistakes in the future, hence requiring fewer selections.

The intuition of what we do is that, for the same number of selected data, we will obtain a better system if we concentrate those data at the beginning of the utilization of the system, because it accelerates the learning process. We thus propose a boosting method based on system performance evolution to increase the selection rate during the initial learning phase and when concept drifts appear. This higher selection rate will increase the quantity of learning data that is explicitly labeled by the user, fasten system improvement, and finally allows reducing the interaction rate to an even lower value.

This feature make it possible to follow concept drifts, whether gradual or abrupt, in a smooth way. The knowledge of the system is continuously updated, but the learning boost speed up the adaptation during concept drifts.

The idea of the learning boost concept is to adapt the selection rate to the classifier learning rate. We want to have a high selection rate while the system is learning, to improve its performance faster. Then, when system performance is better, the selection rate can be reduced because fewer errors are made, and the number of user interactions can then be lower.

We choose to link the learning boost, *i.e.* the selection threshold increase, to the error rate evolution. This method permit the automatic adaptation of the learning boost to system evolution. When there are few new classes that are easy to recognize, the error rate decreases quickly and the learning boost rapidly fades. On the contrary, when the new classes are numerous and complex to recognize, the learning boost remains until system learning converges.

In the end, when system learning has converged after the change, the error/interaction compromise that we obtain with this boosting method is more interesting. Fewer errors and interactions will be made during the future use of the classifier.

If another concept drift happens, the error rate will rise and the learning boost will appear again, until the novelty is learned. The learning boost accelerate system learning speed, during both the initial learning phase and environment changes.

6.2. Boosted Method B-ESU

In practice, the learning boost is achieved by a temporary increase of the selection threshold which is done by adding a boosting term.

$$t = t_{dynamic} + t_{boost} \tag{39}$$

The boosting term of the selection threshold is linked to the absolute value of the gradient of system recognition rate:

$$|\frac{\partial\ error\_rate}{\partial\ time}| \tag{40}$$

The gradient of system recognition rate is approximated as the difference between a short term and a medium term running average.

$$\delta = |\mu_k - \mu_{2*k}| \tag{41}$$

Where μ_k is the running average of the error rate on (approximately) the k last samples. Finally, we limit the boosting term in order not to exceed a maximum interaction rate, to avoid bothering users too much.

$$t_{boost} = min(\delta, t_{boost\_max}) \tag{42}$$

In our experiments, we set k to five times the number of classes, and and use a maximum boost giving 25% of user interaction. With the boosted method B-ESU, the classifier is learning faster and adapting to concept drifts quicker.

The following section presents the experimentation we conducted to evaluate our approach.

7. Experimentation

This section presents the experimental validation of the *IntuiSup* supervisor for the online active supervision of an evolving classifier. The objective is to improve system recognition performance as much as possible, but without soliciting the user too often. We validated experimentally the combination of implicit and explicit supervision strategies for learning the evolving classifier. We show the effectiveness of the Evolving Sampling by Uncertainty (ESU) algorithm for adapting the sampling capacity, and maintain the desired error/interaction compromise. We also validated the efficiency of the boosted method B-ESU to accelerate system learning speed, during both initial learning and concept drifts.

7.1. Evaluation Database

We evaluate our approach on the ILG Data Base[1] [10] using the supplied HBF49 [66] feature set.

This database contains handwritten gestures that have been collected in an immersive environment. The users were asked to customize and use gesture commands in a picture browser/editor, and were not aware that their gesture were recorded. ILGDB contains 6629 mono-stroke gestures, belonging to 21 classes, that have been drawn by 38 writers. This database is very interesting and unique for three main reasons.

First, gestures are ordered chronologically in their drawing order. This chronological order allows seeing changes in user writing styles with time, as the writers evolve from novice to expert as shown figure 5. This is a very unique property among handwritten gesture database, which is necessary to get a realistic evaluation of an online system that adapts to the writer evolution.

Second, class frequencies varies, from 5 to 17 samples per class (per writer). The data stream of each writer is divided in 5 phases, and for two of them classes are drawn at different frequencies, some are even not used. This feature makes this database even more realistic and representative of the real use of an online classifier for gesture command recognition.

Third, for the majority of the database, gesture sets were freely chosen by the writers themselves. That makes the gestures various and very diversified, which is representative of

[1] Freely available at `http://www.irisa.fr/intuidoc/ILGDB.html`

Figure 5: Example of the evolution of user writing style with time, as the writer evolve from novice to expert.

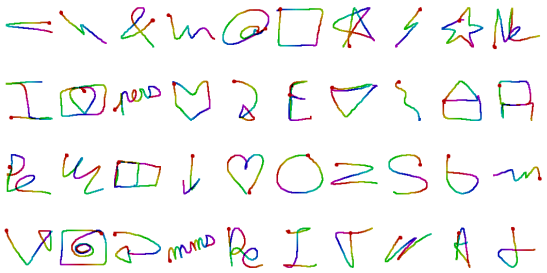

Figure 6: User defined gesture samples from ILGDB.

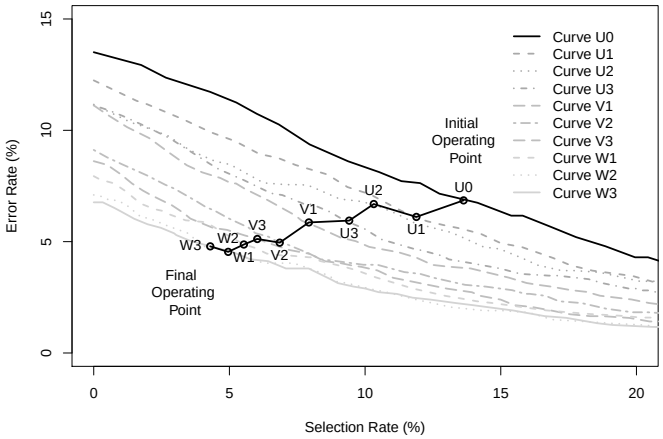

Figure 7: Error/selection curves and operating point evolution over time.

the real use of a customizable system. Some gesture samples invented by ILGDB writers are presented in Figure 6.

These three reasons make this database very realistic and unique. Moreover, it is representative of the real use of a online classifier for gesture command recognition.

The supplied HBF49 [66] feature set is a unified feature representation for universal online symbol recognition. It was designed to cover all the aspects needed for symbol recognition with various characteristics: online, offline, mono-stroke, multi-stroke, and in various contest: handwritten digits, mathematical symbols, iconic gestures, geometrical objects, architectural objects, etc. This versatility makes it very suitable for user defined gesture commands recognition. HBF49 provides a baseline for symbol recognition systems and also serves as a universal benchmarking representation.

7.2. Evaluation Protocol

Drawn symbols are distributed into five phases for each writer. Phase 0 (three symbols per class) is used for system initialization, and phases 1 to 5 ($\sim$ 120 symbols) simulate system use with varying class frequencies. For our experiments, we used 20 random triplets of users (noted U, V and W) that used the same gesture set (group 3). This protocol makes it possible to simulate a longer use of gesture commands with two concept drifts when writer changes (from U to V, and from V to W). We initialize our system on phase 0 of the first writer (U0) and use phases 1 to 3 of the three writers (U1, U2, U3, V1, V2, V3, W1, W2 and W3, $\sim$ 270 symbols) to simulate the online learning of our classifier. We tested our system performance on phase 4 of the three writers ($U4 + V4 + W4 = 63$ symbols) between each of the nine utilization/learning phases.

The error/interaction trade-off is very similar to the error/rejection trade-off. To evaluate the performance of an offline system with a reject option, the most common measure used is the area under the error/reject curve. However, this approach is irrelevant with an online system, where it is not possible to choose any operating point on the error/reject curve. For an online system, the error/reject curve is obtained for the operating point used during the online learning phase, and changing the operating point also changes the curve! Figure 7 shows the evolution of the error/selection curves, and their associated operating points, with time.

In order to evaluate the performance of an online system with a sampling option, we propose a new approach: the error/interaction ratio trajectory. This trajectory allows seeing the changes of the error/interaction compromise, and evaluating if the desired compromise is maintained over time. Additionally, the final operating point enable to evaluate the final state of the system, and compare different configuration in term of error/interaction performance.

7.3. Implicit and Explicit Supervision

Table 1 presents the final raw error rates (including errors among selected samples) obtained on ILGDB with the implicit (cf. section 4.1) and explicit (cf. section 4.2) supervision strategies. As expected, learning without user supervision, neither implicit nor explicit, but using the labels proposed by the classifier itself (self supervision), deteriorates the classifier performance: the raw error rate (including selected samples) raises from 11.53% to 13.28%. It is better not to learn at all, than to learn on potentially mislabeled samples that damage the classifier model. The implicit supervision strategy allows a recognition performance improvement (raw error rate reduction from 11.53% to 10.15%), but not as much as the explicit strategy (raw error rate of 7.27%). It is essential for the classifier to be able to learn from its mistakes. Finally, the *IntuiSup* supervisor, which combined implicit and explicit supervision, makes it possible to improve efficiently the classifier model performance (raw error rate of 6.52%).

Table 2 present the raw error rate (including selected samples) and user interaction rates obtained for the eventual labeling of the remaining data (category B and C of figure 3). The reference strategy, without using data from category B and C, is the above *IntuiSup* supervisor (raw error rate of 6.52%). Using part B and C requires an additional labeling strategy. Automatic labeling with system output doesn't help because those

Table 1: Comparison of the supervision strategies on ILGDB

Supervision Strategy	Error Rate (%)
No online learning	11.53
Self supervision (A+B+C+D)	13.28
Implicit supervision (A)	10.15
Explicit supervision (D)	7.27
IntuiSup supervisor (A+D)	6.52

Table 2: Additional labeling strategy cost and results

IntuiSup (A+D) +	Error Rate (%)	UI Rate (%)
No supervision (B+C)	6.52	7.39
Self supervision (B+C)	6.77	7.64
Explicit supervision (B+C)	5.51	16.4

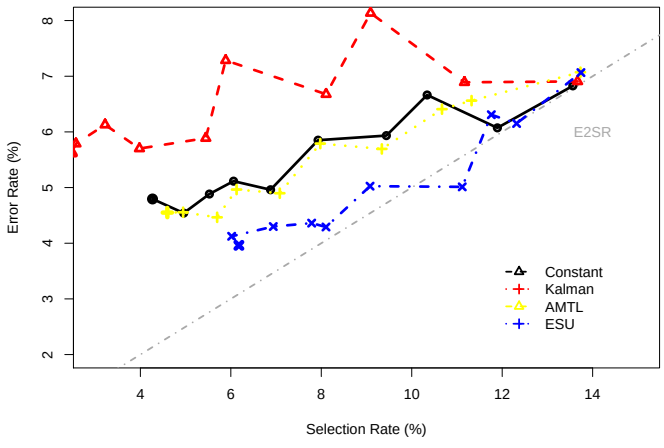

Figure 8: Error/interaction ratio evolution with the different sampling evolution algorithms, with regard to the E2SR compromise (dotted line).

data samples contains recognition errors, the error rate even rise a little (from 6.52% to 6.77%). Explicitly asking the user to label those data samples makes it possible to reduce system error rate from 6.77% to 5.51%, but at the cost of a lot of user interactions. The user interaction rate indeed goes from 7.64% to 16.4% (5.71% of memorization errors, 1.63% of recognition errors, 9.02% of sampling). It means going from 4.80 to 10.32 user interactions on the 63 symbols test set (115% of relative augmentation), making this slight performance improvement very costly, and not worthy from the user point of view. Finally, we don't use data from category B and C in the online learning process of the classifier, which represents 9% of the total data. The *IntuiSup* supervisor uses implicitly validated data (category A of figure 3), and explicitly validated data (category D) to optimize both system performance and user interactions.

7.4. Evolving Sampling by Uncertainty (ESU)

Figure 8 compares the three algorithms for the selection threshold evolution (Kalman, AMTL and ESU), and the constant threshold strategy as reference.

Even if the selection threshold stays constant, the recognition system evolves and improves, which makes the error/interaction compromise change. When system performance improves, so does average system confidence. As a result, system interaction rate decreases faster than the error rate, and the error/interaction compromise changes. Moreover, concept drifts also modify the error/interaction compromise.

The objective is to maintain the desired E2SR compromise of twice as much selections (interactions) as errors, that suits users best (represented by the dotted diagonal on Figure 8). The AMTL algorithm gives similar performance to the constant threshold. The Kalman based algorithm fails to maintain the error/interaction compromise, and yield too much errors and very few interactions. On the contrary, the ESU method is able to follow system improvement, and adapt the selection threshold accordingly. This algorithm globally follows the target diagonal, and gives a compromise of 3.96% of errors for 6.18% of selections (interactions).

It is noteworthy that the Evolving Sampling by Uncertainty (ESU) algorithm could follow any other given error/interaction compromise than the E2SR (1 Error for 2 Selection Rate), like the E3SR (1 Error for 3 Selection Rate) for instance (as shown Figure 9).

7.5. Boosted - Evolving Sampling by Uncertainty (B-ESU)

Figure 9 shows the effects of our boosted method B-ESU, on an evolving threshold. The boosting method increases the interaction rate at the beginning and reduces it as system learns. In the end, the error/interaction ratio obtained when using the boosting method is more interesting. The error rate is 16% lower (from 3.96% to 3.31%) than the one obtained without boosting, for the same interaction rate (6.33%).

For the purpose of comparison, Figure 9 also shows the curve obtained without boosting, but with a higher threshold and a different error/interaction compromise: the E3SR, to get the same final error rate (3.31%). Increasing the threshold (without boosting) to get the same final error rate (3.31%), yield a

Table 3: Average number of user interactions per writer (on the 21 gestures of phase 4)

Test after Phase	0	1	2	3	4	5	6	7	8	9	Total
ESU	4.6	4.1	3.9	3.7	3.0	2.7	**2.6**	**2.3**	2.0	**2.1**	31.0
B-ESU (E2SR)	**6.7**	**5.0**	3.8	3.3	3.3	2.8	**2.5**	**2.6**	2.4	**2.1**	34.5
ESU (E3SR)	**5.7**	**5.3**	4.8	4.3	3.8	3.2	3.3	2.9	2.6	2.7	36.5

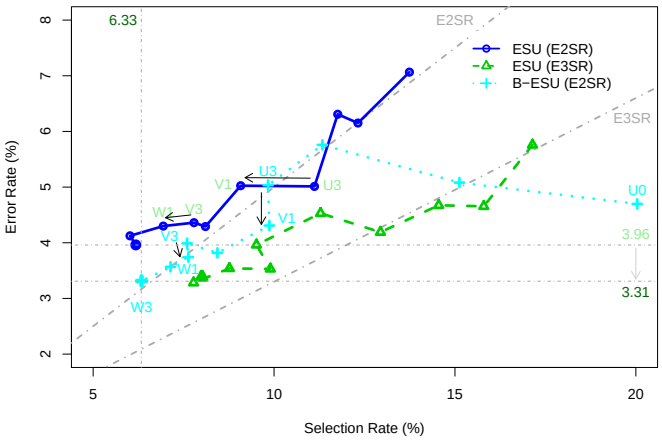

Figure 9: Boosted method B-ESU impact on the error/interaction ratio evolution and final error/interaction compromise.

increase of 27% of the interaction rate: from 6.33% to 8.01%. Overall, the boosted method B-ESU allows a performance improvement, in term of error rate as well as interaction rate.

We can also notice on figure 9 that the interaction rate slightly increases at the operating points at phases $V1$ and $W1$ (shown with black arrows on figure 9) with the boosting method, whereas it sharply decreases without. Those two points actually corresponds to the slight concept drifts due to the changes of writer. When the writer changes, the drawing style also changes, which generates a slight increase of the error rate. As a consequence, the B-ESU method increases the interaction rate to boost system learning and fasten the adaptation to the concept drift.

Table 3 presents the average number of user interactions on the testing data set between each learning phase. The (average) total number of solicitations on the whole experiment goes from 31.0 to 34.5 with the boosted method B-ESU (11.3% of relative augmentation). However, this augmentation is temporary, the number of user interactions is back at the same level at the end of the experiment: 2.1 user interactions in average on the test set. Slightly modifying user interactions during the beginning of system use, and during concept drifts, enable to decrease the error rate by 16% (from 3.96 to 3.31) at the end of the experiment, and thus for the future use of the recognition system. Comparing to a higher threshold (for the E3SR compromise), using a lower threshold with the boosted method B-ESU makes it possible to reduce the number of user interactions by 5.5% (from 34.5 to 36.5) during the experiment; and allows reducing the interaction rate by 20.8% (from 8.01 to 6.33).

8. Conclusion

Learning a classifier for the recognition of gesture commands is an online learning situation that requires a supervision strategy to label run-time data. Most of correctly recognized data can be labeled implicitly with users next action, but it is essential to be able to learn from mis-recognized data. To do so, it is necessary to interact with the user to be able to label complex data and improve the classifier model efficiently. On the other hand, constantly soliciting the user is tedious, and considerably reduces the easiness of use of gesture commands. A compromise must be chosen between the number of user interactions and the number of recognition errors.

We have studied the impact of different supervision strategies and showed the effectiveness of the *IntuiSup* supervisor for the online learning of an evolving classifier of gesture commands. We have seen that some data can be implicitly labeled, to reinforce system knowledge without soliciting the user. As it is fundamental to be able to learn from mis-recognized data samples – with their correct labels – to improve the classifier performance, we interact with the user to actively supervise the classifier online learning. In particular, we use a confidence measure from the classifier to make the sampling decision of our active learning strategy and learn from data samples that don't fit the classifier model.

There is a trade-off between the numbers of errors and interactions, so we conducted an user survey to choose the compromise of 1 Error for 2 Selections Rate (E2SR). As we are facing an online learning situation, the error/interaction compromise will change with system improvement. We presented the Evolving Sampling by Uncertainty (ESU) algorithm, based on a statistic modeling of system evolution, that successfully makes the sampling capacity evolve to maintain the desired error/interaction compromise.

In addition of the evolving sampling capacity, we also presented the boosted method B-ESU to fasten system learning speed. This method increases the interaction rate at the beginning of system use, and during concept drifts, to fasten system learning process. This quicker improvement of the classifier performance then enables to reduce the sampling rate, because the classifier makes less mistakes, and hence to reduce user interactions.

This paper has presented a novel approach for the online active learning of gesture commands – namely the *IntuiSup* supervisor – with three contributions: the combination of implicit and explicit supervision strategies, the Evolving Sampling by Uncertainty (ESU) algorithm and the boosted method B-ESU. Our approach improves the recognition system performance on the experimentation we conducted on the ILG Data Base.

References

[1] J. O. Wobbrock, M. R. Morris, A. D. Wilson, User-defined gestures for surface computing, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '09, ACM, 2009, pp. 1083–1092.

[2] J. Yang, J. Xu, M. Li, D. Zhang, C. Wang, A real-time command system based on hand gesture recognition, in: 2011 Seventh International Conference on Natural Computation (ICNC), Vol. 3, 2011, pp. 1588–1592.

[3] P. Y. Li, N. Renau-Ferrer, E. Anquetil, E. Jamet, Semi-customizable Gestural Commands Approach and Its Evaluation, in: 2012 International Conference on Frontiers in Handwriting Recognition (ICFHR), 2012, pp. 473–478.

[4] P. Li, M. Bouillon, E. Anquetil, G. Richard, User and System Cross-Learning of Gesture Commands on Pen-Based Devices, in: Proceeding of the 14th International Conference on Human-Computer Interaction (INTERACT), Vol. 2, 2013, pp. 337–355.

[5] M. Bouillon, P. Li, E. Anquetil, G. Richard, Using Confusion Reject to Improve (User and) System (Cross) Learning of Gesture Commands, in: Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR), 2013, pp. 1017–1021.

[6] P. Angelov, An approach for fuzzy rule-base adaptation using on-line clustering, International Journal of Approximate Reasoning 35 (3) (2004) 275–289.

[7] P. Angelov, X. Zhou, Evolving Fuzzy-Rule-Based Classifiers From Data Streams, IEEE Transactions on Fuzzy Systems 16 (6) (2008) 1462–1475.

[8] J. O. Wobbrock, A. D. Wilson, Y. Li, Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes, in: Proceedings of the 20th annual ACM symposium on User interface software and technology, UIST '07, ACM, 2007, pp. 159–168.

[9] A. Almaksour, E. Anquetil, Improving premise structure in evolving Takagi-Sugeno neuro-fuzzy classifiers, Evolving Systems 2 (1) (2011) 25–33.

[10] N. Renau-Ferrer, P. Li, A. Delaye, E. Anquetil, The ILGDB database of realistic pen-based gestural commands, in: Proceeding of the 21st International Conference on Pattern Recognition, 2012, pp. 3741–3744.

[11] J. Gama, A survey on learning from data streams: current and future trends, Progress in Artificial Intelligence 1 (1) (2012) 45–55.

[12] C. C. Aggarwal, J. Han, J. Wang, P. S. Yu, On demand classification of data streams, in: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2004, pp. 503–508.

[13] M. Last, Online classification of nonstationary data streams, Intelligent Data Analysis 6 (2) (2002) 129–147.

[14] A. Bifet, R. Gavalda, Learning from Time-Changing Data with Adaptive Windowing., in: SDM, Vol. 7, SIAM, 2007, p. 2007.

[15] M. Markou, S. Singh, Novelty detection: a review—part 1: statistical approaches, Signal processing 83 (12) (2003) 2481–2497.

[16] G. Widmer, M. Kubat, Learning in the presence of concept drift and hidden contexts, Machine Learning 23 (1) (1996) 69–101.

[17] D. W. Aha, D. Kibler, M. K. Albert, Instance-based learning algorithms, Machine learning 6 (1) (1991) 37–66.

[18] T. Kohonen, Learning vector quantization, Springer, 1997.

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

[20] G. Hulten, L. Spencer, P. Domingos, Mining time-changing data streams, in: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2001, pp. 97–106.

[21] S. Hashemi, Y. Yang, Flexible decision tree for data stream classification in the presence of concept change, noise and missing values, Data Mining and Knowledge Discovery 19 (1) (2009) 95–131.

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

[23] R. Elwell, R. Polikar, Incremental learning of concept drift in nonstationary environments, Neural Networks, IEEE Transactions on 22 (10) (2011) 1517–1531.

[24] L. I. Kuncheva, Classifier ensembles for changing environments, in: Proceedings of the 5th International Workshop on Multiple Classifier Systems, Springer, 2004, pp. 1–15.

[25] N. C. Oza, Online bagging and boosting, in: Systems, man and cybernetics, 2005 IEEE international conference on, Vol. 3, IEEE, 2005, pp. 2340–2345.

[26] B. Settles, Active Learning Literature Survey, Computer Sciences Technical Report 1648, University of Wisconsin–Madison (2010).

[27] D. Angluin, Queries and concept learning, Machine learning 2 (4) (1988) 319–342.

[28] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, H.-J. Zhang, Two-Dimensional Active Learning for image classification, in: IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008, 2008, pp. 1–8.

[29] D. Cohn, L. Atlas, R. Ladner, Improving generalization with active learning, Machine learning 15 (2) (1994) 201–221.

[30] D. D. Lewis, W. A. Gale, A sequential algorithm for training text classifiers, in: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval, Springer-Verlag New York, Inc., 1994, pp. 3–12.

[31] A. Fujii, T. Tokunaga, K. Inui, H. Tanaka, Selective sampling for example-based word sense disambiguation, Computational Linguistics 24 (4) (1998) 573–597.

[32] T. Scheffer, C. Decomain, S. Wrobel, Active hidden markov models for information extraction, in: Advances in Intelligent Data Analysis, Springer, 2001, pp. 309–318.

[33] H. S. Seung, M. Opper, H. Sompolinsky, Query by committee, in: Proceedings of the fifth annual workshop on Computational learning theory, ACM, 1992, pp. 287–294.

[34] B. Settles, M. Craven, An analysis of active learning strategies for sequence labeling tasks, in: Proceedings of the conference on empirical methods in natural language processing, Association for Computational Linguistics, 2008, pp. 1070–1079.

[35] N. Roy, A. McCallum, Toward optimal active learning through monte carlo estimation of error reduction, ICML, Williamstown (2001) 441–448.

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

[37] A. Almaksour, E. Anquetil, ILClass: Error-driven antecedent learning for evolving Takagi-Sugeno classification systems, Applied Soft Computing 19 (2014) 419–429.

[38] T. Takagi, M. Sugeno, Fuzzy Identification of Systems and Its Applications to Modeling and Control, Systems, Man, and Cybernetics, IEEE Transactions on 15 (1) (1985) 116–132.

[39] P. Angelov, Evolving Takagi-Sugeno fuzzy systems from data streams (eTS+)., in: P. Angelov, D. Filev, N. Kasabov (Eds.), Evolving intelligent systems : methodology and applications, IEEE Press series in Computational Intelligence, John Wiley and Sons and IEEE Press, 2010, pp. 21–50.

[40] P. Angelov, R. Yager, A simple fuzzy rule-based system through vector membership and kernel-based granulation, in: 2010 5th IEEE International Conference Intelligent Systems, IEEE, 2010, pp. 349–354.

[41] E. Lughofer, P. Angelov, Handling drifts and shifts in on-line data streams with evolving fuzzy systems, Applied Soft Computing 11 (2) (2011) 2057–2068.

[42] N. K. Kasabov, Q. Song, DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction, IEEE Transactions on fuzzy systems 10 (2) (2002) 144–154.

[43] P. Angelov, D. Filev, Simpl_ets: a simplified method for learning evolving Takagi-Sugeno fuzzy models, in: The 14th IEEE International Conference on Fuzzy Systems, 2005. FUZZ'05., IEEE, 2005, pp. 1068–1073.

[44] E. D. Lughofer, FLEXFIS: A Robust Incremental Learning Approach for Evolving Takagi-Sugeno Fuzzy Models, Fuzzy Systems, IEEE Transactions on 16 (6) (2008) 1393–1410.

[45] E. Lughofer, C. Cernuda, S. Kindermann, M. Pratama, Generalized smart evolving fuzzy systems, Evolving Systems 6 (4) (2015) 269–292.

[46] P. Angelov, R. Yager, A new type of simplified fuzzy rule-based system, International Journal of General Systems 41 (2) (2012) 163–185.

[47] M. Pratama, S. G. Anavatti, E. Lughofer, GENEFIS: toward an effective localist network, IEEE Transactions on Fuzzy Systems 22 (3) (2014) 547–562.

[48] M. Pratama, S. G. Anavatti, P. P. Angelov, E. Lughofer, PANFIS: a novel incremental learning machine, IEEE Transactions on Neural Networks and Learning Systems 25 (1) (2014) 55–68.

[49] M. Pratama, S. G. Anavatti, M. Joo, E. D. Lughofer, pClass: an effective classifier for streaming examples, IEEE Transactions on Fuzzy Systems

23 (2) (2015) 369–386.
[50] M. Pratama, J. Lu, G. Zhang, Evolving type-2 fuzzy classifier, IEEE Transactions on Fuzzy Systems 24 (3) (2016) 574–589.
[51] M. Pratama, J. Lu, E. Lughofer, G. Zhang, S. Anavatti, Scaffolding type-2 classifier for incremental learning under concept drifts, Neurocomputing 191 (2016) 304–329.
[52] E. Lughofer, Single-pass active learning with conflict and ignorance, Evolving Systems 3 (4) (2012) 251–271.
[53] M. Pratama, S. G. Anavatti, J. Lu, Recurrent Classifier Based on an Incremental Metacognitive-Based Scaffolding Algorithm, IEEE Transactions on Fuzzy Systems 23 (6) (2015) 2048–2066.
[54] M. Pratama, J. Lu, S. Anavatti, E. Lughofer, C.-P. Lim, An incremental meta-cognitive-based scaffolding fuzzy neural network, Neurocomputing 171 (2016) 89–105.
[55] E. Lughofer, E. Weigl, W. Heidl, C. Eitzinger, T. Radauer, Recognizing input space and target concept drifts in data streams with scarcely labeled and unlabelled instances, Information Sciences 355 (2016) 127–151.
[56] E. Lughofer, Evolving fuzzy models: incremental learning, interpretability, and stability issues, applications, VDM Verlag Dr. Müller, 2008.
[57] P. Angelov, Autonomous learning systems: from data streams to knowledge in real-time, John Wiley & Sons, 2012.
[58] J.-S. Jang, C.-T. Sun, Functional equivalence between radial basis function networks and fuzzy inference systems, Neural Networks, IEEE Transactions on 4 (1) (1993) 156–159.
[59] R. R. Yager, D. P. Fileu, Learning of fuzzy rules by mountain clustering, in: Optical Tools for Manufacturing and Advanced Automation, International Society for Optics and Photonics, 1993, pp. 246–254.
[60] M. Bouillon, E. Anquetil, P. Li, G. Richard, User Interaction Optimization for an Evolving Classifier of Handwritten Gesture Commands, in: Proceedings of the 14th International Conference on Frontiers in Handwriting Recognition (ICFHR), 2014, pp. 720–725.
[61] N. Gorski, Optimizing error-reject trade off in recognition systems, in: Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on, Vol. 2, IEEE, 1997, pp. 1092–1096.
[62] L. K. Hansen, C. Liisberg, P. Salamon, The error-reject tradeoff, Open Systems & Information Dynamics 4 (2) (1997) 159–184.
[63] H. Mouchère, E. Anquetil, A Unified Strategy to Deal with Different Natures of Reject, in: 18th International Conference on Pattern Recognition (ICPR), Vol. 2, 2006, pp. 792 –795.
[64] H. Mouchère, E. Anquetil, Generalization capacity of handwritten outlier symbols rejection with neural network, in: Tenth International Workshop on Frontiers in Handwriting Recognition, Suvisoft, 2006.
[65] R. J. Meinhold, N. D. Singpurwalla, Understanding the Kalman filter, The American Statistician 37 (2) (1983) 123–127.
[66] A. Delaye, E. Anquetil, HBF49 feature set: A first unified baseline for online symbol recognition, Pattern Recognition 46 (1) (2013) 117–130.