Supervision Strategies for the Online Learning of an Evolving Classifier for Gesture Commands

Manuel Bouillon, Eric Anquetil

Supervision Strategies for the Online Learning of an Evolving Classifier for Gesture Commands

Manuel BOUILLON, Eric ANQUETIL
Université Européenne de Bretagne, France
INSA de Rennes, Avenue des Buttes de Coesmes, F-35043 Rennes
IRISA, CNRS UMR 6074, Campus de Beaulieu, F-35042 Rennes
{manuel.bouillon, eric.anquetil}@irisa.fr

***Abstract*—Touch sensitive interfaces enable new interaction methods like using gesture commands. To easily memorize more than a dozen of gesture commands, it is important to be able to customize them. The classifier used to recognize drawn symbols must hence be customisable, able to learn from very few data, and evolving, able to learn and improve during its use. This work studies different supervision strategies for the online training of the evolving classifier. We compare six supervision strategies, depending on user interaction (solicitation by the system), and self-evaluation capacities (notion of reject). In particular, there is a trade-off between the number of user interactions, to supervise the online training, and the error rate of the classifier. We show in this paper that the strategy giving the best results is to learn from data validated by the user, when the confidence of the recognition is too low, and from data implicitly validated.**

I. 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 that those new interfaces offer. Among them, a new concept has recently appeared: to associate commands to gestures. Those gesture commands [1][2] enable users to execute various actions simply by drawing symbols. Previous studies [3][4] have shown that enabling customization is essential to help user memorization of gestures. To use such gesture commands, a handwritten gesture recognition system is required. Moreover, if gestures are personalized, the classifier has to be flexible and able to learn with few data samples.

Gesture commands give rise to a cross-learning situation where the user has to learn and memorize the gestures 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. Moreover, we can't expect users to draw much more than a few gesture samples per class, so the recognition engine must be able to learn with very few data. Some template matching classifiers exist, like the $1 classifier [5] for instance, that don't require much training. However, such simple systems have limited performances, and don't evolve with the user writing style. For example, 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 recognizer requires an online system, a system that learns on the run-time data flow.

Evolving classification systems have appeared in the last decade to meet the need for recognizers that work in changing environments. They use online learning to adapt to the data flow and cope with class adding (or removal) at run-time. This work uses such an evolving recognizer – namely *Evolve* [6][7] – which is a first order fuzzy inference system. It can start learning from few data and then learns incrementally in real time from the run-time data flow, to adapt its model and to improve its performances during its use.

The online learning algorithm is a supervised algorithm that requires labeled data. In the context of gesture command recognition, the only way of knowing the true label of a gesture is to interact with the user. However, soliciting the user after each command cancel the very interest of gesture commands! The method we use consist first, to take advantage of implicit validations of recognitions by the user: if he continues his action without canceling or undoing the executed command, he implicitly validates the recognition. Secondly, we use the classifier self-evaluation capacity to solicit the user and obtain data true label when the confidence of the recognition is low.

The confidence measure of the classifier recognition can be of two kinds: an absolute measure for distance rejection or a relative measure for confusion reject. For the strategies presented in this article, we use an inner confidence measure evaluating the classifier confusion degree. This measure allows to reject data when they are between the classifier models of two classes. We are hence using confusion reject.

This paper is organized as follows. The Section II presents the architecture of our evolving classifier, its incremental learning algorithm and the rejection capacity we introduced to develop our supervision strategies. We detail in Section III how our different strategies for the online training work, and their impact on user interactions. Then, we compare the different supervision strategies in a realistic experimentation in Section IV. Section V concludes and discusses future work.

II. EVOLVING FUZZY INFERENCE SYSTEM

This Section presents the evolving Fuzzy Inference System (FIS) on which this work is based [7]. We quickly describe the architecture of a first order FIS. Next, we present the incremental learning algorithm we use for the online training

of our classifier. Then, we details the confidence measure and rejection capacity we introduced to develop supervision strategies using user interactions.

A. System Architecture

We focus here on Fuzzy Inference Systems (FIS) [8], with first order conclusion structure – so-called Takagi-Sugeno FIS [9]. FIS have demonstrated their good performances for incremental classification of changing data flows [10]. 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* [6] that we use to recognize our gesture commands.

Fuzzy Inference Systems consist of a set of fuzzy inference rules like the following rule example.

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

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

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. 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 the system output.

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).

The activation degree $\alpha^{(i)}(\mathbf{x})$ of each fuzzy prototype is computed using the multivariate normal distribution.

2) Conclusion Structure: In a first order FIS, rule conclusions are linear functions of the input:

$$\hat{\mathbf{y}}^{(i)\top} = (l_1^{(i)}(\mathbf{x})\,;\,\ldots\,;\,l_c^{(i)}(\mathbf{x})) \tag{3}$$

$$l_k^{(i)}(\mathbf{x}) = \mathbf{x}^\top \cdot \boldsymbol{\theta}_k^{(i)} = \theta_{0,k}^{(i)} + \theta_{1,k}^{(i)} \cdot x_1 + \cdots + \theta_{n,k}^{(i)} \cdot x_n \tag{4}$$

The *i*-th rule conclusion can be reformulated as:

$$\hat{\mathbf{y}}^{(i)\top} = \mathbf{x}^\top \cdot \Theta^{(i)} \tag{5}$$

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

$$\Theta^{(i)} = (\boldsymbol{\theta}_1^{(i)}\,;\,\ldots\,;\,\boldsymbol{\theta}_c^{(i)}) \tag{6}$$

3) Inference Process: The inference process consists of three steps:

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

$$\alpha^{(i)}(\mathbf{x}) = \frac{\alpha^{(i)}(\mathbf{x})}{\sum_{k=1}^{r} \alpha^{(k)}(\mathbf{x})} \tag{7}$$

where r is the number of rules.

2) Rules outputs are computed using Equation 5 and system output is obtained by sum-product inference:

$$\hat{\mathbf{y}} = \sum_{k=1}^{r} \alpha^{(k)}(\mathbf{x}) \cdot \hat{\mathbf{y}}^{(k)} \tag{8}$$

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

$$\text{class}(\mathbf{x}) = \arg\max_{k=1}^{c}(\hat{y}_k) \tag{9}$$

Figure 1 represents a FIS with first order conclusion structure as a radial basis function (RBF) neural network.

B. 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:

$$M_i = f(M_{i-1}, \mathbf{x}_i) \tag{10}$$

whereas a batch learning process would be:

$$M_i = f(\mathbf{x}_1, \ldots, \mathbf{x}_i) \tag{11}$$

In our recognizer *Evolve* [6], both rule premises and conclusions are incrementally adapted:

1) Rule prototypes are statistically updated to model the run-time data:

$$\boldsymbol{\mu}_t^{(i)} = \frac{(t-1)\cdot\boldsymbol{\mu}_{t-1}^{(i)} + \mathbf{x}_t}{t} \tag{12}$$

$$\Sigma_t^{(i)} = \frac{(t-1)\cdot\Sigma_{t-1}^{(i)} + (\mathbf{x}_t - \boldsymbol{\mu}_{t-1}^{(i)})(\mathbf{x}_t - \boldsymbol{\mu}_t^{(i)})}{t} \tag{13}$$

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

$$\Theta_t^{(i)} = \Theta_{t-1}^{(i)} + \alpha^{(i)}\mathrm{C}_t^{(i)}\mathbf{x}_t(\mathbf{y}_t^\top - \mathbf{x}_t^T\Theta_{t-1}^{(i)}) \tag{14}$$

$$\mathrm{C}_t^{(i)} = \mathrm{C}_{t-1}^{(i)} - \frac{\mathrm{C}_{t-1}^{(i)}\mathbf{x}_t\mathbf{x}_t^\top\mathrm{C}_{t-1}^{(i)}}{\frac{1}{\alpha^{(i)}} + \mathbf{x}_t^\top\mathrm{C}_{t-1}^{(i)}\mathbf{x}_t} \tag{15}$$

New rules, with their associated prototypes and conclusions, are created by the incremental clustering method *eClustering* [11] when needed.

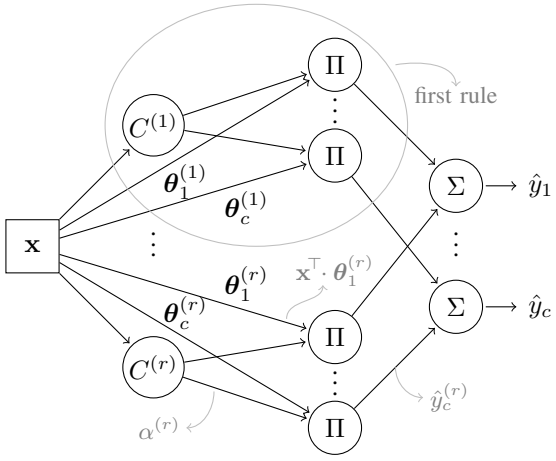

Fig. 1. First order FIS as a radial basis function (RBF) neural network

C. Confidence Measure and Rejection Threshold

We use confusion reject principles to evaluate the system confidence of recognized labels. Usually, confusion reject is based on system output (membership to all classes). However, we try to detect confusion, to evaluate our model quality, at a very early stage of the online learning process. As a result, inference rules conclusions are still rough and unstable, and not very representative of the system confidence. Instead, we choose to use rules premises which are much more stable at this early stage of the online training. Even though every prototype participates in the recognition of every class, each prototype has been created by and is mainly associated with a single class. We use that fact to detect confusion when some gesture activates different prototypes at similar levels.

We use the Mahalanobis distance 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 $\Sigma^{(i)}$).

$$distance(C^{(i)}, \mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu}^{(i)})^{\top} (\Sigma^{(i)})^{-1} (\mathbf{x} - \boldsymbol{\mu}^{(i)})^{\top} \quad (16)$$

From this distances, we compute similarity measures that are smoother than prototype activations.

$$similarity(C^{(i)}, \mathbf{x}) = \frac{1}{1 + distance(C^{(i)}, \mathbf{x})} \quad (17)$$

With this similarity measures, we compute system confidence as:

$$confidence = \frac{s_{first} - s_{second}}{s_{first}} \quad (18)$$

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

The optimization of the rejection threshold is a multi-objective problem: we want to maximize both classifier performance and accuracy.

$$Performance = N_{Correct}/N_{Total} \quad (19)$$

$$Accuracy = N_{Correct}/(N_{Correct} + N_{Errors}) \quad (20)$$

Where $N_{Correct}$ is the number of correctly classified gestures, N_{Errors} is the number of incorrectly classified gestures, and N_{Total} is the total number of gestures. As the threshold increases, the number of rejected gestures raises and the number of classification errors reduces.A high threshold will yield many rejections, which will increase system accuracy, whereas a low threshold will yield only a few rejections, which will increase system performance. There is a trade-off between the classifier performance and accuracy.

To solve this trade-off, we must define the cost of an error of classification, and the cost of a rejection. One the one hand, a rejection 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 do it again. Our goal is to reject data that don't fit well into the classifier model, to reduce classification errors but also to improve its model. However, we don't want to reject too many data and solicit the user too often.

III. Supervision Strategies

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 label;
- Case D — The classifier rejects the gesture and asks the user to confirm or correct its recognition.

When either the user or the system makes a mistake, 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 wants. Data can be divided into four categories like shown in Figure 2.

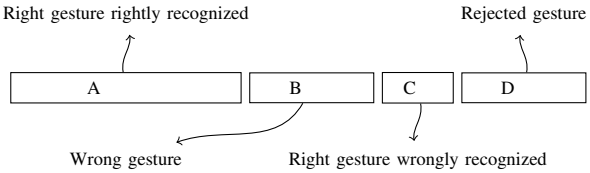

Fig. 2. Data partitioning as a result of the user and system cross-learning

The online learning algorithm used to train the classifier during its use is a supervised algorithm. It is hence necessary to label run-time data. Different strategies can be used: implicit strategies, without interacting with the user, and explicit strategies, that solicit the user to obtain data true label.

A. Supervision strategies based on implicit labeling, without user interaction

The advantage of implicit strategies is that they don't disturb users during their use of gesture commands. In this Section, using reject isn't compulsory and the D category of Figure 2 can be empty.

1) Strategy S1: using recognized label: A first and naive supervision strategy consists of labeling run-time data is to use the label recognized by the classifier, without soliciting the user. However, this labeling will contain mistakes each time gestures are wrongly recognized. In those cases (category C of Figure 2), the classifier will learn with this wrong label, which will reinforce his mistake and deteriorate his model.

2) Strategy S2: using recognized label when implicitly validated by the user (no learning otherwise): 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 undoes this command, either because it doesn't correspond to the gesture he has drawn (classification error), either because he has drawn a wrong gesture (memorization error), or just because he changed his mind.
- The user continues his actions, which is likely to indicate that the executed command suit his needs, he implicitly validates the recognition.

A second learning strategy is again to use the recognized label, but to learn only when the user implicitly validates it (by doing another command that cancel/undo). The classifier will learn from the data samples it has correctly recognized (category A of Figure 2), 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 to be sure not to deteriorate the classifier model, but reduces the number of data that can be used for the online training. Furthermore, the classifier only learns from the data it can correctly recognize, learning from them is interesting but not as much as learning from data that are incorrectly classified.

B. Supervision strategies based on explicit labeling by user interactions

Learning from incorrectly recognized data requires to interact with the user to obtain the true label of the gesture he has drawn. It seems obvious that soliciting the user after each command would be very tedious for the user. We must carefully select the data samples we ask him to label. To do so, we use the classifier confidence measure to select the data samples that aren't well described by the classifier model, and from which it will be very beneficial to learn.

1) Strategy S3: supervision based on explicit labeling by user interaction in case of rejection (no learning otherwise): A third strategy is to use the rejection capacity we introduced in our classifier *Evolve* to solicit the user from times to times, when the recognized label doesn't have a sufficient degree of confidence. By doing so, the classifier can learn from the gestures that are complex to recognize (category D of Figure 2), for which it would have probably made a mistake. On the other hand, as the number of rejections has to be kept as low as possible to minimize user solicitation, few data are available for the online training.

2) Strategy S4: supervision based on explicit labeling in case of rejection, and on the recognized label otherwise: A fourth supervision strategy is to associate the previous strategy, of labeling by user interaction in case of rejection, with the implicit strategy based on the recognized label when no rejection (strategy S2). All the data (category A, B, C and D) are here used for the online training, but here again some data may be mislabeled (data sample not rejected but wrongly recognized).

3) Strategy S5: supervision based on explicit labeling in case of rejection, and on the recognized label if implicitly validated (no learning otherwise): The fifth strategy is similar to the fourth, but without using data from categories B and C of Figure 2 because their label hasn't been validated neither explicitly nor implicitly. Only data from categories A (implicitly validated) and D (explicitly validated) are hence used for the online training. This choice takes out a few data that aren't used for learning, but allow to be sure not to deteriorate the classifier model by learning on potentially mislabeled data.

TABLE I. SUMMARY OF THE CATEGORIES OF DATA USED BY THE DIFFERENT SUPERVISION STRATEGIES (UI: USER INTERACTION)

	Category A	Category B	Category C	Category D
Strategy S1	**yes**	**yes**	**yes**	**yes** (*without* UI)
Strategy S2	**yes**	no	no	no
Strategy S3	no	no	no	**yes** (with UI)
Strategy S4	**yes**	**yes**	**yes**	**yes** (with UI)
Strategy S5	**yes**	no	no	**yes** (with UI)
Strategy S6	**yes**	**yes** (with UI)	**yes** (with UI)	**yes** (with UI)

4) Strategy S6: supervision based on recognized label if implicitly validated and on user solicitation otherwise: Finally, a last strategy can be to use recognized label if implicitly validated, and to solicit the user to get the correct label otherwise. However, user memorization rates vary a lot between users (from 60% to 100%) even if good in average for customized gestures (ILGDB group 1): 94.29% (but 80.02% for all ILGDB) [4]. Users would be solicited very often: when they make a mistake (category B: 5.71%), when the recognizer makes a mistake (category C: 2.13%), and when data are rejected (category D: 10.53%). Moreover, users would also be solicited when they cancel/undo a command for another reason than a recognition or memorization error.

Table I synthesizes which categories of data are used by each supervision strategy.

IV. EXPERIMENTATION

Our objective is to improve our classifier recognition performances as much as possible, but without soliciting the user too often. We compared experimentally the six supervision strategies presented above, and a reference strategy without any online training, summarized below.

- Strategy S0 No online learning (reference strategy);
- Strategy S1 Learning using the recognized label;
- Strategy S2 Learning using the recognized label if implicitly validated (no learning otherwise);
- Strategy S3 Learning using the user label if the gesture is rejected (no learning otherwise);
- Strategy S4 Learning using the user label if the gesture is rejected, and with the recognized label otherwise;
- Strategy S5 Learning using the user label if the gesture is rejected, and with the recognized label if implicitly validated (no learning otherwise);
- Strategy S6 Learning using the recognized label if implicitly validated, and using the user label otherwise (perfect labeling).

A. Evaluation Protocol

We evaluated the different supervision strategies on the ILG Data Base[1] [12] using the supplied HBF49 [13] feature set.

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

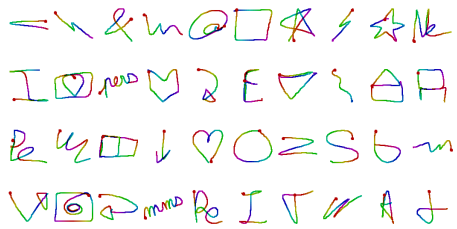

Fig. 3. Gesture samples of ILGDB (group 1: free gestures)

ILGDB contains 6629 mono-stroke gestures, belonging to 21 classes, that have been drawn by 38 writers in an immersive environment. This database is very interesting for three reasons. First, gestures are chronologically ordered (in their drawing order) which enables to see the evolution of user writing styles with time. Second, class frequencies vary, from 5 to 17 samples per class (per writer). Third, for the majority of the database, gesture classes where freely chosen by the writers themselves. These three reasons make this database very realistic and representative of the real use of a handwritten gesture online classifier. Furthermore, the low number of samples per writer (less than 180), and per class (less than 20), makes this database a challenging benchmark for evolving classifiers. Some gesture samples invented by ILGDB writers are presented in Figure 3.

Drawn symbols are distributed into five phases for each writer. Phase 0 contains three symbols per class and is used to initialize the classifier. We have used phases 1 to 3 (~ 90 symbols) for the recognizer online training. Then, we tested our classifier performances on phase 4 (21 symbols, one per class).

B. Rejection Threshold

The rejection threshold is quite difficult to estimate in our applicative context where we are learning from very few data samples. Moreover, each user chooses his own set of symbols/gestures, which may be quite complex to learn by the classifier. The rejection threshold must be automatically estimated for each user and his custom gesture set, and computed in a sufficiently simple and robust manner to yield good results with the few available data.

To estimate the rejection threshold, we chose to initialize our classifier on two of the three initialization samples per class, and to measure the recognition confidence on the third sample. We then compute the mean μ_{reco} and standard deviation σ_{reco} of the correctly recognized data and set the rejection threshold to one standard deviation below the mean: $threshold = \mu_{reco} - \sigma_{reco}$.

A posteriori, we plotted the average error/reject curve in Figure 4 and place the operating point obtained with our threshold estimation method (2.51% of error and 7.39% of reject for the fifth strategy). This operating point is very close to the line of slope 0.5 (drawn in Figure 4), which represent two times more rejections than classification errors. This threshold is very satisfying if we consider that the cost of an error is twice the cost of a rejection, which seems very plausible. If that ratio between the error and rejection costs is different, the threshold can easily be adjusted to move the operating point to one side or another.

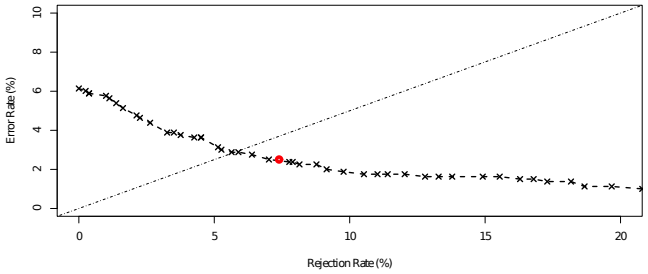

Fig. 4. *A posteriori* error/reject average plot.

C. Supervision Strategies Based on Implicit Labeling, Without User Interaction

Results obtained with implicit strategies are presented in Table II. The first strategy (S1), that consist of learning with the recognized label, deteriorates recognition performances by 1.4% (12% of relative diminution) with respect to the reference strategy (S0) without online learning. Learning with mislabeled data is counterproductive, it is better not to learn at all!

Still, learning on implicitly validated data improves recognition performances, by strengthening the classifier model. The second strategy (S2) reduces the error rates by 1.81% (16% of relative diminution).

D. Supervision Strategies Based on Explicit Labeling by User Interaction

Results obtained with explicit supervision strategies are presented in Table III. The third strategy (S3), which is soliciting the user to learn with the correct label for rejected data, reduces the error rate by 4.23% (37% of relative diminution). Being able to learn from the data that are complex and hard to recognize is very important to improve our classifier model. The fourth strategy (S4), which consist of learning from both rejected data (labeled by user interaction) and accepted data (with the recognized label), only improves very slightly recognition performances (by 0.50% or 7% relatively) with regard to strategy S3.

TABLE II. RESULTS OF IMPLICIT SUPERVISION STRATEGIES, WITHOUT USER INTERACTION

	Error Rate Without Reject (%)	Non-Rejected Error Rate (%)	Rejection Rate (%)
Strategy 0	11.5	4.01	18.0
Strategy 1	12.9	8.27	8.65
Strategy 2	9.69	3.88	10.1

TABLE III. RESULTS OF EXPLICIT SUPERVISION STRATEGIES, WITH USER INTERACTIONS

	Error Rate Without Reject (%)	Non-Rejected Error Rate (%)	Rejection Rate (%)
Strategy 3	7.27	2.38	12.7
Strategy 4	6.77	3.26	7.64
Strategy 5	6.14	2.51	7.39
Strategy 6	5.51	1.63	9.02

TABLE IV. COSTS OF STRATEGIES WITH REGARD TO THE ERROR COST C_e AND THE REJECTION COST C_r.

	error (%)	Solicitation (%)	$C_e = C_r$	$C_e = 2 * C_r$
Strategy 5	2.51	7.39	9.90	12.4
Strategy 6	1.63	16.4	16.4	18.0

The fifth strategy (S5), which associates strategies 2 and 3, allows to get even better performances than when learning from all the data with the correct labels (strategy S6). Learning from data of categories A and D only reduces the error rate by 0.63% (9% of relative diminution) with respect to strategy S4 which uses all data. Here again, it is better not to learn than learning from potentially mislabeled data. Moreover, this strategy only solicits the user for 7.39% of the data, whereas the sixth strategy (S6) requires user interactions for 16.4% of the data: 5.71% for memorization errors (category B), 1.63% for classification errors (category C), and 9.02% for rejection (category D).

Even if the costs of an error and a rejection are difficult to estimate, we can reasonably say that a rejection has a smaller cost than a recognition error. A rejection solicits the user to validate, or correct, the recognized label, whereas an error forces the user to cancel or undo his command and do it again. Table IV compares the cost of the fifth strategy to the one with perfect labeling (user solicitation on data from categories B, C and D from Figure 2) with regard to the error cost C_e and rejection cost C_r.

The fifth strategy, which doesn't use all the data for the online learning, has a lower cost for the user than the sixth strategy, where all data are used (with the correct labels). Finally, learning on data from categories B and C of Figure 2 is not interesting because it requires a lot of user interactions to label data that doesn't improve significantly our classifier model and performances.

V. CONCLUSION

Training a classifier for the recognition of gesture commands is an online learning situation that requires a supervision strategy to label run-time data. Different strategies can be used, some labeling data implicitly and others soliciting explicitly the user to get the correct labels. On the one hand, it is necessary to interact with the user to be able to label complex data and improve our 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 compared six different supervision strategies for the online learning of an evolving classifier. As a result, it is compulsory to correctly label data, at the risk of deteriorating the quality of the classifier model. Actually, it is better not to learn than to learn from potentially mislabeled data. It still remains fundamental to be able to learn from misrecognized data with their correct labels to improve the classifier performances. In particular, we use an inner confidence measure to solicit the user when some data don't fit with the classifier model, and that it will be very gainful to learn from it, but without interacting too often.

This work shows the importance of correctly labeling data in online learning situations, and the difficulty of this labeling task in the context of gesture commands. We also showed the interest of our inner confidence measure to select the data to be labeled, and we highlighted the error/reject (and hence user interaction) trade-off. We optimized our rejection threshold to yield twice as much rejection than classification errors because we believe errors are much more bothering than rejection.

Further optimizing the rejection threshold would require to carry out a complete user experiment to estimate more precisely the error and rejection costs. Moreover, it could be interesting to study the impact of a varying threshold on the error/rejection trade-off and the online learning of the classifier. Interacting with users more often at the beginning of the learning process could fasten the classifier learning, which would then yield less rejections. In the same way, the rejection threshold could be updated online to follow the improvement of the classifier model.

REFERENCES

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

[2] J. Yang, J. Xu, M. Li, D. Zhang, and 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, and 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, and 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 2013*, vol. 2, 2013, pp. 337–355.

[5] J. O. Wobbrock, A. D. Wilson, and 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*, ser. UIST '07, New York, NY, USA, 2007, pp. 159–168.

[6] A. Almaksour and E. Anquetil, "Improving premise structure in evolving takagi-sugeno neuro-fuzzy classifiers," *Evolving Systems*, vol. 2, no. 1, pp. 25–33, 2011.

[7] ——, "ILClass: Error-driven antecedent learning for evolving takagi-sugeno classification systems," *Applied Soft Computing*, 2013.

[8] E. Lughofer, *Evolving fuzzy models: incremental learning, interpretability, and stability issues, applications*. VDM Verlag Dr. Muller, 2008.

[9] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *Systems, Man, and Cybernetics, IEEE Transactions on*, vol. 15, no. 1, pp. 116–132, 1985.

[10] P. Angelov and X. Zhou, "Evolving fuzzy-rule-based classifiers from data streams," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 6, pp. 1462–1475, 2008.

[11] P. Angelov and D. Filev, "An approach to online identification of takagi-sugeno fuzzy models," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 34, no. 1, pp. 484–498, 2004.

[12] N. Renau-Ferrer, P. Li, A. Delaye, and 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.

[13] A. Delaye and E. Anquetil, "HBF49 feature set: A first unified baseline for online symbol recognition," *Pattern Recognition*, vol. 46, no. 1, pp. 117–130, 2013.