Optimisation de la coopération utilisateur/système pour l'apprentissage en-ligne d'un classifieur évolutif

Manuel Bouillon, Eric Anquetil

Optimisation de la coopération utilisateur/système pour l'apprentissage en-ligne d'un classifieur évolutif

Manuel BOUILLON[1] Eric ANQUETIL[1]

[1] Université Européenne de Bretagne
INSA de Rennes, Avenue des Buttes de Coesmes, 35043 Rennes
IRISA, CNRS UMR 6074, Campus de Beaulieu, 35042 Rennes
{manuel.bouillon; eric.anquetil}@irisa.fr

Résumé

Les interfaces homme-machine tactiles permettent de nouveaux modes d'interaction comme l'utilisation de commandes gestuelles. Afin de mémoriser facilement plus d'une douzaine de commandes, il est important de pouvoir les personnaliser. Le classifieur utilisé pour reconnaitre les symboles dessinés doit donc être personnalisable, pouvoir s'initialiser à partir de très peu de données, et évolutif, pouvoir s'améliorer pendant son utilisation. Ces travaux étudient l'importance et les impacts de l'utilisation du rejet pour superviser l'apprentissage en-ligne du classifieur. L'objectif est d'obtenir un système de commandes gestuelles gérant au mieux sa coopération avec l'utilisateur : pour apprendre de ses erreurs sans pour autant le solliciter trop souvent. Il faut donc faire un compromis entre le nombre de sollicitations de l'utilisateur, pour superviser l'apprentissage, et le nombre d'erreurs faites par le système, qui nécessitent une correction de l'utilisateur.

Mots Clef

Commande gestuelles, classifieur évolutif, apprentissage en-ligne, interaction utilisateur, rejet de confusion.

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 customizable, able to learn from very few data, and evolving, able to learn and improve during its use. This work studies the importance and the impact of using reject to supervise the online training of the evolving classifier. The objective is to obtain a gesture command system that cooperates as best as possible with the user : to learn from its mistakes without soliciting him too often. There is a trade-off between the number of user interactions, to supervise the online learning, and the number of classification error, that require a correction from the user.

Keywords

Gesture commands, evolving classifier, online learning, user interaction, confusion reject.

1 Introduction

Avec la démocratisation des écrans tactiles, les interactions Homme-machine évoluent. De nouvelles méthodes d'interaction ont été inventées pour tirer parti du nouveau potentiel d'interaction offert par ces nouvelles interfaces. Parmi elles, un nouveau concept est apparu récemment : associer des commandes à des gestes/symboles. Ces commandes gestuelles [1] [8][10] permettent à l'utilisateur d'exécuter de nombreuses actions simplement en traçant les symboles associés. Pour mémoriser facilement une douzaine de commande, il est essentiel de laisser l'utilisateur choisir lui-même ses propres gestes [6]. L'utilisation de commandes gestuelles requiert un système de reconnaissance de symboles manuscrits. De plus, pour que les commandes gestuelles soient personnalisables, ce reconnaisseur doit être capable d'apprendre à partir de peu de données et d'évoluer pendant son utilisation.

Comme l'utilisateur ne va pas dessiner plus de quelques symboles par classes pour initialiser le système, le moteur de reconnaissance doit être capable d'apprendre à partir de très peu de données. Les classifieurs basé sur des algorithmes d'alignement, comme le *$1 classifier* [9] par exemple, ne nécessitent que peu d'exemples d'apprentissage. Cependant, ces classifieurs basiques ont des performances relativement limitées et n'évoluent pas avec l'utilisateur. En effet, au fur et à mesure qu'un utilisateur se sert du système de commandes gestuelles, il va progressivement passer de l'état de novice à celui d'expert et il va réaliser ses gestes de plus en plus fluidement et rapidement. Dans ce cas, il faut que le classifieur s'adapte à l'utilisateur, et non l'inverse. Le classifieur doit donc être évolutif : il doit être capable d'apprendre en-ligne pendant son utilisation.

Les classifieur évolutifs sont apparus ces dernières an-

1. Voir http://youtu.be/qOx4IY6uYf8 pour une démonstration de commandes gestuelles

nées pour répondre au besoin de classification dans un environnement/contexte non stationnaire. Ils utilisent des algorithmes d'apprentissage incrémental pour apprendre du flux de données et permettent l'ajout de classes en cours d'utilisation. Les travaux présentés dans cet article s'ancrent sur le classifieur évolutif *Evolve* [2] qui est un système d'inférence floue évolutif d'ordre un. Il est capable d'apprendre à partir de très peu de données, et de s'améliorer pendant son utilisation en apprenant incrémentalement en temps réel.

L'apprentissage incrémental mis en œuvre est un apprentissage supervisé qui nécessite des données étiquetées pour apprendre. Dans le cas de la reconnaissance de commandes gestuelles, la seule manière de connaitre l'étiquette d'un geste est d'interagir avec l'utilisateur. Cependant, solliciter l'utilisateur à la suite de chaque reconnaissance réduit fortement l'intérêt et la facilité d'utilisation des commandes gestuelles. Il est donc nécessaire de sélectionner intelligemment les symboles que l'on fait étiqueter par l'utilisateur. La stratégie consiste, d'une part à exploiter la notion de validation implicite de l'utilisateur (si l'utilisateur poursuit ses actions, c'est qu'il valide implicitement son action précédente) ; et d'autre part de profiter de capacité d'auto-évaluation du classifieur pour solliciter l'utilisateur uniquement en cas de confusion potentielle.

L'objectif est de gérer au mieux la coopération du système avec l'utilisateur pour optimiser l'apprentissage sans pour autant solliciter trop souvent l'utilisateur. Les sollicitations utilisateurs seront asservies aux capacités de rejet du classifieur. Autrement dit, du point de vue de la classification, il faut faire un compromis entre le nombre de rejets et le nombre d'erreurs de reconnaissance. Cependant, ce compromis est particulièrement complexe à gérer dans une situation d'apprentissage en-ligne (incrémental en temps réel) où le classifieur évolue en permanence. Le rejet d'une donnée qui aurait été mal reconnue ne représente pas seulement une erreur évitée, mais également une donnée d'apprentissage supplémentaire. Dans cet article, nous cherchons à mettre en évidence qu'il est important de chercher la meilleure stratégie de rejet pour optimiser la coopération. En effet, puisque le rejet d'un geste, engendre une sollicitation utilisateur, cela se traduit par une donnée supplémentaire étiquetée pour l'apprentissage du classifieur incrémental. Aussi il semble important d'accélérer l'apprentissage du classifieur au début de son utilisation pour atteindre ensuite un meilleur point de fonctionnement du système. Nous avons donc travaillé sur la conception d'une stratégie favorisant le rejet au début de l'utilisation du système pour ensuite le réduire quand le système aura atteint une bonne expertise des données qu'il a à traiter. La stratégie proposée ici consiste donc à rejeter davantage au début de l'utilisation du système, pour diminuer son taux d'erreurs rapidement, puis à réduire le taux de rejet pour minimiser le nombre d'interactions utilisateurs.

Cet article est organisé comme suit. La section 2 présente le classifieur évolutif utilisé, ainsi que son apprentissage incrémental et les notions de rejet que nous avons introduites pour développer la stratégie de supervision. Nous présentons en section 3 les différentes stratégies de supervision pour l'apprentissage en-ligne du classifieur, et leurs impacts sur l'interaction utilisateur. Ensuite, nous comparons expérimentalement ces différentes stratégies de supervision en section 4. La section 5 conclue cet article et discute des travaux futurs.

2 SIF d'ordre un

Cette section présente l'architecture de notre système d'inférence floue (SIF) *Evolve*. Nous évoquons ensuite l'algorithme d'apprentissage incrémental utilisé pour entrainer notre classifieur pendant son utilisation. Enfin, nous détaillons les notions de rejet que nous avons introduites pour développer les différentes stratégies de supervision avec interaction utilisateur.

2.1 Architecture du système

Nous travaillons ici avec un système d'inférence floue (SIF) d'ordre un – dit de Takagi-Sugeno – qui offre de bonnes performances face à des problèmes de classification complexes. De plus, il supporte bien l'ajout de nouvelles classes « à la volée » et peut facilement être entrainé de manière incrémentale en temps réel. Des classifieurs évolutifs basés sur des SIF ont été proposés par [3] et par [1].
Un système d'inférence floue d'ordre un se compose d'un ensemble de règles de la forme suivante :

$$\begin{aligned}\textbf{Règle}^{(i)} : \textbf{SI } \mathbf{x} \text{ appartient à } C^{(i)} \\ \textbf{ALORS } \hat{\mathbf{y}}^{(i)} = (l_1^{(i)}(\mathbf{x}) \,;\, \ldots \,;\, l_c^{(i)}(\mathbf{x}))\end{aligned} \tag{1}$$

où $\mathbf{x} \in \mathbb{R}^n$ est le vecteur des caractéristiques et $\hat{\mathbf{y}}^{(i)} \in \mathbb{R}^c$ le vecteur d'appartenance au c différentes classes.
Les prémisses des règles sont définies par l'appartenance floue à des prototypes $C^{(i)}$ – des regroupements (*clusters*) dans l'espace d'entrée – qui sont caractérisés par un centre $\mu^{(i)}$ et une matrice de covariance $\Sigma^{(i)}$. Le degré d'appartenance floue $\beta^{(i)}(\mathbf{x})$ est calculé par une fonction à base radiale hyper-ellipsoïdale, en fonction de la distance de Mahalanobis $d_{\Sigma^{(i)}}(\mathbf{x}, \mu^{(i)})$ entre $\mathbf{x}$ et $C^{(i)}$:

$$\beta^{(i)}(\mathbf{x}) = \frac{1}{1 + d_{\Sigma^{(i)}}(\mathbf{x}, \mu^{(i)})} \tag{2}$$

Les conclusions des règles donnent l'appartenance floue aux différentes classes, et pour les SIF d'ordre un, ces degrés d'appartenance sont obtenus par des fonctions linéaires des entrées :

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

où $l_k^{(i)}(\mathbf{x})$ représente la fonction conséquence linéaire de la règle i pour la classe k :

$$l_k^{(i)}(\mathbf{x}) = \theta_k^{(i)\top}\mathbf{x} = \theta_{k,1}^{(i)} x_1 + \ldots + \theta_{k,n}^{(i)} x_n \tag{4}$$

L'inférence floue de type somme-produit est ensuite utilisée pour calculer la sortie du système pour chaque classe :

$$\hat{\mathbf{y}}(\mathbf{x}) = \sum_{i=1}^{r} \beta^{(i)}(\mathbf{x}) \cdot \hat{\mathbf{y}}^{(i)}(\mathbf{x}) \tag{5}$$

où r représente le nombre de règles floues. La classe de $\mathbf{x}$ est choisie comme étant l'étiquette correspondant à la composante maximale de la sortie du système $\hat{\mathbf{y}}(\mathbf{x}) = (\hat{y}_1(\mathbf{x})\,;\,...\,;\,\hat{y}_c(\mathbf{x}))$:

$$\text{classe}(\mathbf{x}) = \arg\max\ \hat{y}_k(\mathbf{x}) \quad k = 1, ..., c \tag{6}$$

La figure 1 représente un système d'inférence floue (SIF) du premier ordre sous la forme d'un réseau de neurones à fonctions de bases radiales (RBF).

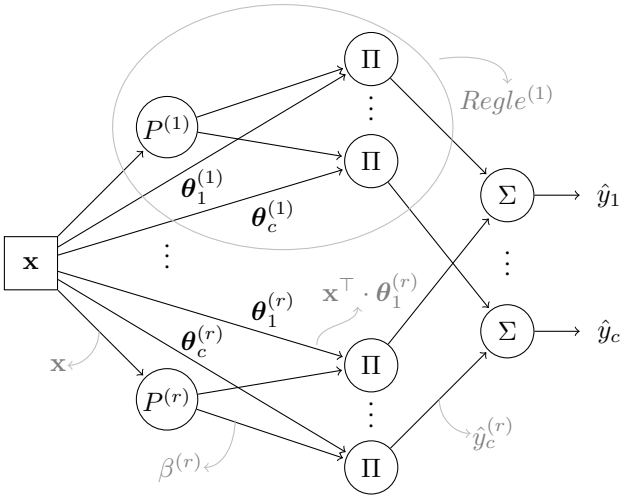

FIGURE 1 – Système d'inférence floue (SIF) du premier ordre sous la forme d'un réseau de neurones à fonctions de bases radiales (RBF)

2.2 Apprentissage incrémental

Dans un problème d'apprentissage incrémental en-ligne, les données d'apprentissage ne sont disponibles qu'au fur et à mesure de l'utilisation du système. Le classifieur doit donc être appris en utilisant les premières données arrivées, puis continuer à évoluer, de manière transparente du point de vue de l'utilisateur, lorsque les données suivantes arrivent.

Ainsi, les prototypes sont adaptés incrémentalement à l'arrivée de chaque nouvelle donnée. Ce processus d'adaptation permet de mettre à jour les centres des prototypes en fonction de chaque nouvelle donnée d'apprentissage disponible, et de recalculer de manière récursive les matrices de covariances des prototypes. Un algorithme de *clustering* incrémental [4] est utilisé pour créer de nouveaux prototypes, et donc de nouvelles règles, lorsque que cela est opportun.

L'apprentissage incrémental des conséquences dans un SIF d'ordre un peut être effectué par la méthode des Moindres Carrés Récursifs (MCR) pondérés par les activations des règles [3]. Soit $\Theta^{(i)}$ la matrice de tous les paramètres des conséquences linéaires de la règle i :

$$\Theta^{(i)} = \begin{bmatrix} \theta_{1,1}^{(i)} & \dots & \theta_{1,c}^{(i)} \\ \vdots & & \vdots \\ \theta_{n,1}^{(i)} & \dots & \theta_{n,c}^{(i)} \end{bmatrix} \tag{7}$$

où c représente le nombre de classes, et n la taille du vecteur des caractéristiques. Ces matrices peuvent être récursivement apprises :

$$\Theta_t^{(i)} = \Theta_{t-1}^{(i)} + P_t^{(i)} \mathbf{x}_t \beta^{(i)}(\mathbf{x}_t)(\mathbf{y}_t - \mathbf{x}_t^\top \Theta_{t-1}^{(i)})\ ; \quad \Theta_0^{(i)} = 0 \tag{8}$$

Où les matrices de covariances $P^{(i)} = (\sum_{k=1}^{t} \mathbf{x}_k \cdot \mathbf{x}_k^\top)^{-1}$ sont également mises à jour récursivement :

$$P_t^{(i)} = P_{t-1}^{(i)} - \frac{P_{t-1}^{(i)} \mathbf{x}_t \mathbf{x}_t^\top P_{t-1}^{(i)}}{\frac{1}{\beta^{(i)}(\mathbf{x}_t)} + \mathbf{x}_t^\top P_{t-1}^{(i)} \mathbf{x}_t}\ ; \quad P_0^{(i)} = 100 \cdot Id \tag{9}$$

2.3 Mesure de confiance

Nous utilisons les principes du rejet de confusion pour mesurer la confiance du système lors de la reconnaissance. De manière classique, le rejet de confusion est évalué sur les probabilités d'appartenance aux différentes classes en sortie du classifieur. Cependant, nous voulons faire du rejet, c'est-à-dire évaluer la qualité du modèle de notre classifieur, à un stade très précoce de l'apprentissage de notre système. À ce stade, les conclusions des règles d'inférence sont encore assez approximatives et variables, et ne sont pas assez représentatives de la confiance du système. Nous avons donc choisi de nous baser sur les prototypes des règles d'inférence qui sont beaucoup plus stables que les conclusions à ce stade de l'apprentissage. Même si toutes les règles participent à la reconnaissance de toutes les classes, chaque prototype est associé à une classe principale. Nous utilisons cela pour détecter les confusions lorsque plusieurs prototypes sont activés à des niveaux similaires par une donnée.

Notre mesure de confiance est calculée à partir de la distance de Mahalanobis entre une donnée $\mathbf{x}$ et les prototypes $C^{(i)}$ (définis par leur centre $\boldsymbol{\mu}^{(i)}$ et leur matrice de covariance $\Sigma^{(i)}$).

$$distance(C^{(i)}, \mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu}^{(i)})^\top (\Sigma^{(i)})^{-1} (\mathbf{x} - \boldsymbol{\mu}^{(i)})^\top \tag{10}$$

Cette distance est utilisée pour calculer une mesure de similarité plus lisse que l'activation des prototypes.

$$similarite(C^{(i)}, \mathbf{x}) = \frac{1}{1 + distance(C^{(i)}, \mathbf{x})} \tag{11}$$

Enfin, notre mesure de confiance est obtenue en évaluant la similarité entre la donnée et chaque prototype et en prenant la différence normalisée entre les deux plus grandes valeurs :

$$confiance = \frac{s_{first} - s_{second}}{s_{first}} \tag{12}$$

Où s_{first} et s_{second} représentent la plus forte valeur de similarité et la seconde respectivement. Cette mesure de confiance est utilisée pour rejeter les données qui donnent des valeurs inférieures à un certain seuil.
L'optimisation de ce seuil de rejet est un problème multi-objectif : il faut maximiser à la fois la performance : $Performance = N_{Correct}/N_{Total}$ et la précision du classifieur : $Precision = N_{Correct}/(N_{Correct} + N_{Erreurs})$. Où $N_{Correct}$ est le nombre de données correctement classées par le classifieur, N_{Errors} le nombre de données mal classées, et N_{Total} le nombre total de données. Lorsque le seuil augmente, le nombre de données rejetées augmente et le nombre d'erreurs de classification diminue.Un seuil de rejet haut va engendrer un fort taux de rejet, ce qui va augmenter la précision, alors qu'un seuil de rejet bas va générer peu de rejet, ce qui va augmenter la performance du système. Il faut donc faire un compromis entre la performance et la précision – entre le taux de rejet et le taux d'erreur – du classifieur.
Ce compromis dépend du coût associé à une erreur de classification et de celui associé au rejet d'une donnée à reconnaitre. D'un côté un rejet oblige l'utilisateur à valider ou corriger l'étiquette reconnue par le classifieur. De l'autre côté, une erreur de reconnaissance oblige l'utilisateur à annuler sa commande et à la recommencer.

3 Stratégies de supervision de l'apprentissage en-ligne

Dans le contexte des commandes gestuelles, l'utilisateur initialise le système en réalisant quelques exemples de gestes (trois dans notre expérimentation). Pour améliorer la reconnaissance des commandes gestuelles, le classifieur apprend de façon incrémentale au fur et à mesure de son utilisation. Il est donc nécessaire d'étiqueter les données au fur et à mesure de l'utilisation.

3.1 Supervision de l'apprentissage en-ligne

Supervision implicite. En pratique lors de l'utilisation du système, l'utilisateur dessine un geste qui est reconnu par le classifieur et la commande correspondante est effectuée. Plusieurs cas sont alors possibles.

- L'utilisateur annule la commande, soit parce qu'elle ne correspond pas au geste qu'il a dessiné (erreur de reconnaissance du classifieur), soit parce qu'il s'est trompé de geste (erreur de mémorisation de l'utilisateur), ou soit parce qu'il a juste changé d'avis.
- L'utilisateur continue ses actions, ce qui laisse vraisemblablement penser que la reconnaissance du geste précédent lui convient, il la valide implicitement.

L'étiquetage implicite consiste à utiliser l'étiquette reconnue par le classifieur lorsque l'utilisateur la valide implicitement (en effectuant une autre action que « annuler »). Le système va donc apprendre à partir des exemples qu'il a bien reconnus. Par contre, le système n'apprendra pas à partir des erreurs, soit du classifieur, soit de l'utilisateur. L'avantage de l'étiquetage implicite est qu'il ne dérange pas l'utilisateur pendant l'utilisation du système.

Supervision avec sollicitation explicite de l'utilisateur. Une autre manière de procéder est de solliciter l'utilisateur pour obtenir la bonne étiquette du geste qu'il a dessiné. Il apparaît comme évident que solliciter l'utilisateur après chaque commande est une stratégie fastidieuse pour l'utilisateur. Il faut donc sélectionner avec soin les données que nous lui demandons d'étiqueter. Pour cela, nous utilisons la capacité de rejet de notre classifieur *Evolve* pour solliciter ponctuellement l'utilisateur lorsque le résultat de la reconnaissance n'a pas un indice de confiance suffisamment élevé. Ainsi la vraie étiquette des gestes complexes à reconnaître est disponible et le classifieur peut apprendre à partir des données complexes sur lesquelles il aurait probablement commis une erreur.

3.2 Supervision avec un seuil de rejet constant

Pour optimiser l'apprentissage du système, on va pouvoir jouer sur le seuil de rejet du classifieur en tenant compte des impacts liés au risque d'une sollicitation trop fréquente de l'utilisateur. Il est donc nécessaire de trouver le bon compromis entre le nombre de rejets et le nombre d'erreurs de reconnaissance. La stratégie de supervision explicite la plus simple est de fixer un seuil de rejet constant optimisant de manière classique le compromis erreur/rejet. De cette manière, les exemples qui risquent le plus d'être mal reconnus sont rejetés. Les erreurs de reconnaissance et les rejets sont ainsi répartis de manière relativement homogène au court du temps. Il est cependant important de souligner que nous avons un système évolutif qui s'améliore au cours du temps. Par conséquent, avec un seuil fixe de rejet, implicitement le système va rejeter de moins en moins de données.

3.3 Supervision avec un seuil de rejet variable

Le compromis erreur/rejet est complexe dans cette situation d'apprentissage incrémental. Le rejet d'une donnée qui aurait été mal reconnue ne représente pas seulement une erreur d'évitée, mais également une donnée d'apprentissage supplémentaire pour améliorer le système. L'intuition, que l'on a voulu démontrer dans ce papier est que pour un même nombre de rejets, on obtient un système plus performant si l'on concentre le rejet en début de séquence d'apprentissage (pour accélérer cet apprentissage). Une deuxième stratégie consiste donc à rejeter davantage au début de l'utilisation du système, pour diminuer son taux d'erreurs rapidement, puis à réduire le taux de rejet pour minimiser le nombre d'interactions et aboutir à un point de fonctionnement optimal pour l'utilisateur.

4 Résultats expérimentaux

Notre objectif est d'obtenir un système avec les meilleures performances de reconnaissance possible, en tenant

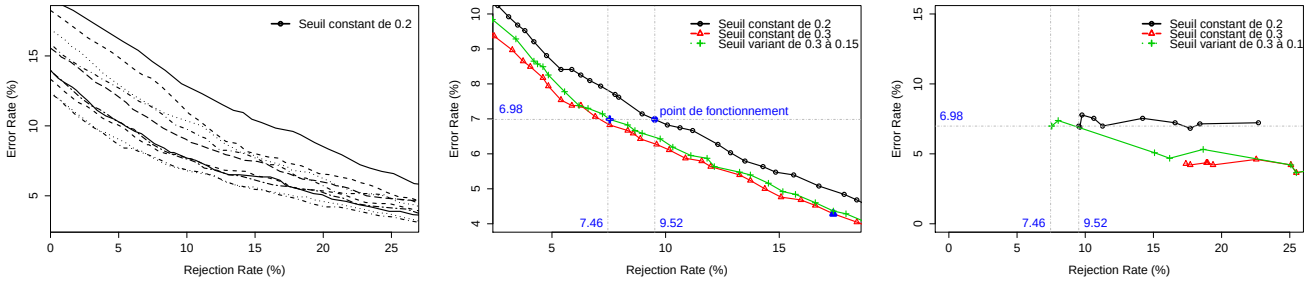

FIGURE 2 – (a) Évolution des courbes erreur/rejet – (b) Courbes finales – (c) Évolution des points de fonctionnements

Phase	0	1	2	3	4	5	6	7	8	9	Total
Seuil constant de 0.2	4.77	3.87	3.71	3.48	2.98	2.37	2.23	2.05	2.02	2.00	88.5
Seuil constant de 0.3	6.87	5.83	5.35	5.25	4.73	4.07	3.98	3.95	3.72	3.65	142
Seuil variable	6.87	5.83	5.35	5.25	3.90	3.40	3.17	1.67	1.55	1.57	115

TABLE 1 – Nombre moyen d'interactions utilisateurs par scripteur (sur les 21 de la base de test)

compte de la coopération induite par les sollicitations utilisateur. Nous avons donc comparé expérimentalement les stratégies de supervision avec un seuil de rejet constant (cf. section 3.2) et avec un seuil variable pour accélérer l'apprentissage au début (cf. section 3.3).

4.1 Protocole d'évaluation

Nous avons évalué nos différentes stratégies d'étiquetage et d'apprentissage sur la base de données ILGDB [2] [7] en utilisant le jeu de caractéristiques HBF49 [5] fournis.
Cette base de données contient 6629 gestes mono-stroke, appartenant à 21 classes, qui ont été saisis par 38 scripteurs dans un environnement immersif. Cette base de données est très intéressante pour plusieurs raisons. Tout d'abord, les gestes sont ordonnés chronologiquement suivant leur ordre de saisie ce qui permet de voir l'évolution de la manière de dessiner des scripteurs avec le temps. Ensuite, les fréquences des différentes classes varient, de 5 à 17 exemples par classe (par scripteur). Enfin, pour toute une partie de la base, les gestes des 21 classes ont été définis par les utilisateurs eux-mêmes. Ces caractéristiques font que cette base de données contient des gestes très réalistes et représentatifs de l'utilisation d'un classifieur en-ligne pour la reconnaissance de commandes gestuelles. En revanche, l'inconvénient de cette base est son faible nombre de gestes par scripteur (moins de 180) et par classe pour chaque scripteur (entre 5 et 17 gestes).
Afin de simuler une utilisation à plus long terme, nous avons utilisé 20 triplets d'utilisateurs qui avaient utilisé les mêmes gestes (groupe 3). Les symboles réalisés par chaque scripteur sont répartis en cinq phases. La phase 0 comporte trois symboles par classe et celle du premier utilisateur sert à l'initialisation du classifieur. Nous avons ensuite utilisé les phases 1 à 3 des trois scripteurs ($\sim$ 270 symboles) pour entrainer incrémentalement notre classifieur. Nous avons testé les performances de reconnaissance de notre système sur les phases 4 des trois scripteurs (63 symboles) entre chacune des neuf phases d'utilisations.

2. Disponible gratuitement : `http://www.irisa.fr/intuidoc/ILGDB.html`

4.2 Évolution de la courbe erreur/rejet

La figure 2 (a) présente l'évolution de la courbe erreur/rejet au cours du temps pour un seuil de rejet constant de 0.2. La courbe erreur/rejet s'améliore au cours du temps, le taux d'erreurs et le taux de rejet diminuent tous les deux car le classifieur apprend et s'améliore pendant son utilisation. On remarque que cette diminution est plus rapide au début de l'apprentissage, et que la courbe erreur/rejet finie par se stabiliser.

Courbes erreur/rejet finales. La figure 2 (b) présente les courbes erreur/rejet finales obtenues avec un seuil de rejet constant de 0.2 de 0.3 et un seuil de rejet variant de 0.3 à 0.15. Un seuil de rejet plus élevé (0.3) permet d'améliorer les performances du système par rapport à un seuil plus faible (0.2), cependant le point de fonctionnement engendré par ce seuil de rejet élevé n'est pas idéal, il sollicite souvent l'utilisateur. Faire varier le seuil de rejet permet d'obtenir une courbe finale aussi intéressante qu'avec un seuil de rejet élevé, tout en garantissant un point de fonctionnement optimal au niveau des interactions utilisateur.

4.3 Évolution des points de fonctionnements

La figure 2 (c) présente l'évolution des points de fonctionnements obtenus avec un seuil de rejet constant de 0.15 et un seuil de rejet variant de 0.3 à 0.15. Les points de fonctionnement se déplacent vers la gauche, le taux de rejet diminue, car le système s'améliore et les scores de confiance augmentent. Augmenter le taux de rejet au début de l'apprentissage permet d'améliorer l'apprentissage du système et d'obtenir ensuite, lorsque l'on réduit le rejet, un point de fonctionnement avec un taux de rejet plus faible (7.46% à la place de 9.52%, soit une diminution relative de 22%) pour un taux de d'erreur similaire (6.98%). La stratégie d'interaction avec un seuil de rejet variable est optimale pour l'apprentissage incrémental d'un système. Elle per-

met d'atteindre un meilleur point de fonctionnement que celui obtenu par une stratégie "statique" sans pour autant pénaliser la coopération avec l'utilisateur.

Évolution du nombre d'interactions utilisateurs. Le tableau 1 présente le nombre moyen d'interactions utilisateurs après chaque phase d'utilisation, et donc d'apprentissage. Le nombre de sollicitations moyen sur les dix tests passe de 142 pour le seuil constant de 0.3 à 115 pour le seuil variable (soit 23% de diminution relative) pour un taux d'erreur similaire. Augmenter de 31% le nombre d'interactions utilisateur au début (par rapport à un seuil constant de 0.2) permet de le diminuer de 21% à la fin de notre expérimentation, et donc pour tout le reste de son utilisation future.

5 Conclusion

L'apprentissage d'un classifieur pour la reconnaissance de commandes gestuelles est un problème d'apprentissage en-ligne qui nécessite d'étiqueter les données d'utilisation. Pour cela, il est nécessaire de solliciter l'utilisateur pour étiqueter les données mal reconnues et pouvoir apprendre efficacement. Cependant, solliciter constamment l'utilisateur est fastidieux pour celui-ci et réduit considérablement l'intérêt des commandes gestuelles. Il faut donc faire un compromis entre le nombre de sollicitations de l'utilisateur et le nombre d'erreurs de reconnaissances.

Nous avons étudié les impacts du rejet pour la supervision de l'apprentissage en-ligne de commandes gestuelles, et comment optimiser la coopération entre l'utilisateur et le système. Ainsi, faire varier le taux de sollicitation au cours du temps permet de solliciter davantage l'utilisateur au début de l'utilisation du système, ce qui va accélérer l'apprentissage et donc améliorer les performances du classifieur. Cette amélioration plus rapide des performances permet ensuite de réduire le taux d'interactions car le système commet moins d'erreurs.

Nous avons comparé expérimentalement deux stratégies de supervision de l'apprentissage par le rejet, étiquetant plus ou moins de données, et sollicitant plus ou moins l'utilisateur. La première est d'utiliser un seuil constant, alors que la deuxième est d'utiliser un seuil variable : de partir d'un seuil plus élevé et de revenir ensuite à une valeur plus faible. Cette deuxième stratégie permet d'accélérer l'apprentissage du système au début de son utilisation en étiquetant davantage de données. La courbe erreur/rejet finale et le point de fonctionnement ainsi obtenu sont plus intéressants : le taux de rejet est plus faible de 22% pour un même taux d'erreur.

Cette nouvelle stratégie de supervision, avec un seuil de rejet variable, améliore les performances du système sur nos expérimentations réalisées à partir d'une base de données existante. Il serait maintenant intéressant d'étudier son impact en situation réelle, et en particulier ses conséquences sur le comportement des utilisateurs. De plus, faire des expérimentations sur un temps d'utilisation plus long avec un même scripteur devrait encore amplifier l'avantage de la stratégie présentée dans cet article.

Références

[1] Abdullah Almaksour and Eric Anquetil. Improving premise structure in evolving takagi-sugeno neuro-fuzzy classifiers. *Evolving Systems*, 2(1) :25–33, 2011.

[2] Abdullah Almaksour and Eric Anquetil. ILClass : Error-driven antecedent learning for evolving takagi-sugeno classification systems. *Applied Soft Computing*, 2013.

[3] P.P. Angelov and Xiaowei Zhou. Evolving fuzzy-rule-based classifiers from data streams. *IEEE Transactions on Fuzzy Systems*, 16(6) :1462–1475, 2008.

[4] Pplamen Angelov and Dimitar Filev. An approach to online identification of takagi-sugeno fuzzy models. *Systems, Man, and Cybernetics, Part B : Cybernetics, IEEE Transactions on*, 34(1) :484–498, 2004.

[5] Adrien Delaye and Eric Anquetil. HBF49 feature set : A first unified baseline for online symbol recognition. *Pattern Recognition*, 46(1) :117–130, 2013.

[6] Peiyu Li, Manuel Bouillon, Eric Anquetil, and Grégoire 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*, volume 2, pages 337–355, 2013.

[7] Ney Renau-Ferrer, Peiyu Li, Adrien Delaye, and Eric Anquetil. The ILGDB database of realistic pen-based gestural commands. In *Proceeding of the 21st International Conference on Pattern Recognition*, pages 3741–3744, 2012.

[8] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. User-defined gestures for surface computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1083–1092, New York, NY, USA, 2009. ACM.

[9] Jacob O. Wobbrock, Andrew D. Wilson, and Yang 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, pages 159–168, New York, NY, USA, 2007. ACM.

[10] Jufeng Yang, Jing Xu, Mingda Li, Dingzhen Zhang, and Congchao Wang. A real-time command system based on hand gesture recognition. In *2011 Seventh International Conference on Natural Computation (ICNC)*, volume 3, pages 1588–1592, 2011.