Stratégies de supervision pour l'apprentissage en-ligne d'un classifieur évolutif de commande gestuelles

Manuel Bouillon, Eric Anquetil

Stratégies de supervision pour l'apprentissage en-ligne d'un classifieur évolutif de commande gestuelles

Manuel Bouillon* — **Eric Anquetil***

* *Université Européenne de Bretagne*
INSA de Rennes, Avenue des Buttes de Coesmes, 35043 Rennes
IRISA, CNRS UMR 6074, Campus de Beaulieu, 35042 Rennes

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 différentes stratégies d'étiquetage du flux de données d'utilisation pour l'apprentissage en-ligne du classifieur. Nous comparons sept stratégies de supervision, dépendants des interactions utilisateur (sollicitation par le système) et des capacités d'auto-évaluation du classifieur (notion de rejet). Nous montrons dans cet article que la stratégie donnant les meilleurs résultats est d'apprendre à partir des données validées implicitement par l'utilisateur, et de celles soumises à validation explicite car rejetées par le classifieur.*

ABSTRACT. *Touch sensitive interface 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 samples, and evolving, able to learn and improve during its use. This work studies several methods of labeling run-time data for the classifier on-line training. We compare seven supervision strategies, depending on user interactions, and system self-evaluation capacities (notion of reject). We show in this paper that the strategy giving the best results is to learn from data implicitly validated by the user, and from data explicitly validated because rejected by the classifier.*

MOTS-CLÉS : *Commande gestuelles, classifieur évolutif, apprentissage en-ligne, rejet.*

KEYWORDS: *Gesture commands, evolving classifier, online learning, 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] (Wobbrock *et al.*, 2009)(Yang *et al.*, 2011) permettent à l'utilisateur d'exécuter de nombreuses actions simplement en traçant les symboles associés. Pour mémoriser facilement une douzaine de commandes, il est essentiel de laisser l'utilisateur choisir lui-même ses propres gestes (Li *et al.*, 2013). 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.

Figure 1. *Exemple d'interaction par commande gestuelle.*

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 basés sur des algorithmes d'alignement, comme le *$1 classifier* (Wobbrock *et al.*, 2007) 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. Il faut donc que le classifieur soit évolutif : qu'il soit capable d'apprendre en-ligne pendant son utilisation.

Les classifieurs évolutifs sont apparus ces dernières années pour répondre au besoin de classification dans un environnement/contexte non stationnaire. Ils utilisent

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

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* (Almaksour et Anquetil, 2013) 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. La stratégie consiste donc, 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 des capacités d'auto-évaluation du classifieur pour solliciter l'utilisateur uniquement en cas de confusion potentielle. Autrement dit, il est nécessaire de sélectionner intelligemment les symboles que l'on fait étiqueter par l'utilisateur.

Le degré de confiance relatif à la réponse du classifieur permet d'appréhender deux grands types de critères de rejet : le rejet de distance et le rejet de confusion. Nous utilisons, pour les stratégies présentées dans cet article, une mesure de confiance évaluant le degré de confusion du reconnaisseur afin de rejeter les exemples pour lesquels le classifieur « hésite » entre deux réponses, c'est donc du rejet de confusion.

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 les différentes stratégies 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. Systèmes d'inférence floue 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 entraîner 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 d'inférence floue*

Nous travaillons ici avec un systèmes d'inférence floue (SIF) d'ordre un – dit de Takagi-Sugeno – qui offre de bonnes performances face à des problèmes de classi-

fication 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 (Angelov et Zhou, 2008) et par (Almaksour et Anquetil, 2011).

Un système d'inférence floue d'ordre un se compose d'un ensemble de règles de la forme suivante :

$$\textbf{Règle}^{(i)} : \textbf{SI } \mathbf{x} \text{ appartient à } C^{(i)} \textbf{ ALORS } \hat{\mathbf{y}}^{(i)} = (l_1^{(i)}(\mathbf{x}) ; \ldots ; l_c^{(i)}(\mathbf{x})) \quad [1]$$

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 $C^{(i)}$:

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

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 :

$$\hat{\mathbf{y}}^{(i)}(\mathbf{x}) = (l_1^{(i)}(\mathbf{x}) ; \ldots ; l_c^{(i)}(\mathbf{x})) \quad [3]$$

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

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

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

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

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 $\hat{\mathbf{y}}(\mathbf{x}) = (\hat{y}_1(\mathbf{x}) ; \ldots ; \hat{y}_c(\mathbf{x}))$:

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

La figure 2 représente un système d'inférence floue (SIF) du premier ordre sous la forme d'un réseau de neurones à fonctions de bases radiales (RBF).

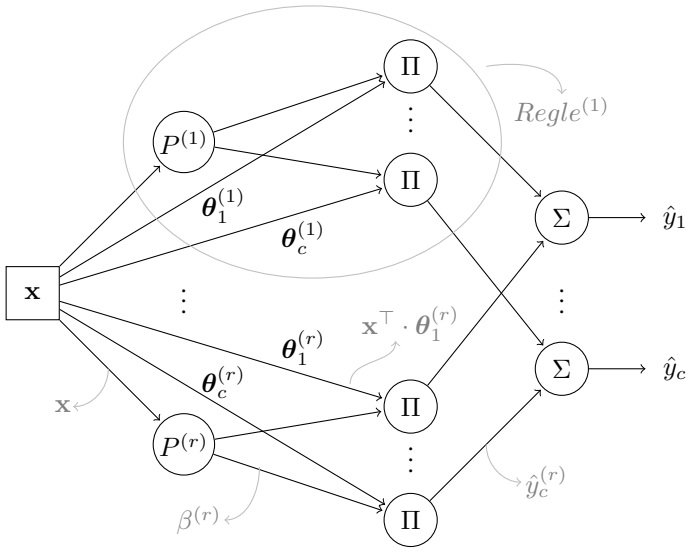

Figure 2. *Système d'inférence floue (SIF) du premier ordre sous la forme d'un réseau de neurones à fonctions de bases radiales (RBF)*

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 disponibles, 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 (Angelov et Filev, 2004) est utilisé pour créer de nouveaux prototypes, et donc de nouvelles règles, lorsque 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 (Angelov et Zhou, 2008). Soit $\Theta^{(i)}$ la matrice de tous les paramètres des conséquences linéaires de la règle i :

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

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

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

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 :

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

2.3. *Mesure de confiance et seuil de rejet*

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 classifier. 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 $\Sigma^{(i)}$).

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

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

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

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 :

$$confiance = \frac{s_{first} - s_{second}}{s_{first}} \qquad [12]$$

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 et la précision du classifieur :

$$Performance = N_{Correct}/N_{Total} \quad [13]$$

$$Precision = N_{Correct}/(N_{Correct} + N_{Erreurs}) \quad [14]$$

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.

En parallèle de l'apprentissage du classifieur, l'utilisateur apprend également : il doit mémoriser quel geste est associé à quelle commande (Li *et al.*, 2013). Dans cette situation d'apprentissage croisé, plusieurs cas sont possibles :

Cas A l'utilisateur dessine le bon geste et celui-ci est bien reconnu : la commande souhaitée est exécutée ;

Cas B l'utilisateur se trompe de geste ;

Cas C le classifieur se trompe lors de la reconnaissance du geste ;

Cas D le classifieur rejette le geste et le système demande confirmation à l'utilisateur.

Lorsque l'utilisateur se trompe de geste, ou lorsque le classifieur se trompe lors de la reconnaissance, la commande exécutée ne correspond pas à celle souhaitée. L'utilisateur doit alors annuler la commande exécutée et ré-essayer. Les données peuvent ainsi être réparties en quatre catégories comme illustré figure 3.

Afin d'entrainer le classifieur incrémentalement, il est nécessaire d'étiqueter les données au fur et à mesure de l'utilisation. Pour cela plusieurs stratégies peuvent être

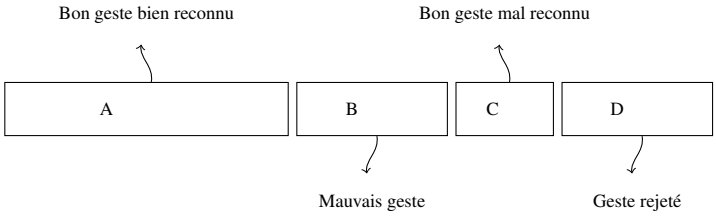

Figure 3. *Partitionnement des données en fonction du résultat de l'apprentissage croisé de l'utilisateur et du classifieur*

envisagées : des stratégies implicites, sans solliciter l'utilisateur, et des stratégies explicites, qui vont interagir avec l'utilisateur pour obtenir la bonne étiquette.

3.1. *Supervision basée sur un étiquetage implicite, sans sollicitation utilisateur*

L'avantage des stratégies d'étiquetage implicites est qu'elles ne dérangent pas l'utilisateur pendant son utilisation du système. Dans cette section, l'utilisation de rejet n'est pas obligatoire et la dernière catégorie (partie D) du partitionnement des données de la figure 3 peut être vide.

3.1.1. *Stratégie 1 : utilisation de l'étiquette reconnue*

Une première stratégie simpliste consiste à étiqueter les données sans solliciter l'utilisateur, en utilisant l'étiquette reconnue par le classifieur. Cependant, cet étiquetage va comporter des erreurs à chaque fois que des gestes sont mal reconnus. Dans ces cas (partie C de la figure 3), le classifieur va apprendre avec cette étiquette erronée, ce qui risque de le renforcer dans son erreur et de dégrader la qualité de son modèle.

3.1.2. *Stratégie 2 : utilisation de l'étiquette reconnue si validation implicite, pas d'apprentissage sinon*

En pratique, 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 qu'il laisse vraisemblablement penser que la reconnaissance du geste précédent lui convient, il la valide implicitement.

Une seconde stratégie d'apprentissage est de toujours utiliser l'étiquette reconnue par le classifieur, mais de n'apprendre que 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 (partie A de la figure 3), mais il n'apprendra pas à partir des erreurs – soit du classifieur (partie C), soit de l'utilisateur (partie B) – plutôt que de risquer d'apprendre avec une mauvaise étiquette. Cette solution permet d'être sûr de ne pas dégrader le modèle du classifieur, mais limite les données disponibles pour l'apprentissage.

3.2. *Supervision basée sur un étiquetage 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 le système lui demande d'étiqueter.

3.2.1. *Stratégie 3 : supervision basée sur un étiquetage explicite par sollicitation utilisateur en cas de rejet*

Une troisième stratégie est d'utiliser 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 sur lesquelles il aurait probablement commis une erreur (partie D de la figure 3). En revanche, comme le nombre de rejet doit rester le plus faible possible pour minimiser les sollicitations utilisateur, peu de données sont disponibles pour l'apprentissage.

3.2.2. *Stratégie 4 : supervision basée sur un étiquetage explicite en cas de rejet et sur l'étiquette reconnue sinon*

Une quatrième stratégie est d'associer la stratégie précédente, d'étiquetage par l'utilisateur en cas de rejet du classifieur, avec la stratégie d'étiquetage implicite avec l'étiquette reconnue par le classifieur (lorsqu'il n'y a pas de rejet). Toutes les données (parties A, B, C et D) servent alors à l'apprentissage, mais là encore certaines peuvent être mal étiquetées (données non rejetées mais mal reconnues).

3.2.3. *Stratégie 5 : supervision consolidée basée sur un étiquetage explicite en cas de rejet et implicite si l'utilisateur poursuit ses actions*

Enfin, une cinquième stratégie est de reprendre la précédente mais sans utiliser les données B et C de la figure 3, car leur étiquette n'est ni validée implicitement ni explicitement. Seules les données A (validées implicitement) et D (validées explicitement) sont donc utilisées pour l'apprentissage. Ce choix enlève quelques données pour l'apprentissage, mais permet d'être sûr de ne pas dégrader le modèle du classifieur en apprenant sur des données mal étiquetées.

3.2.4. *Stratégie 6 : utilisation de l'étiquette reconnue si validation implicite et sollicitation de l'utilisateur sinon*

Une dernière stratégie pourrait être d'utiliser l'étiquette reconnue si elle est validée implicitement, et sinon de solliciter l'utilisateur afin d'obtenir la bonne étiquette explicitement. Cependant, le taux de mémorisation des gestes varie entre 90% et 95% (5.71% d'erreur en moyenne pour des gestes personnalisés (Li *et al.*, 2013)), et ce taux est très variable d'un utilisateur à l'autre. L'utilisateur serait donc sollicité très souvent : lorsqu'il se trompe de geste (partie B : 5.71%), lorsque le classifieur fait une erreur de reconnaissance (partie C) et lorsque le classifieur rejette le geste (partie D). De plus, l'utilisateur serait également sollicité chaque fois qu'il annule sa commande pour une autre raison (changements d'avis) mais que la commande effectuée était bien celle souhaitée initialement.

Le tableau 3.2.4 résume quelle partie des données est utilisée par chaque stratégie de supervision.

	Données A	Données B	Données C	Données D
Stratégie 1	**oui**	**oui**	**oui**	**oui** (*sans IU*)
Stratégie 2	**oui**	non	non	non
Stratégie 3	non	non	non	**oui** (avec IU)
Stratégie 4	**oui**	**oui**	**oui**	**oui** (avec IU)
Stratégie 5	**oui**	non	non	**oui** (avec IU)
Stratégie 6	**oui**	**oui** (avec IU)	**oui** (avec IU)	**oui** (avec IU)

Tableau 1. *Récapitulatif des données utilisées par les stratégies de supervision*

4. Résultats expérimentaux

Notre objectif est d'obtenir un système avec les meilleures performances de reconnaissance possible, mais sans solliciter l'utilisateur trop souvent. Nous avons donc comparé expérimentalement les six stratégies de supervision (et une stratégie de référence sans apprentissage) énoncées précédemment et résumées ci-dessous.

Stratégie 0 Pas d'apprentissage en-ligne (stratégie de référence) ;

Stratégie 1 Apprentissage avec l'étiquette reconnue par le classifieur ;

Stratégie 2 Apprentissage avec l'étiquette reconnue par le classifieur si validation implicite de l'utilisateur (pas d'apprentissage sinon) ;

Stratégie 3 Apprentissage avec étiquetage par l'utilisateur si le geste est rejeté par le classifieur (pas d'apprentissage sinon) ;

Stratégie 4 Apprentissage avec étiquetage par l'utilisateur si le geste est rejeté par le classifieur et avec l'étiquette reconnue par le classifieur sinon ;

Stratégie 5 Apprentissage avec étiquetage par l'utilisateur si le geste est rejeté par le classifieur et avec l'étiquette reconnue par le classifieur si validation implicite de l'utilisateur (pas d'apprentissage sinon) ;

Stratégie 6 Apprentissage avec étiquetage par l'utilisateur pour chaque geste non validé implicitement (cas où l'étiquetage est parfait).

4.1. *Protocole d'évaluation*

Nous avons évalué nos différentes stratégies d'étiquetage et d'apprentissage sur la base de données ILGDB[2] (Renau-Ferrer *et al.*, 2012) en utilisant le jeu de caractéristiques HBF49 (Delaye et Anquetil, 2013) fournis.

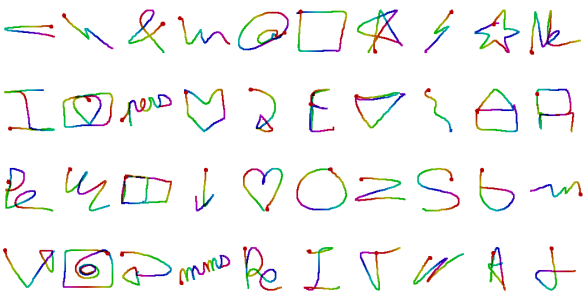

Figure 4. *Exemples de gestes du groupe 1 (gestes libres) de la base ILGDB.*

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). Des exemples des gestes inventés par les scripteurs sont présentés figure 4.

Les symboles réalisés par chaque scripteur sont répartis en cinq phases. La phase 0 comporte trois symboles par classe et sert à l'initialisation du classifieur. Nous avons ensuite utilisé les phases 1 à 3 ($\sim$ 90 symboles) pour entrainer incrémentalement

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

notre classifieur. Enfin, nous avons testé les performances de reconnaissance de notre système sur la phase 4 (21 symboles).

4.2. *Seuil de rejet*

Le seuil de rejet est relativement difficile à fixer dans notre contexte applicatif où l'on apprend avec très peu d'exemples et où chaque utilisateur peut définir son propre jeu de commandes/symboles, qui peut donc être plus ou moins complexe à appréhender par le classifieur. Le seuil de rejet doit donc être ré-estimé automatiquement pour chaque utilisateur, d'une manière suffisamment simple et robuste pour donner de bons résultats avec le peu de données disponibles pour le calculer.

Nous avons donc choisi d'entrainer le classifieur sur deux des trois exemples d'initialisation, puis nous mesurons le score de confiance lors de la reconnaissance du troisième. Nous avons ensuite calculé la moyenne μ_{reco} et l'écart-type σ_{reco} des données bien reconnues. Nous avons fixé le seuil de rejet à un écart-type en dessous de la confiance moyenne des données bien reconnues :

$$seuil = \mu_{reco} - \sigma_{reco} \quad [15]$$

A posteriori, nous avons construit la courbe erreur/rejet moyenne en figure 5 et nous avons placé le point de fonctionnement obtenu avec les seuils de rejet estimés automatiquement. Ce point de fonctionnement est très proche de la droite de coefficient directeur 0.5, soit deux fois plus de rejets que d'erreurs. Ce seuil de rejet est très satisfaisant si l'on considère que le coût d'une erreur est deux fois supérieur à celui d'un rejet. Si le rapport est différent, il est possible de modifier le seuil pour déplacer le point de fonctionnement.

4.3. *Supervision basée sur un étiquetage implicite, sans sollicitation utilisateur*

	Taux d'erreur total (%)	Taux d'erreur non rejetée (%)	Taux de rejet (%)
Stratégie 0	11.5	4.01	18.0
Stratégie 1	12.9	8.27	8.65
Stratégie 2	9.69	3.88	10.1

Tableau 2. *Résultats avec étiquetage implicite, sans sollicitation utilisateur*

Les résultats obtenus avec les stratégies d'étiquetage implicites sont présentés dans le tableau 2. La stratégie 1 qui consiste à apprendre avec la sortie du classifieur comme étiquette dégrade les performances de 1.4% (12% d'accroissement relatif) par rapport à la stratégie de référence sans apprentissage. L'apprentissage sur des données mal étiquetées est contre-productif, il vaut mieux ne pas apprendre du tout.

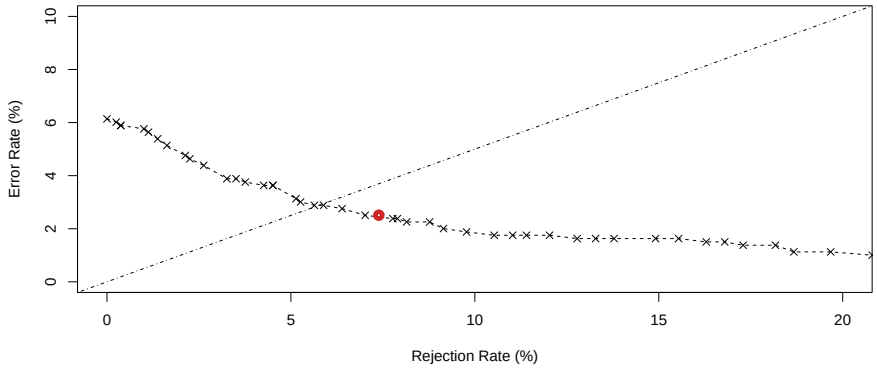

Figure 5. *Courbe erreur/rejet a posteriori moyenne.*

Par contre, apprendre en cas de validation implicite de la reconnaissance permet d'améliorer les performances, en renforçant le modèle du classifieur. La stratégie 2 permet ainsi de réduire de 1.81% le taux d'erreur (16% de diminution relative).

4.4. *Supervision basée sur un étiquetage avec sollicitation explicite de l'utilisateur*

	Taux d'erreur total (%)	Taux d'erreur non rejetée (%)	Taux de rejet (%)
Stratégie 3	7.27	2.38	12.7
Stratégie 4	6.77	3.26	7.64
Stratégie 5	6.14	2.51	7.39
Stratégie 6	5.51	1.63	9.02

Tableau 3. *Résultats avec étiquetage explicite par sollicitation de l'utilisateur*

Les résultats obtenus avec les stratégies de supervision explicites sont présentés dans le tableau 3. La stratégie 3 qui consiste à solliciter l'utilisateur pour étiqueter les données en cas de rejet permet d'apprendre efficacement, elle permet de diminuer le taux d'erreur de 4.23% (37% de diminution relative). Pouvoir apprendre à partir des données mal ou difficilement reconnues est très important pour améliorer le modèle du classifieur. La stratégie 4, qui consiste à apprendre avec la sortie du classifieur pour les données qui ne sont pas rejetées n'améliore que très légèrement les performances par rapport à la stratégie 3 (0.50% soit 7% de diminution relative).

La stratégie 5, qui combine les stratégies 2 et 3, permet d'obtenir des performances proches du cas où l'étiquetage est idéal (stratégie 6). Ne pas apprendre sur les données

qui ne sont ni rejetées ni validées implicitement diminue le taux d'erreur de 0.63% (9% de diminution relative). Là encore, mieux vaut ne pas apprendre qu'apprendre sur des données potentiellement mal étiquetées. De plus, cette stratégie permet de ne solliciter l'utilisateur que pour 7.39% des données, alors que la stratégie 6 nécessite de solliciter l'utilisateur 16.4% du temps (5.71% d'erreur de mémorisation, 1.63% d'erreur de classification, 9.02% de rejet).

	erreur (%)	Sollicitations (%)	$C_e = C_r$	$C_e = 2 * C_r$
Stratégie 5	2.51	7.39	9.90	12.4
Stratégie 6	1.63	16.4	16.4	18.0

Tableau 4. *Coûts des stratégies en fonction des coûts d'erreur C_e et de rejet C_r.*

Même si les coûts d'une erreur et d'un rejet sont difficiles à estimer sans sondage auprès des utilisateurs, on peut raisonnablement penser qu'un rejet a un coût plus faible qu'une erreur de reconnaissance. Un rejet sollicite l'utilisateur juste pour confirmer (ou corriger) l'étiquette reconnue, alors qu'une erreur oblige l'utilisateur à annuler sa commande et à ré-essayer. Le tableau 4 compare la stratégie 5 au cas où l'étiquetage est parfait (l'utilisateur est sollicité pour les parties B, C et D de la figure 3) en fonction des coûts d'erreur (C_e) et de rejet (C_r).

La stratégies 5, où toutes les données ne sont pas utilisées pour l'apprentissage, est moins coûteuse pour l'utilisateur que la stratégie 6, où l'étiquetage est parfait. Finalement, apprendre sur les données B et C de la figure 3 n'est pas intéressant car cela nécessite de solliciter l'utilisateur très souvent pour une faible réduction du taux d'erreur.

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, de nombreuses stratégies d'étiquetage existent, sollicitant plus ou moins l'utilisateur. D'une part, il est nécessaire de solliciter l'utilisateur pour étiqueter les données mal reconnues et pouvoir apprendre efficacement. D'autre part, 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 reconnaissance.

Nous avons comparé expérimentalement six stratégies d'étiquetage des données, étiquetant plus ou moins de données, plus ou moins bien, et sollicitant plus ou moins l'utilisateur. Il apparaît comme indispensable de bien étiqueter les données, sous peine de dégrader la qualité du modèle du classifier en le renforçant dans ses erreurs. En fait, il vaut mieux ne pas apprendre plutôt que d'apprendre sur des données mal étiquetées. Il reste cependant fondamental de pouvoir apprendre avec la bonne étiquette des données mal reconnues pour améliorer le classifieur. En particulier, l'utilisation du rejet

permet de solliciter l'utilisateur lorsqu'une donnée est difficilement ou mal reconnue, et qu'il sera très profitable de l'utiliser pour l'apprentissage, sans pour autant le solliciter trop souvent.

Ces travaux montrent l'importance de bien étiqueter les données d'apprentissage, et la difficulté de cette tâche dans le cadre de la reconnaissance de commandes gestuelles. Nous avons ainsi montré l'intérêt du rejet pour sélectionner les données à étiqueter, et mis en évidence le compromis erreurs/sollicitations. Pour optimiser précisément le seuil de rejet, et donc ce compromis erreurs/sollicitations, il sera nécessaire d'effectuer un sondage parmi un échantillon d'utilisateur pour déterminer plus précisément les coûts d'erreurs et de rejets (interaction utilisateur).

De plus, il pourrait être intéressant de faire varier le taux de sollicitation au cours du temps. Solliciter davantage l'utilisateur au début pourrait permettre d'améliorer plus vite les performances du classifieur, et donc de réduire le nombre d'interactions ensuite. De même, le seuil de rejet pourrait être recalculé incrémentalement pour suivre l'amélioration du modèle du classifieur.

6. Bibliographie

Almaksour A., Anquetil E., « Improving premise structure in evolving Takagi-Sugeno neuro-fuzzy classifiers », *Evolving Systems*, vol. 2, n° 1, p. 25-33, 2011.

Almaksour A., Anquetil E., « ILClass : Error-driven antecedent learning for evolving Takagi-Sugeno classification systems », *Applied Soft Computing*, 2013.

Angelov P., Filev D., « An approach to online identification of Takagi-Sugeno fuzzy models », *Systems, Man, and Cybernetics, Part B : Cybernetics, IEEE Transactions on*, vol. 34, n° 1, p. 484 - 498, 2004.

Angelov P., Zhou X., « Evolving Fuzzy-Rule-Based Classifiers From Data Streams », *IEEE Transactions on Fuzzy Systems*, vol. 16, n° 6, p. 1462-1475, 2008.

Delaye A., Anquetil E., « HBF49 feature set : A first unified baseline for online symbol recognition », *Pattern Recognition*, vol. 46, n° 1, p. 117-130, 2013.

Li P., Bouillon M., Anquetil E., Richard G., « User and System Cross-Learning of Gesture Commands on Pen-Based Devices », *Proceeding of the 14th International Conference on Human-Computer Interaction - INTERACT 2013*, vol. 2, p. 337-355, 2013.

Renau-Ferrer N., Li P., Delaye A., Anquetil E., « The ILGDB database of realistic pen-based gestural commands », *Proceeding of the 21st International Conference on Pattern Recognition*, p. 3741-3744, 2012.

Wobbrock J. O., Morris M. R., Wilson A. D., « User-defined gestures for surface computing », *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, ACM, New York, NY, USA, p. 1083-1092, 2009.

Wobbrock J. O., Wilson A. D., Li Y., « Gestures without libraries, toolkits or training : a $1 recognizer for user interface prototypes », *Proceedings of the 20th annual ACM symposium on User interface software and technology*, UIST '07, ACM, New York, NY, USA, p. 159–168, 2007.

Yang J., Xu J., Li M., Zhang D., Wang C., « A real-time command system based on hand gesture recognition », *2011 Seventh International Conference on Natural Computation (ICNC)*, vol. 3, p. 1588 -1592, 2011.