Apprentissage incrémental et synthèse de données pour la reconnaissance de caractères manuscrits en-ligne

A. Almaksour, Harold Mouchère, Eric Anquetil

Apprentissage incrémental et synthèse de données pour la reconnaissance de caractères manuscrits en-ligne

Abdullah Almaksour[1,2] – Harold Mouchère[1,2] – Eric Anquetil[1,2]

[1] IRISA
Campus Universitaire de Beaulieu, Avenue du Général Leclerc
35042 Rennes Cedex, France
[2] INSA de Rennes
20, Avenue des Buttes de Coësmes
35043 Rennes Cedex, France

{Abdullah.Almaksour, Harold.Mouchere, Eric.Anquetil}@irisa.fr

Résumé : *Dans ce papier, nous proposons une stratégie d'apprentissage incrémental d'un système de reconnaissance de caractères manuscrits. Cette stratégie est en-ligne et rapide, dans le sens où toute nouvelle classe de caractères peut être apprise par le système à partir de très peu d'exemples de caractères. La stratégie proposée vise à surmonter le problème du manque de données d'apprentissage lors de l'introduction d'une nouvelle classe de caractères. La synthèse de caractères manuscrits est utilisée à cette fin. Les résultats ont montré qu'un bon taux de reconnaissance (environ 90%) est atteint en utilisant seulement 3 exemples d'apprentissage. De plus, ce taux augmente rapidement pour atteindre 96% pour 10 exemples, et 97% pour 30. Une réduction d'erreur de 45% est obtenue en utilisant la synthèse de caractères par rapport à une stratégie sans synthèse.*

Mots-clés : synthèse de données, apprentissage incrémental, reconnaissance de caractères manuscrits.

1 Introduction

Avec l'émergence des assistants personnels numériques (PDA) et des téléphones mobiles de nouvelle génération (smartphone) utilisant des interfaces orientées stylo, les performances des systèmes de reconnaissance de caractères en-ligne se doivent d'être plus hautes. De plus en plus d'efforts sont nécessaires pour rendre ces systèmes plus robustes et adaptables afin de répondre à l'accroissement rapide des besoins des utilisateurs. Un de ces nouveaux besoins est un système de reconnaissance permettant à l'utilisateur de choisir son propre groupe de gestes et de les assigner à différentes commandes interactives, par exemple, « copier », « coller », « annuler », etc. Ce contexte applicatif impose des contraintes spécifiques à la technique d'apprentissage utilisée. Une telle technique doit être capable d'apprendre rapidement en utilisant peu de données. En effet, les utilisateurs sont rarement prêts à attendre ou à répéter chaque nouveau geste plus d'une douzaine de fois pour l'apprentissage du système. Pour répondre à ces besoins, cette étude propose un modèle original d'apprentissage incrémental rapide pour les systèmes de reconnaissance d'écriture manuscrite en-ligne. Nous validons ici notre modèle sur les lettres manuscrites, mais l'objectif à long terme est d'utiliser une telle technique dans n'importe quel système de reconnaissance de gestes manuscrits.

La difficulté principale dans la problématique proposée est de construire à la volée un classifieur d'écriture manuscrit à partir de zéro, avec la contrainte d'avoir peu de connaissances disponibles (quelques exemples d'apprentissage), puis de l'adapter progressivement afin d'atteindre de forts taux de reconnaissance le plus rapidement possible. Dans ce papier, nous proposons un modèle d'apprentissage incrémental en deux phases. Dans la première phase, correspondant à un apprentissage rapide, le système acquiert le maximum possible de connaissances. Par contre, dans la deuxième phase l'apprentissage se fait en adaptant les connaissances existantes. Ensuite, afin de surmonter le problème du manque d'exemples, et d'augmenter rapidement le taux de reconnaissance du système, nous intégrons dans le processus d'apprentissage les techniques originales de synthèse de caractères manuscrits, présentés dans de précédents travaux [MOU 06b].

Le reste de cet article est organisé comme suit. Dans la section 2, nous faisons un bilan des possibilités existantes pour l'apprentissage incrémental, et nous présentons la méthode de classification utilisée dans notre système. Ensuite la section 3 présente notre modèle de base pour l'apprentissage incrémental. La section 4 présente les méthodes de la synthèse de données. Puis nous proposons dans la section 5 la version améliorée du modèle en utilisant la synthèse de caractères. Enfin la section 6 présente le protocole et les résultats de nos expérimentations.

2 Etat de l'art

Nous commençons par présenter les approches existantes d'apprentissage incrémental puis la méthode de classification que nous avons choisie, un Système d'Inférence Floue (SIF).

2.1 Apprentissage incrémental

Dans beaucoup d'approches d'apprentissage automatique, la phase d'acquisition d'un ensemble représentatif de données d'apprentissage est souvent longue et coûteuse. Dans notre contexte, un nouveau classifieur doit être créé à

partir de zéro et le temps nécessaire pour construire un jeu de données représentatif doit être minimisé pour qu'il soit acceptable du point de vue de l'utilisateur. Pour cela, le classifieur doit être appris rapidement avec peu d'exemples, pour être ensuite mis à jour progressivement par chaque nouvel exemple disponible.

Un algorithme d'apprentissage incrémental est défini dans [POL 01] comme répondant aux critères suivants :

1. il doit être capable d'apprendre des connaissances supplémentaires à partir des nouvelles données,
2. il ne doit pas nécessiter l'accès aux données d'origine (c'est-à-dire les données qui sont été utilisées pour apprendre le classifieur actuel),
3. il doit préserver les connaissances déjà acquises,
4. et il doit être en mesure d'apprendre de nouvelles classes susceptibles d'être introduites avec des nouvelles données.

Ces quatre points, qui s'appliquent pour tout problème général de l'apprentissage incrémental, correspondent parfaitement aux caractéristiques particulières de notre problème, un algorithme d'apprentissage rapide pour un système de reconnaissance en-ligne des caractères manuscrits. Plusieurs techniques d'apprentissage incrémental ont été proposées dans différents contextes applicatifs et pour différentes approches de classification, dont certaines sont décrites ci-dessous.

Learn++ [POL 01] est un algorithme incrémental basé sur la performance synergique d'un ensemble de classifieurs faibles. Ces classifieurs sont appris et ajoutés au système de façon incrémentale après l'acquisition d'une certaine quantité de données. Cela nous a amené à considérer cette technique comme un processus d'apprentissage incrémental hors-ligne (c'est-à-dire non-instantané). Cependant, les classifieurs créés ne peuvent pas être adaptés après leur apprentissage. Par conséquent, l'ajout de nouveaux classifieurs est la seule méthode qui peut être utilisée par le processus d'apprentissage incrémental pour ajouter de nouvelles connaissances.

ILFN [GAR 01] est un système d'apprentissage incrémental basé sur les réseaux de neurones flous. Le classifieur dans ILFN utilise des fonctions gaussiennes à base radiale pour définir les frontières de décision. Il emploie un système d'apprentissage hybride combinant à la fois une approche supervisée et une autre non-supervisée pour générer ses prototypes. Cet algorithme vérifie les quatre caractéristiques décrites ci-dessus, et il peut être considéré comme stratégie d'apprentissage rapide. Par contre, il utilise une structure de prototype assez simple, qui ne peut faire face à la complexité du problème de reconnaissance de caractères manuscrits.

Fuzzy ARTMAP [CAR 92] est basé sur la création de nouveaux « clusters » de décision à partir de nouveaux exemples suffisamment différents de ceux vus précédemment. Le seuil de similarité doit être paramétré par l'utilisateur. L'inconvénient de cette technique est sa sensibilité à la sélection du seuil de similarité, au niveau de bruit dans les données d'apprentissage, et à l'ordre dans lequel les données sont présentées.

Les trois systèmes présentés dans cette section ne sont pas utilisables dans le contexte de l'apprentissage incrémental rapide de l'écriture manuscrit. L'objectif est alors de trouver un modèle d'apprentissage incrémental qui soit d'abord capable d'apprendre « instantanément » des nouvelles classes, comme dans ILFN. Et en même temps, ce système doit être basé sur un classifieur complexe et puissant, comme dans Learn++. Nous proposons dans cet article un modèle d'apprentissage original appelé AI2P. Afin d'améliorer la rapidité et la performance du système, nous utilisons la synthèse de caractères manuscrits artificiels. Cette idée est intégrée dans le modèle AI2P (Apprentissage Incrémental en 2 Phases), créant un nouveau modèle AI2P++. Ces deux modèles seront présentés respectivement dans les sections 3 et 5.

Le classifieur SIF que nous avons choisi d'utiliser est un classifieur léger basé sur des prototypes (voir section 2.2). La flexibilité des SIF répond à notre besoin d'une approche de classification adaptative.

2.2 Les principes de SIF

Les Systèmes d'Inférence Floue (SIF) étendent les principes des systèmes à base de règles classiques en modélisant les imperfections liées aux connaissances manipulées grâce aux outils de la théorie des sous-ensembles flous. Les mécanismes de raisonnement en résultant sont ainsi plus robustes et plus proches de la réalité [BOU 03].

Les SIF que nous utilisons sont du type Takagi-Sugeno d'ordre 0 constitués de N règles. Chaque règle R_r est composée d'une prémisse et d'une conclusion. La prémisse correspond à une modélisation intrinsèque d'une classe ou d'une partie d'une classe par un prototype flou P_r défini dans dans l'espace des caractéristiques E. La conclusion associe à chaque prototype son degré d'appartenance s_c^r à chaque classe c. Dans un problème à C classes, les règles s'écrivent donc :

SI X est P_r **ALORS** $s_1^r = a_1^r$ **... et** $s_c^r = a_c^r$ **... et** $s_C^r = a_C^r$

avec X la forme à reconnaître dans E et les a_c^r sont des valeurs constantes (ordre 0). Les prototypes flous P_r sont définis par le degré d'appartenance $\beta_r(\vec{X})$ d'une forme X à l'ensemble flou correspondant. On parle aussi du *degré d'activation* du prototype. Cette fonction d'appartenance correspond à une fonction à base radiale hyper-ellipsoïdale de centre μ_r et dont la forme est donnée par la matrice de covariance Q_r. Le degré d'appartenance utilise une fonction de Cauchy basée sur la distance de Mahalanobis $d_{Q_r}(X, \mu_r)$:

$$\beta_r(X) = \frac{1}{1 + d_{Q_r}(X, \mu_r)} \,. \tag{1}$$

Pour déterminer la classe d'une forme inconnue X, l'activation β_r de chacun des N prototypes flous est d'abord calculée. Ensuite chaque règle est appliquée et combinée par l'inférence floue de type *somme-produit* pour calculer un score s_c pour chaque classe :

$$s_c = \frac{\sum_{r=1}^{N} \beta_r s_c^r}{\sum_{r=1}^{N} \beta_r}. \tag{2}$$

Cette équation montre comment les différents prototypes participent à la reconnaissance de toutes les classes : plus un

prototype est activé et plus il appartient à la classe, plus il participe au score global de la classe.

3 AI2P : un modèle d'Apprentissage Incrémental en 2 Phases

Nous allons présenter d'abord le modèle en général, puis nous allons expliquer ses deux phases.

3.1 Les principes de AI2P

Il y a deux phases pour apprendre progressivement un système de reconnaissance à base de prototypes. La première consiste à créer un nouveau prototype pour chaque nouvel exemple. En revenche, la seconde consiste à modifier les prototypes existants à chaque nouvel exemple. Dans le modèle AI2P, nous avons proposé ces deux méthodes comme deux techniques complémentaires pour arriver à un apprentissage incrémental rapide et fiable.

Bien que la création de prototypes contribue à améliorer significativement le taux de reconnaissance, la complexité du classifieur peut devenir inacceptable - en termes de temps et de mémoire - si nous continuons à créer des prototypes pour chaque nouvel exemple. Pour cela, l'utilisation de l'adaptation devient indispensable afin d'avoir un système à la fois dynamique et léger. L'adaptation de prototypes peut améliorer le taux de reconnaissance, sans surcharger le système, et en créant « occasionnellement » de nouveaux prototypes. Nous utilisons pour cela la méthode ADAPT présentée dans [MOU 07]

En outre, nous supposons que les processus d'apprentissage et d'adaptation sont supervisés : chaque exemple est étiqueté correctement. Cet étiquetage est possible en demandant au scripteur de vérifier le résultat de la reconnaissance ou à l'aide d'une technique auto-supervisée comme dans [OUD 04].

La figure 1 présente le modèle AI2P, plus de détails sont présentés dans l'algorithme 1, puis dans les sections suivantes.

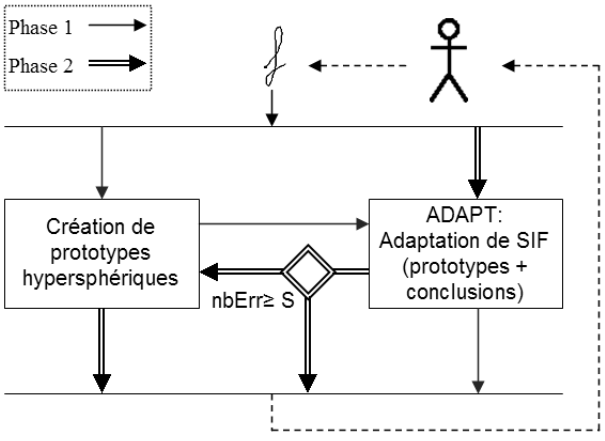

FIG. 1 – Le modèle AI2P, explicitant les deux phases.

Algorithme 1 : L'algorithme d'AI2P

pour chaque *nouvel exemple* e *de la classe* C **faire**
  si *classe* C *est dans phase 1* **alors**
    appeler l'algorithme de création de prototypes avec e;
    appeler l'algorithme d'adaptation;
    si *nombre d'exemples de classe* $C \geq N$ **alors**
      basculer vers phase 2
    fin
  fin
  si *classe* C *est dans phase 2* **alors**
    appeler l'algorithme d'adaptation;
    si e *est mal-classé* **alors**
      $nbErr[C] + +$;
      si $nbErr[C] \geq S$ **alors**
        appeler l'algorithme de création de prototypes;
        $nbErr[C] = 0$;
      fin
    fin
  fin
fin

3.2 Phase 1 : Apprentissage incrémental rapide

Dans cette phase, le processus d'apprentissage incrémental se compose de deux algorithmes principaux : la création de prototypes et le calcul des conclusions. Un prototype hypersphérique est créé autour de chaque nouvel exemple. Ses conclusions sont initialisées à 1 pour la conclusion qui correspond à la classe du caractère, et à 0 pour toutes les autres classes. Puis tous les prototypes existants et leurs conclusions sont adaptés par ADAPT pour ajuster globalement le système au nouvel exemple. Le système passe de la phase 1 à la phase 2 pour une classe donnée après avoir eu N exemples (N prototypes) de cette classe.

3.3 Phase 2 : Apprentissage par adaptation

Nous utilisons dans cette phase la méthode ADAPT, qui permet de modifier les prototypes de SIF en les déplaçant et déformant pour prendre en compte les connaissances acquises par les nouveaux exemples. Un nouveau prototype est créé pour une classe donnée quand le nombre d'erreurs de reconnaissance (nbErr) pour cette classe dépasse un seuil défini (S).

Dans la phase 1 et la phase 2, la création de prototypes est confrontée au problème du manque de données représentatives, puisque les prototypes sont créés à partir d'un exemple (dans la phase 1) ou S exemples(dans la phase 2) .

4 Synthèse de données artificielles

Afin d'obtenir un apprentissage incrémental rapide et de surmonter le manque de données disponibles au début du processus d'apprentissage, il existe deux stratégies que nous développons dans les sous-sections suivantes : synthèse dans l'espace de caractéristiques et synthèse dans l'espace de formes.

4.1 Synthèse dans l'espace de caractéristiques

Dans cette stratégie, après avoir calculé les caractéristiques en n dimensions qui représentent le caractère d'origine, nous ajoutons du bruit à chaque attribut dans le but de créer de nouveaux points dans l'espace de classification. Cette technique de base peut être appliquée pour générer des données d'apprentissage suffisantes pour les modèles de classification qui nécessitent un nombre minimal d'exemples pour représenter les nouvelles connaissances, par exemple les systèmes de type MLP ou SVM. Ce type de génération de données n'est pas utile pour notre système, car il n'y a pas de condition préalable sur le nombre de données pour créer un nouveau prototype. En effet nous pouvons créer un prototype simple, sphérique, autour du caractère original. Malheureusement cette approche ne peut pas contribuer à créer un prototype représentatif, car une sphère n'est pas capable de modéliser le style d'écriture de l'utilisateur qui est beaucoup plus complexe.

4.2 Synthèse dans l'espace de formes

Comme il est proposé dans [MOU 06b], différentes techniques de génération de l'écriture manuscrite en ligne peuvent être utilisées pour construire des exemples artificiels d'écriture, en restant fidèle au style de l'utilisateur.

Dans [MOU 06b], deux stratégies pour générer des caractères manuscrits ont été proposées. La première utilise les distorsions classiques de l'image (de type hors-ligne), comme l'étirement et l'inclinaison. La seconde, fondée sur la particularité de l'écriture manuscrite en-ligne, applique deux distorsions en-ligne sur le caractère : modification de la vitesse et modification de la courbure.

Il s'agit de générer plusieurs variations d'un même caractère à partir d'un seul exemple de ce caractère en-ligne. Pour cela nous appliquons, sur l'exemple disponible, une ou plusieurs des déformations présentées ci-dessus. À chaque nouvelle génération il faut choisir des paramètres de déformation différents pour obtenir une certaine variabilité dans les données générées. Mais il ne faut pas non plus utiliser des paramètres engendrant des déformations trop importantes pour ne pas risquer de synthétiser un caractère qui ne ressemble plus à celui d'origine. En effet, il s'agit de conserver le style du scripteur et non de générer un nouveau style. Pour chaque paramètre, une borne minimales et une borne maximale sont fixées pour limiter les déformations générées. Ces bornes sont fixées empiriquement en visualisant pour chacune d'elles les déformations obtenues pour plusieurs classes de caractères. Une fois fixées, les bornes restent les mêmes tout au long des expérimentions.

Il s'agit alors de créer des prototypes représentants au mieux ces données artificielles, ce qui est fait dans le modèle AI2P++.

5 AI2P++ : Accélérer l'apprentissage par la synthèse de caractères

En raison du manque de connaissance au début de l'apprentissage, la synthèse des caractères manuscrits va aider à améliorer la création de prototypes, en termes de qualité et de quantité de données d'apprentissage. Nous utilisons donc ici la synthèse dans l'espace des formes présentée dans la section 4.2 pour créer des prototypes flous représentant fidèlement les différentes variations d'un même caractère.

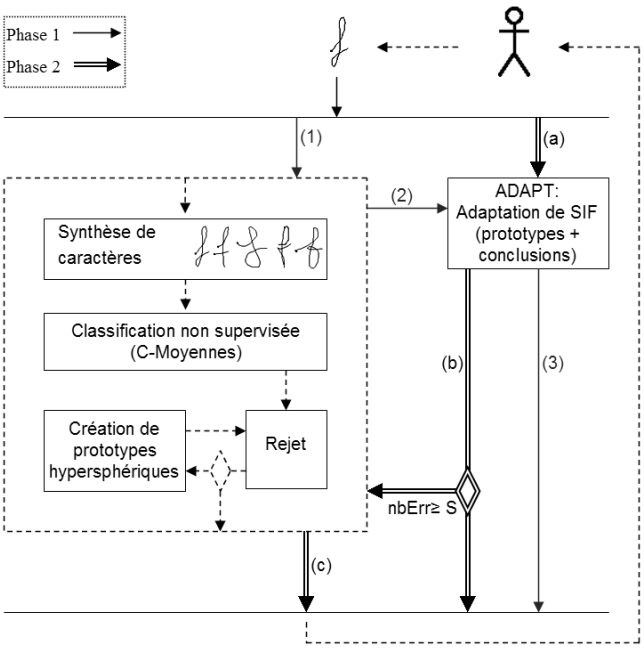

FIG. 2 – Le modèle AI2P++.

Utilisant les caractères générés et le caractère original, les prototypes flous sont appris en appliquant l'algorithme de classification non supervisée des C-Moyennes Possibilistes [KRI 93] sur chaque classe séparément. Ainsi les prototypes permettent une description intrinsèque de chaque classe. Ensuite, et toujours en profitant des caractères générés, les conclusions de chaque règle sont calculées grâce à la méthode de descente de gradient (et non plus initialisés à 0 et 1).

Dans un problème d'apprentissage incrémental rapide, nous avons besoin de représenter et de bénéficier de toutes les connaissances disponibles. Cependant, l'algorithme de classification non supervisée néglige les points isolés en les considérant comme du bruit, alors que dans notre contexte ces points isolés peuvent aider à modéliser le style de l'écriture du scripteur. Pour répondre à ce besoin, nous avons ajouté une extension sur l'algorithme de classification non supervisée pour qu'il soit capable de couvrir l'ensemble du nuage de points par le nombre minimum de prototypes possibles. Le mécanisme de rejet proposé dans [MOU 06a] est utilisé dans notre système pour distinguer les points qui ne sont pas couverts (et donc pas bien représentées par les prototypes). Un point donné est considéré comme rejeté si les activations de tous les prototypes créés sont inférieurs au seuil de rejet. Pour chaque point rejeté, un prototype hypersphérique est créé pour le représenter. L'Algorithme 2 formalise cette approche.

La figure 3 illustre la différence entre les deux approches AI2P et AI2P++ au niveau de la création des nouveaux prototypes (les données d'origines sont en gras).

Algorithme 2 : Création de prototypes dans AI2P++

```
Générer np points (caractères) artificiels;
Appliquer l'algorithme C-Moyennes sur les points
originaux et artificiels pour construire le prototype;
pour chaque point p faire
    si p est rejeté alors
        créer un nouveau prototype hypersphérique
        auteur de ce point.
    fin
fin
Calculer les conclusions pour les nouveaux prototypes;
```

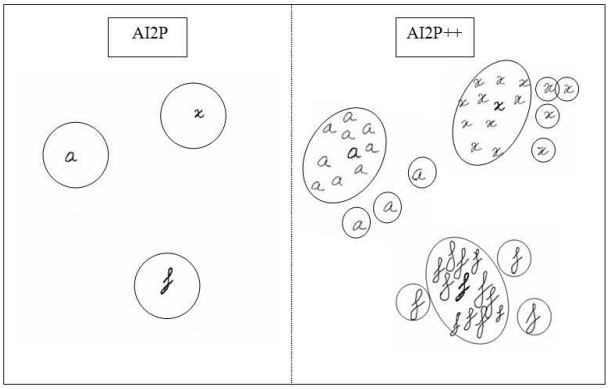

FIG. 3 – Représentation 2D des prototypes du SIF pour les deux modèles.

6 Expérimentations

Les expérimentations portent sur la reconnaissance des 26 lettres latines minuscules isolées en-ligne, sans aucune autre contrainte pour les scripteurs. Les bases de caractères mono-scripteur ont été écrites sur un PDA par onze scripteurs. Chaque scripteur a saisi 40 fois chaque caractère i.e. 1040 caractères par scripteur. Dans cette expérience, les caractères sont saisis dans un ordre aléatoire. Pour estimer les performances de l'adaptation sur chaque scripteur, nous procédons à un test mono-scripteur en utilisant le principe de la validation croisée stratifiée en quatre parties. Trois quarts de la base du scripteur (780 lettres) sont utilisés pour adapter le système à son style et un quart (260 lettres) est utilisée pour évaluer le taux de reconnaissance du système tout au long de l'adaptation. Les résultats que nous présentons correspondent à la moyenne sur les onze scripteurs du taux de reconnaissance moyen pour chaque un. Dans ces expériences, les caractères sont décrits par 21 caractéristiques. La phase d'apprentissage incrémental se termine pour chaque classe après avoir présenté 10 exemples de cette classe. Le système passe ensuite à la phase d'apprentissage par adaptation.

Pour chaque nouvel exemple d'apprentissage réel, lors de la création d'un nouveau prototype, une base synthétique de 300 caractères artificiels est générée par les distorsions d'image et les distorsions en-ligne (le nombre optimal de caractères artificiels générés à été expérimentent validé).

La figure 4 montre l'évolution du taux moyen de reconnaissance pour les 11 scripteurs. Un taux de reconnaissance d'environ 90 % est réalisé en utilisant seulement 3 exemples d'apprentissage réels. Ce taux augmente rapidement pour atteindre 96 % pour 10 exemples, et 97 % pour 30. La figure 4 montre aussi que l'utilisation du modèle AI2P++ fait diminuer l'erreur de reconnaissance de 45 % en moyenne, en comparant avec le modèle AI2P. La figure 5 montre le

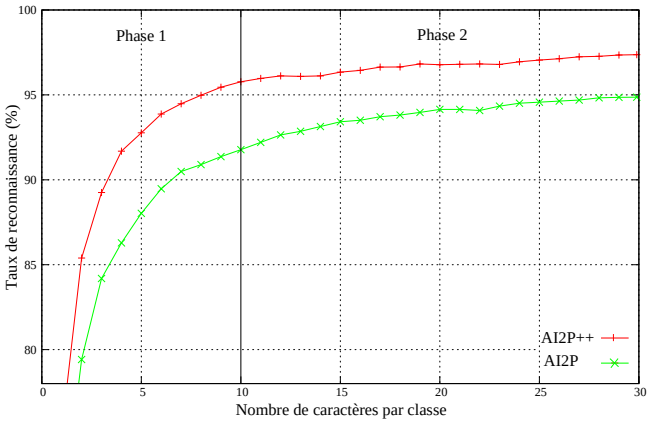

FIG. 4 – L'évolution du taux de reconnaissance lors du processus d'apprentissage.

nombre de prototypes de SIF lors du processus d'apprentissage. Nous pouvons constater que pendant de la phase 2, ce nombre augmente très lentement (8 prototypes pour 20 caractères par classe) alors que le taux de reconnaissance continue d'augmenter pendant cette phase.

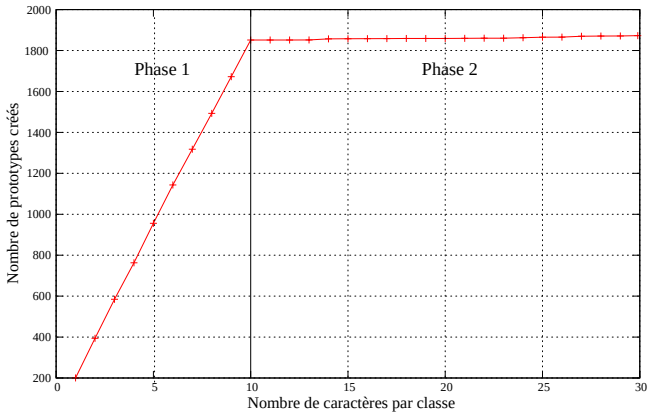

FIG. 5 – Nombre de prototypes de SIF lors des deux phases de AI2P++.

7 Conclusion et travaux futurs

Dans le contexte de la reconnaissance de l'écriture manuscrite en-ligne, nous avons présenté un nouveau modèle d'apprentissage incrémental basé sur un système d'inférence floue. Grâce à ce modèle, le système de reconnaissance est capable d'apprendre des nouvelles formes à partir de très peu de données. Il peut de plus s'adapter et s'améliorer pour chaque nouvelle donnée disponible. Nous avons intégré dans le modèle des techniques de synthèse de caractères manuscrits pour accélérer l'apprentissage et ainsi améliorer le taux de reconnaissance.

Les travaux futurs à court terme consisteront à diminuer le nombre de prototypes dans le système lors de la phase 2.

Cela peut se faire soit en supprimant les prototypes peu activés, où en fusionnant les prototypes quasi-superposés. A plus long terme, nous envisageons d'explorer d'autres approches pour synthétiser des caractères manuscrits en s'inspirant notamment des techniques proposées par [PLA 06] sur les modèles Delta-lognormal et Sigma-lognormal.

Références

[BOU 03] BOUCHON-MEUNIER B., MARSALA C., Eds., *Logique Floue, Principes, Aide à la Décision*, Hermès-Lavoisier, 2003.

[CAR 92] CARPENTER G., GROSSBERG S., MARKUZON N., REYNOLDS J., ROSEN D., Fuzzy ARTMAP : A neural network architecture for incremental supervised learning of analog multidimensional maps, *IEEE Transactions on Neural Networks*, vol. 3, 1992.

[GAR 01] GARY G. YEN P. M., An effective neuro-fuzzy paradigm for machinery condition health monitoring, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 31-4, 2001, pp. 523 - 536.

[KRI 93] KRISHNAPURAM R., KELLER J., A possibilistic approach to clustering, *IEEE Trans. on Fuzzy Systems*, vol. 1, n° 2, 1993, pp. 98-110.

[MOU 06a] MOUCHERE H., ANQUETI E., A Unified Strategy to Deal with Different Natures of Reject, *18th ICPR*, 2006.

[MOU 06b] MOUCHERE H., ANQUETIL E., Synthèse de caractères manuscrits en-ligne pour la reconnaissance de l'écriture, *Actes du Colloque International Francophone sur l'Ecrit et le Document (CIFED'06)*, 2006, pp. 187-192.

[MOU 07] MOUCHERE H., ANQUETIL E., RAGOT N., Online writer adaptation for handwriting recognition using fuzzy inference systems, *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*, vol. 21(1), 2007, pp. 99-116.

[OUD 04] OUDOT L., PREVOST L., MOISES A., MILGRAM M., Self-Supervised Writer Adaptation using Perceptive Concepts : Application to On-Line Text Recognition, *17th ICPR*, vol. 2, 2004, pp. 598-601.

[PLA 06] PLAMONDON R., DJIOUA M., A multi-level representation paradigm for handwriting stroke generation, *Human Movement Science*, vol. 25, 2006, pp. 586-607.

[POL 01] POLIKAR R., UDPA L., UDPA S., HONAVAR V., Learn++ : An Incremental Learning Algorithm for Supervised Neural Networks, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 31, 2001, pp. 497-508.