Un classifieur hybride pour la reconnaissance d'expressions mathématiques manuscrites en-ligne

Ahmad-Montaser Awal Harold Mouchère Christian Viard-Gaudin

IRCCyN/IVC – UMR CNRS 6597
Ecole polytechnique de l'université de Nantes

Rue Christian Pauc – BP 50609 – 44306 Nantes CEDEX 3 – France
{ahmad-montaser.awal, harold.mouchere, christian.viard-gaudin }@univ-nantes.fr

Résumé

Dans cet article nous proposons une architecture hybride combinant deux classifieurs pour gérer les hypothèses de segmentation/reconnaissance dans un système de reconnaissance d'expressions mathématiques manuscrites en-lignes. Un premier classifieur est chargé de discriminer les bonnes hypothèses de segmentation des mauvaises. Le second classifieur, lui est spécialisé pour reconnaître les formes admissibles. Cette architecture est comparée avec un classifieur unique intégrant explicitement une classe additionnelle correspondant au cas de mauvaise segmentation. L'apprentissage du classifieur de rejet est réalisé en-ligne à partir d'expressions manuscrites complètes. Ce système est entraîné et testé sur une large base synthétisée d'expressions, mais aussi avec une base d'expressions réelles complexes.

Mots clefs

Formules mathématiques, écriture manuscrite, classification, segmentation, analyse structurelle, langages bidimensionnels.

Abstract

In this paper we propose a hybrid architecture that combines two classifiers to control the segmentation/recognition hypotheses for the recognition of online handwritten mathematical expressions. A first classifier, termed as junk classifier, is in charge of distinguishing between correct and wrong hypotheses of segmentation. The second classifier is specialized to recognize acceptable forms. This architecture is compared with a unique classifier integrating explicitly an additional class for wrong segmentations. The training of the junk classifier is done online from complete handwritten expressions. This system is trained and tested with large dataset of synthetic expressions, but also with a set of real complex expressions.

Keywords

Mathematic expressions, handwriting, recognition, segmentation, structural analysis, bi-dimensional languages.

1 Introduction

Récemment les systèmes de reconnaissance de textes manuscrits ont réalisé des progrès significatifs, grâce à l'évolution des outils de segmentation, de classification et à la prise en compte de modèles de langage. Ces mêmes systèmes sont moins puissants quand il s'agit de traiter des langages visuels en deux dimensions (2D).

Les expressions mathématiques [1], schémas, diagrammes, etc. sont des exemples de tels langages bidimensionnels. Dans ces cas, on cherche à résoudre les mêmes problématiques de segmentation, de reconnaissance, et d'interprétation mais dans un espace 2D.

Les mathématiques sont utilisées dans la plupart des domaines scientifiques, comme la physique, l'ingénierie, la médecine, l'économie, etc. En conséquence, une méthode efficace pour saisir des expressions mathématiques dans les documents scientifiques est indispensable.

De nombreux outils sont déjà disponibles pour réaliser cette tâche. Cependant, la plupart de ces outils nécessitent une certaine expertise pour une utilisation efficace. Latex et MathML, par exemple, exigent la connaissance d'un ensemble prédéfini de mots clés pour décrire les symboles mathématiques et les fonctions. D'autres outils, tels que Math Type, dépendent d'un environnement visuel pour ajouter des symboles à l'aide de la souris. Il en résulte un temps de saisie largement supérieur à celui nécessaire pour écrire l'expression sous forme manuscrite.

L'objectif de ces travaux est de proposer une contribution pour la reconnaissance d'expressions mathématiques manuscrites en-lignes. La plupart des recherches émergeantes dans ce domaine considèrent un ensemble de sous-classes des expressions mathématiques et obtiennent alors des résultats prometteurs. Nombre de ces recherches abordent la reconnaissance d'expressions mathématiques en tant qu'une suite de sous tâches indépendantes. En conséquence, les erreurs vont se propager d'une étape à l'autre.

Nous avons proposé [20] une architecture effectuant une optimisation simultanée de la segmentation, la reconnaissance et l'interprétation des expressions mathématiques. Dans ce cadre, le classifieur utilisé pour la reconnaissance de symboles de base est appris au sein du système global de reconnaissance des équations. Cet apprentissage global permet d'apprendre les symboles au

cours de la segmentation au lieu d'utiliser un classifieur simplement appris sur une base de symboles isolés.

Notre contribution dans cet article, est de proposer une architecture plus adaptée pour traiter les hypothèses de bonnes et mauvaises segmentations grâce à l'utilisation d'un classifieur hybride. Ce classifieur consiste en deux modules dont l'un est également appris au sein de l'architecture globale. Il a comme objectif de réduire la complexité du classifieur en essayant d'améliorer la précision de reconnaissance.

Dans la suite, nous introduisons le problème de la reconnaissance d'expressions mathématiques. Puis nous expliquons l'architecture proposée et pour finir nous présentons quelques expérimentations et résultats. Nous comparons nos résultats avec ceux obtenus dans [19].

2 La reconnaissance d'expressions mathématiques

Généralement, de nombreux outils –comme LATEX ou MathML- peuvent être utilisés pour saisir une expression mathématique dans un document numérique. Cependant, cette méthode devient longue et fastidieuse surtout quand il s'agit d'expressions structurellement complexes. L'utilisation d'un stylo numérique, permettant une saisie manuscrite de ces expressions mathématiques, constitue une alternative rendant cette tâche beaucoup plus simple et efficace.

Une expression mathématique se présente sous la forme d'un arrangement de symboles mathématiques en deux dimensions (2D). Cette disposition des symboles est contrôlée par une grammaire elle-même en 2D. Une autre différence avec la reconnaissance de textes manuscrits tient au nombre de symboles. Il est dans ce cas beaucoup plus élevé. En effet, plus de 220 symboles sont nécessaires pour couvrir correctement la plupart des applications scientifiques. En conséquence, la tâche du classifieur de symboles sera beaucoup plus compliquée.

La figure 1 montre une autre différence avec du simple texte. Plusieurs symboles mathématiques sont ambiguës par nature. Cette ambiguïté ne peut être levée que grâce au contexte global. Par exemple, une barre horizontale peut représenter une barre de fraction, le signe moins, une composante du signe plus-ou-moins, ou même du symbole égal.

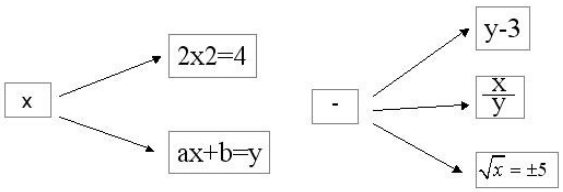

Figure 1 Exemple d'ambiguïté de symboles mathématiques

De plus, les relations spatiales dans une expression ont une nature floue. Ces relations sont également relatives au contexte global. La Figure 2 montre que la relation entre b et c n'est résolu qu'au niveau global [13].

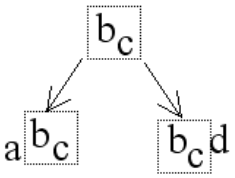

Figure 2 Ambiguïté locale [13]

D'une façon classique, la reconnaissance d'expressions mathématiques se déroule en trois étapes principales [2] : la segmentation de symboles, la reconnaissance de symboles, et l'interprétation.

Nous appellerons *trait* un tracé réalisé entre un poser et un lever de stylo. Le *trait* est l'unité de base et nous supposons que ce *trait* n'appartient qu'a un seul symbole. Cette hypothèse est peu contraignante et largement admise.

Par contre, on considère que chacun des symboles consiste en **un** ou **plusieurs** traits, **réalisés séquentiellement ou non**. Cette dernière hypothèse est très peu souvent supportée par les systèmes actuels. Elle complique singulièrement la segmentation car alors, un graphe classique de segmentation basé sur l'ordonnancement temporel n'est plus suffisant.

L'étape de segmentation vise donc à regrouper les traits qui appartiennent au même symbole. La Figure 3 montre plusieurs groupements possibles d'un ensemble de traits. On voit qu'il s'agit d'un problème complexe et qui devient de plus en plus difficile quand des traits retardés sont présents. Le nombre de segmentations possibles est défini par le nombre de Bell. Sur le simple exemple de la Figure 3, en considérant la présence de 7 traits, on obtient $B_7 = 877$ segmentations distinctes.

Les nombres de Bell satisfont la formule de récurrence :

$$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k \quad \text{où} \quad \binom{n}{k} \quad \text{est un coefficient binomial.}$$

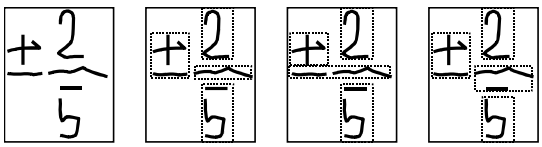

Figure 3 Exemples de segmentations possibles

De nombreuses méthodes ont été explorées pour effectuer la segmentation des symboles. Les projections alternées des traits sur les axes X Y [3] ou des boîtes englobantes [4] des traits peuvent être utilisées pour construire un arbre des relations spatiales. D'autre part, la segmentation peut être guidée par la reconnaissance des symboles ; les traits qui ont la plus forte probabilité d'être un symbole sont regroupés ensemble [5, 19, 20].

Pour l'étape de classification des symboles, les méthodes classiques de reconnaissance de formes sont utilisées dans les systèmes de reconnaissance d'expressions mathématiques. Des méthodes de filtrage par motif sont utilisées dans quelques systèmes et

donnent de bons résultats [7, 19]. Mais ces méthodes restent gourmandes au niveau du temps d'exécution et de stockage. D'autres systèmes [2, 8] utilisent des primitives structurelles et les comparent avec celles de la base d'apprentissage. Les réseaux de neurones ont également obtenus des bons taux de reconnaissance [9, 10]. En plus, ils sont plus rapides et plus légers. Ils représentent donc une solution très intéressante pour la reconnaissance de symboles. On peut également obtenir simultanément une segmentation et une reconnaissance en utilisant des méthodes statistiques comme les Modèles de Markov Cachés [11].

En suite, et avant d'appliquer l'analyse syntaxique, il faut trouver la description structurelle de l'expression [12]. Cette description définit les relations entre les symboles. Mais comme nous l'avons souligné, certaines de ces relations sont ambiguës.

Finalement, l'analyse syntaxique vise à résoudre ces ambiguïtés et à trouver la structure de l'expression. A cause de la nature bidimensionnelle des expressions mathématiques, cette analyse est basée sur une grammaire 2D [14]. La complexité est réduite en utilisant des techniques et méthodes spéciales basées sur des algorithmes géométriques [15].

Dans l'approche proposée nous essayons de compenser les défauts de chaque étape en évaluant conjointement les résultats de la segmentation, de la reconnaissance et leurs interprétations.

3 Le système de reconnaissance

Un premier système de reconnaissance a été proposé [20]. Dans cette première version, un classifieur unique gérait d'une part la reconnaissance des symboles et d'autre part était capable de détecter les formes ne correspondant pas à de vrais symboles, et cela à cause des problèmes de segmentation. En effet, beaucoup d'hypothèses de segmentation sont invalides, elles correspondent soit à des sous-ensembles de symboles, lorsqu'un symbole est écrit en plusieurs traits, soit à des sur-ensemble de symboles lorsque le regroupement considéré de traits fusionne plusieurs symboles. Pour cela, une classe explicite était ajoutée au classifieur. Dans ce cas, le classifieur était un réseau de neurones dont l'apprentissage était réalisé par une méthode d'apprentissage globale incorporée au système complet de segmentation et d'interprétation. Les résultats de cette méthode seront qualifiés dans les tableaux présentés à la section 5 sous le nom « Global ».

L'originalité de l'approche présentée dans cet article est de proposer une architecture utilisant un classifieur hybride, cf. figure 5. Il est constitué de deux classifieurs remplissant des fonctions complémentaires. Un premier classifieur, à deux classes, se contente de discriminer les vrais symboles des fausses hypothèses de segmentation. Les exemples provenant des deux classes seront fournis lors d'un apprentissage global en situation de segmentation et d'interprétation où les résultats fournis par le second classifieur seront mis en jeu. Ce second classifieur est lui plus conventionnellement un classifieur de symboles entraînés hors-ligne à partir d'une base de symboles isolés.

Les scores des deux classifieurs sont ensuite combinés afin d'obtenir le score de reconnaissance d'une hypothèse de symbole. Cette architecture hybride permet d'utiliser des classifieurs qui sont difficiles à apprendre en apprentissage incrémental. Par exemple, les séparateurs à vaste marge (SVM) ne se prêtent pas aisément à un apprentissage incrémental, mais sont intéressant pour être utilisés comme classifieur de symboles isolés. En revanche, les réseaux de neurones (RN) peuvent facilement être mis à jour en utilisant l'algorithme de descente du gradient, et cela dans le cadre d'un apprentissage en-ligne global. Nous présenterons les résultats de ce système sous le nom « hybride ».

De plus, nous comparons ces deux méthodes avec une troisième méthode de base. Dans ce dernier cas, seul le classifieur de symboles isolés est utilisé, il n'y a pas de connaissance explicite de la classe rejet. Ensuite, il est simplement utilisé dans le système. Ce système se trouve sous le nom « isolé » dans les résultats.

Nous comparons également nos résultats avec ceux du système proposé dans [19]. Ce dernier système effectue simultanément la segmentation et la reconnaissance, mais en utilisant un classifieur appris en isolé, soit une architecture proche de la troisième méthode décrite ci-dessus.

3.1 Architecture globale

Une expression donnée est un ensemble de traits représentant des symboles. Reconnaître une expression consiste donc à trouver le meilleur regroupement possible de ses traits, en reconnaître les symboles, les relations entre ces symboles et l'interprétation finale de l'expression.

L'apprentissage du système et la reconnaissance d'expressions mathématiques se font en utilisant la même architecture globale de reconnaisseur d'expressions, voir Figure 4. Il consiste en :

- Un générateur d'hypothèses de symboles élaborant des combinaisons des traits. Un groupe de traits s'appelle une hypothèse de symbole (HS).
- Un classifieur de symboles associant un score de reconnaissance, mais aussi une étiquette pour chacune des hypothèses. Il peut être le classifieur hybride ou un classifieur unique avec ou sans classe rejet.
- Un analyseur de structure permettant d'obtenir des informations structurelles sur chaque hypothèse afin d'établir un coût structurel.
- Un modèle de langage défini par la grammaire de production d'expressions mathématiques.
- Un bloc de décision choisissant l'ensemble des hypothèses minimisant le coût global et respectant le modèle de langage.

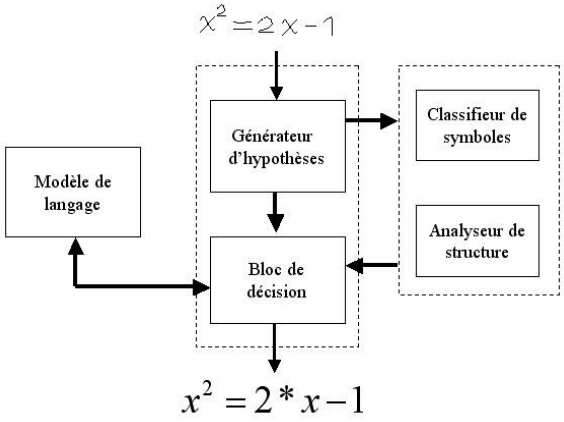

Figure 4 Le reconnaisseur d'expressions

Du point de vue du calcul, la programmation dynamique (DP) [16] est bien adaptée à ce type de problème. Néanmoins, un algorithme classique de DP ne considère que des regroupements de traits consécutifs, et donc nous adoptons une extension de cet algorithme en deux dimensions permettant de gérer des regroupements de traits non-consécutifs [20].

3.2 Classifieur hybride de symboles

Au lieu d'ajouter une sortie additionnelle pour le rejet [20], le classifieur est découpé en deux classifieurs spécialisés, voir Figure 5. Il est constitué d'un classifieur cible destiné à reconnaître les symboles valides. Celui-ci est appris soit en isolé, lorsque le classifieur ne supporte pas un apprentissage incrémental –cas du SVM par exemple-, soit il peut être également appris globalement dans l'architecture globale du système –cas du RN-. Ce classifieur va donc participer à l'apprentissage global du deuxième classifieur, dit classifieur de rejet.

En ce qui le concerne, le classifieur rejet a comme objectif la reconnaissance de deux classes, rejet et non-rejet (symboles valides). Ce classifieur est appris globalement puis que les exemples du rejet sont obtenus au cours de l'étape de segmentation d'expression.

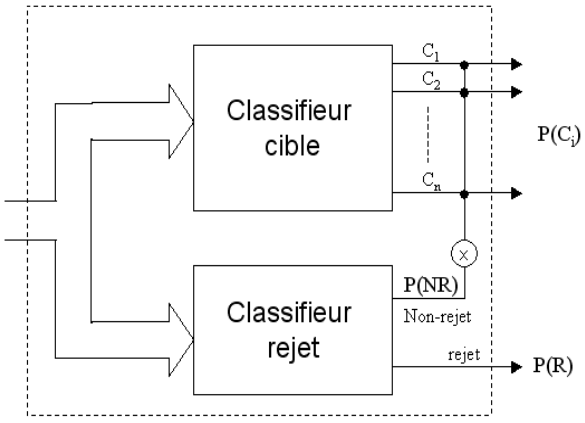

Figure 5 Le classifieur hybride

Nous avons choisi un perceptron multicouches (PMC) comme classifieur de rejet. Pour entraîner ce classifieur, nous utilisons un algorithme de rétro propagation basé sur la descente du gradient. Il prend en compte la vérité terrain d'une expression donnée (la segmentation idéale et l'étiquette de chaque symbole) et la meilleure interprétation obtenue par le système dans son état actuel.

Prenons l'exemple de la Figure 6 : l'expression « x – 1 » soit segmentée et reconnue comme « x1 » à cause d'une mauvaise segmentation regroupant les traits du 'x' et du '–'. Cette mauvaise segmentation entraîner trois mises à jour du classifieur de rejet. Premièrement, l'ensemble de traits correspondant à la mauvaise segmentation est appris en tant que classe rejet. Puis, les traits du 'x' sont appris en tant que non-rejet (vrai symbole), de même pour le trait du '–'. Ce processus est répété sur toute la base d'apprentissage d'expressions complètes jusqu'à convergence de l'apprentissage du classifieur de rejet.

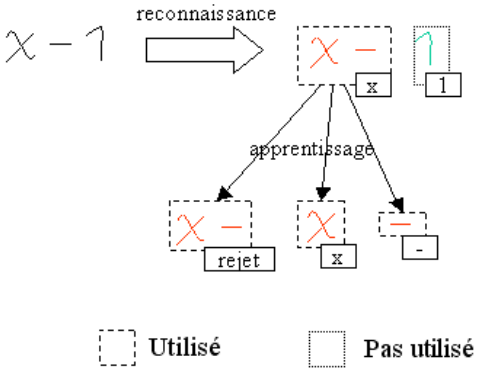

Figure 6 Illustration de l'apprentissage du classifieur « rejet »

3.3 Combinaison des scores des classifieurs

Il n'est pas direct de combiner les scores de reconnaissance de ces deux classifieurs. Ils peuvent être de natures différentes et donc ne peuvent pas être directement combinés. Ce problème est surmonté en appliquant une fonction de type « softmax » sur les sorties des classifieurs [23]. Ainsi, les sorties de chaque classifieur sont en forme de probabilités indépendantes.

La probabilité qu'une sortie du classifieur cible soit la classe C_i, sachant que ce n'est pas un rejet (NR) est :

$$p(C_i \mid \mathrm{NR}) \ ; \ \text{avec} \ \sum_{i=1}^{N} p(C_i \mid \mathrm{NR}) = 1 \qquad (1)$$

où N est le nombre de symboles.

Par ailleurs, la sortie du classifieur rejet nous donne la probabilité qu'une hypothèse soit une classe à accepter (NR) ou une classe à rejeter (R):

$$p(NR) + p(R) = 1 \qquad (2)$$

Mais la probabilité qu'une sortie du classifieur hybride soit la classe C_i est :

$$p(C_i) = p(C_i, NR) + p(C_i, R)$$
$$= p(C_i \mid \mathrm{NR}).\mathrm{p(NR)} + p(C_i \mid \mathrm{R}).\mathrm{p(R)} \qquad (3)$$

Mais par définition: $p(C_i \mid \mathrm{R}) = 0$, donc:

$$p(C_i) = p(C_i \mid \text{NR}).\text{p(NR)} \quad (4)$$

Cette probabilité est normalisée par rapport à celle du rejet permettant donc de combiner le rejet avec les classes du classifieur cible, car :

$$\sum_{i=1}^{N} p(C_i) + p(R) = 1 \quad (5)$$

Ces scores de reconnaissance sont ensuite transformés en coûts utilisant la fonction suivante :

$$\text{Coût}_{\text{reco}} = -\log(\text{p}(\text{C}_\text{i})) \quad (6)$$

En plus de ces coûts de reconnaissance, un coût structurel ($\text{Coût}_{\text{struct}}$) représentant la bonne disposition des hypothèses dans le contexte de l'expression est ajouté. Ce coût est calculé à partir des informations spatiales de chaque hypothèse et prend en compte le type de symbole reconnu. Le coût de reconnaissance d'une sous-expression est :

$$\text{Coût}_{\text{sous-exp}} = \text{Coût}_{\text{reco}} + \alpha.\text{Coût}_{\text{struct}} \quad (7)$$

Où le facteur alpha sert de pondération entre le coût de reconnaissance et le coût structurel. Le coût final de la reconnaissance d'une expression est donc la somme des coûts des sous-expressions formant cette expression.

Enfin, la solution doit respecter le modèle de langage et les expressions non-valides ne sont pas acceptées dans les solutions. Dans la section suivante, nous présentons les résultats expérimentaux obtenus sur notre base d'expressions mathématiques.

4 La base d'expressions mathématiques

Une base de symboles mathématiques manuscrits en-lignes a été collectée auprès de 280 scripteurs. Cette base sert à l'apprentissage du classifieur cible. De plus, la même base de symboles est utilisée afin de générer une base d'expressions manuscrites pseudo-synthétiques. Pour accomplir cette tâche, nous avons utilisé un outil « Latex2Ink » [18] développé par notre groupe de recherche. L'objectif de cette base synthétique est de disposer d'une grande quantité d'expressions pour l'apprentissage et aussi le réglage du système. En complément, une base de vraies expressions a été collectée pour élargir les conditions d'évaluation de ce système.

La difficulté de l'apprentissage est liée à la complexité des expressions et au nombre possible de symboles. Nous proposons deux scénarios d'évaluation correspondant à deux corpus d'expressions mathématiques bien distincts. Le premier « Calculette », présente une base d'expressions simples et permet donc de bien configurer les méta-paramètres du système. Le second est tiré de la base « Aster » [17], elle présente une grande diversité d'expressions, plus complexes permettant donc de bien tester le système avec des expressions très réalistes.

4.1 La base Calculette

Les expressions de cette base sont réduites à une simple application de calculatrice. Le nombre de classes de symboles est 15. Ces symboles sont les chiffres [0-9], les simples opérations mathématiques [+, -, *, ÷] et le signe égal [=]. Le tableau 1 montre la constitution de la base d'apprentissage et de test pour les symboles isolés et les expressions de ce corpus.

Tableau 1 La constitution de la base Calculette

	# scripteurs	# Symboles isolés	# Expressions	# Symboles
Apprentissage	180	180 × 15 = 2700	180 × 5 = 900	5448
Test	100	100 × 15 = 1500	100 × 5 = 500	3051

Nous avons généré 1 400 expressions dont 900 expressions sont utilisées pour l'apprentissage global du système, les 500 expressions restantes sont utilisées pour le test. La Figure 7 montre un exemple d'expression générée pour cette base. Cette expression a été aléatoirement générée à partir du modèle « [1-999] [+, –, ×, ÷] [1-999]= [1-999]». Chaque symbole subit une déformation et un positionnement s'appuyant sur un processus stochastique.

Figure 7 Un exemple d'une expression synthétique de la base Calculette

4.2 La base Aster

La base Aster [17] comporte 62 expressions différentes couvrant une majorité des domaines mathématiques. Une expression contient 13 symboles en moyenne. Le nombre total de symboles est de 839 répartis dans 48 classes distinctes : chiffres, lettres latines, lettres grecques, opérations binaires, symboles et des fonctions élastiques.

En raison de la grammaire que nous avons implémentée, nous avons considéré un sous-ensemble de 36 expressions de la base Aster couvrant une bonne partie des disciplines communes des mathématiques, voir le tableau 2. Pour ce sous-ensemble le nombre des classes de symboles est réduit à 34 symboles.

Tableau 2 Le corpus d'expressions extraites de la base Aster

Domaines	# Expressions
Fractions	8
Exemples de Knuth	6
Fraction continue	1
Expressions algébriques	3
Racines	2
Trigonométries	6
Logarithmes	3
Séries	3
Intégrales	1

Sommes Fonctions hyperboliques	2 1
Total	36

Le tableau 3 montre la constitution de la base d'apprentissage et la base de test des expressions. La Figure 8 montre quelques exemples d'expressions générées à partir de la base Aster.

De plus, afin de tester le système avec une base de vraies expressions mathématiques manuscrites en-lignes, chacune de ces 36 expressions a été écrite deux fois, dix scripteurs ayant participés à cette base. Donc, nous avons obtenu un total de 72 vraies expressions, voir le tableau 3.

Tableau 3 Constitution de la base Aster

	# scripteurs	# symboles isolés	# Expressions	# Symboles
Apprentissage	180	180 × 34 = 6120	180 × 36 = 6480	180 × 412 = 74160
Test (synthétique)	100	100 × 34 =3400	100 × 36 = 3600	100 × 412= 41200
Test (naturelle)	10	-	2 × 36 = 72	2 × 412 = 824

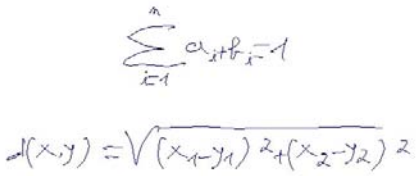

Figure 8 Exemples d'expressions synthétiques de la base Aster

5 Expérimentations et résultats

L'évaluation du système au niveau expressions complètes est trop globale pour être significative, et cela dans la mesure où une expression peut être très longue. En conséquence, il est important de connaître les performances aux niveaux intermédiaires. Nous avons donc choisi trois mesures similaires à celles utilisées récemment dans [6, 19] comparer nos résultats, celles-ci sont présentées à la section 5.2. Toutefois, il est bon de rappeler que les bases d'expressions sont différentes, et donc que la validité des comparaisons des résultats qui sont présentés au tableau 6 est limitée.

Les classifieurs cibles sont appris avec les bases de symboles isolés. Nous avons testé trois classifieurs différents pour le classifieur cible. D'abord, nous avons choisi un perceptron multicouches (PMC), avec une seule couche cachée utilisant 100 neurones. Puis, nous utilisons un réseau de neurones à convolution [22] (TDNN), intéressant pour ces propriétés d'être insensible aux changements de position. Le dernier classifieur utilisé est un séparateur à vaste marge [21] (SVM) utilisant un noyau Gaussien. En ce qui concerne la structure du classifieur rejet, nous utilisons soit un PMC avec seulement 50 neurones dans la couche cachée, soit un TDNN pour le cas où le classifieur cible est aussi un TDNN. D'autres types de classifieurs de symboles peuvent être utilisés puisque celui-ci est appris indépendamment du reste du système dans le cadre de cette architecture hybride.

5.1 Performances du classifieur de symboles isolés

Les classifieurs cibles sont appris à partir des bases de symboles isolés. Leurs performances sur les bases de tests isolés sont présentées dans le tableau 4.

Tableau 4 Performances des classifieurs isolés

Classifieur	Calculette Test % (15 classes)	Aster Test % (34 classes)
PMC	96.6	95.4
TDNN	96.8	95.7
SVM	96.7	96.2

Ces classifieurs appris en isolé sont ensuite intégrés dans l'architecture globale du système. Le rôle de ces classifieurs est de guider le processus d'apprentissage du deuxième classifieur, le classifieur de rejet. Nous avons remarqué que la tendance du système est similaire quand on utilise un PMC ou un TDNN comme classifieur cible. Dans la section suivante nous allons nous limiter à présenter et comparer les résultats d'utilisant un TDNN ou un SVM comme classifieur cible.

5.2 La performance de la reconnaissance d'expressions

Les performances du système ont été évaluées avec ces trois mesures:

- Taux de bonne segmentation, SegRate = #symboles bien segmentés/#total de symboles
- Taux de bonne reconnaissance, RecRate = #symboles bien reconnus//#total de symboles
- Taux de bonne reconnaissance d'expressions, ExpRate = #expressions bien reconnues/#total d'expressions.

Une expression est bien reconnue si tous ses symboles sont bien reconnus et bien interprétés. Ainsi si l'expression *« 2^x »* est reconnue comme *« 2x »* , bien que tous ses symboles soient bien reconnus, l'expression est considérée mal reconnue.

Le tableau 5 montre les résultats obtenus sur la base de la Calculette. Les résultats de la base Aster sont montrés au tableau 6.

Tableau 5 Taux de reconnaissance de la base test Calculette

		SegRate%	RecoRate%	ExpRate%
TDNN	a) Isolé	96.6	94.3	81.1
	b) Global	99.2	96.2	84.2
	c) Hybride	99.2	97	87.3
SVM	a) Isolé	88.6	87.5	82.2
	c) Hybride	99.6	97.6	89.2

Ces résultats permettent de comparer le comportement du classifieur hybride c), avec un classifieur appris en isolé a) sans capacité explicite de traiter le rejet, et un autre classifieur appris globalement avec apprentissage explicite de la classe rejet b) [20]. Nous pouvons vérifier que les résultats de segmentation (SegRate) dépendent bien du reconnaisseur utilisé car la segmentation n'est pas faite en amont indépendamment du reconnaisseur, mais elle est bien couplée à celui-ci. Le tableau 5 montre que l'architecture hybride apporte un progrès significatif surtout au niveau taux de reconnaissance d'expressions. Le meilleur score est atteint en associant le classifieur SVM avec un taux de reconnaissance au niveau expression de 89.2 %. Cette observation montre que le classifieur hybride est le plus adapté au problème de reconnaissance d'expressions de la base Calculette.

Table 6 Taux de reconnaissance de la base Aster

Expressions synthétiques		SegRate%	RecoRate%	ExpRate%
TDNN	a) Isolé	64.2	62.9	25.6
	b) Global	86.9	84.6	61.8
	c) Hybride	89.8	87.4	51.8
SVM	a) Isolé	74.6	73.5	45.5
	c) Hybride	73.6	72.9	50.6
Expressions naturelles		SegRate%	RecoRate%	ExpRate%
Autre travail [19]		94.8	84.8	29.2
TDNN	a) Isolé	50	46.6	11.4
	b) Global	78.7	72.5	27.1
	c) Hybride	79.4	74.6	30
SVM	a) Isolé	67	63.7	28.6
	c) Hybride	67.6	64	27.1

Le Tableau 6 montre que le TDNN et le SVM ont des comportements différents quand on passe du classifieur isolé a) au hybride c). En effet, la progression est plus spectaculaire avec le TDNN dont le taux de reconnaissance d'expressions augmente de 25.6% à 51.8% sur la base synthétique et de 11.4% à 30% sur la base de vraies expressions. A la différence de la base calculette, ici c'est l'hybridation avec le TDNN qui donne le meilleur résultat. On met en exergue ici une propriété intéressante du TDNN, à savoir sa capacité à reconnaître des formes dans un environnement de taille variable, et cela mieux que ne le fait le SVM quand l'environnement est complexe.

En considérant le TDNN et en comparant le système hybride c) avec le classifieur unique appris globalement b), nous constatons que celui-ci se comporte un peu mieux sur la base synthétique que le système hybride. Par contre, le classifieur hybride est meilleur sur la base des expressions naturelles.

A l'inverse quand on compare le SVM isolé et son intégration dans l'hybride, le comportement est différent entre la base synthétique et la base réelle. Il y a une amélioration sur la base synthétique (45.5% à 50.6%), et une baisse de performances (28.6% à 27.1%) sur la base des vraies expressions. Nous pensons qu'il conviendrait d'avoir une approche plus rigoureuse pour l'application de la fonction softmax aux sorties du SVM afin de pouvoir effectivement interprétée les sorties en tant que probabilités.

Pour résumer, on peut dire que l'ajout d'une classe additionnelle pour modéliser les mauvaises segmentations n'est pas une solution toujours bénéfique. Cela dépend des propriétés de modélisation intrinsèque du classifieur. Ainsi, avec un TDNN en choisissant de rajouter une classe de rejet soit apprise en même temps que les autres (b), soit apprise après les autres dans le système hybride proposé ici (c) les performances du système sont améliorées vis-à-vis du système de base (a) sans classe de rejet. Par contre avec un SVM, l'apport de la classe supplémentaire dans le système hybride (c) n'apporte pas grand chose, et même détériore légèrement les performances sur la base des expressions naturelles. On peut en déduire que le SVM a une capacité de modélisation intrinsèque supérieure au TDNN ce qui lui permet de mieux gérer les situations non apprises.

6 Conclusion et perspective

Dans cet article, nous avons proposé un nouveau classifieur hybride pour la reconnaissance d'expressions mathématiques manuscrites en-lignes. Le système est appris en utilisant une base d'expressions synthétiques. Ensuite, il est testé non seulement avec des expressions synthétiques, mais aussi avec des vraies expressions.

L'association d'un classifieur de type TDNN appris sur une base de symboles isolés et d'un second classifieur permettant de gérer le rejet qui lui est entraîné au sein du système global donne les meilleures performances sur la base de test des expressions mathématiques naturelles. On obtient ainsi près de 80% de taux de bonne segmentation, près de 75% de symboles reconnus, ce qui permet de reconnaître 30% de ces expressions complexes.

Bibliographie

[1] Blostein D., A.G., Recognition of mathematical notation, dans *Handbook on Optical Character Recognition and Document Image Analysis*, Q.s.U., 1997, World Scientific Publishing Company: Kingston, Ontario, Canada. p. 557-582.

[2] Chan K-, D.-Y.Y., An efficient syntactic approach to structural analysis of on-line handwritten mathematical expressions, *Pattern Recognition*, 2000. 33: p. 375 - 384.

[3] C. Faure, Z.X.W, Automatic perception of the structure of handwritten mathematical expressions, dans *Computer Processing of Handwriting*, 1990, World scientific, Singapore.

[4] J. Ha, R.M.H., et I. T. Phillips, Understanding mathematical expressions from document images, 3e *International Conference on Document Analysis and Recognition*, 1995, p. 956-959.

[5] Steve Smithies, K.N., James Arvo. A Handwriting-Based Equation Editor, the Graphics Interface, 1999, Kingston, Ontario, Canada.

[6] Yamamoto R., S.S., Nishimoto T., Sagayama S., On-Line Recognition of Handwritten Mathematical Expressions Based on Stroke-Based Stochastic Context-Free Grammar, *tenth International Workshop on Frontiers in Handwriting Recognition* 2006, La Baule, France. p. 249 - 254

[7] Nakayama, Y. A prototype pen-input mathematical formula editor, dans *EDMEDIA*, 1993.

[8] A. Belaid, J.-P.H, A syntactic approach for handwritten mathematical formula recognition, dans *Transactions on Pattern Analysis and Machine Intelligence*, 1984.
[9] Marzinkewitsch R., Operating computer algebra systems by handprinted document, *International Symposium on Symbolic and Algebraic Computation*, 1991.
[10] Yannis A. Dimitriadis, J.L.C., Towards an ART based mathematical editor that uses online handwritten symbol recognition, *Pattern Recognition*, 1995. 28(6): p. 807 822.
[11] Lehmberg S., H.-J.W., Lang M., A Soft-decision approach for symbol segmentaiton within handwritten mathematical expressions, *Int. Conference on Acoustics, Speech, and Signal Processing*, 1996, Atlanta, USA.
[12] Fukuda R., S.I., Tamari F., Xie M., et Suzuki M., A technique of mathematical expression structure analysis for the handwriting input system, *fifth International Conference on Document Analysis and Recognition* , 1999. p. 131 - 134.
[13] Martin, W. Computer input/output of mathematical expressions, dans *Symbolic and Algebraic Manipulations*, 1971, New York.
[14] Prusa D., V.H. 2D Context-Free Grammars: Mathematical Formulae Recognition. dans *The Prague Stringology Conference*, 2006, Prague.
[15] Liang P., M.N., Shilman M., et Paul Viola. Efficient Geometric Algorithms for Parsing in Two Dimensions, *eighth International Conference on Document Analysis and Recognition*, 2005, Seoul, South Korea. p. 1172 - 1177.
[16] Held M., R.M.K., The construction of discrete dynamic programming algorithms, *IBM Syst. J.* 4 (2), 1965: p. 136-147.
[17] Raman, T.V., Audio system for technical readings, 1994, Cornell University.
[18] Awal A.M., Cousseau R., Viard-Gaudin C. Convertisseur d'équations LATEX2Ink,. *dans Colloque International Francophone sur l'Ecrit et le Document* 2008, Rouen, France, . p. 193 – 194.
[19] Rhee T-K., K.-E.K., Kim J., Robust Recognition of Handwritten Mathematical Expressions Using Search-based Structure Analysis, 11th *International Conference on Frontiers dans Handwriting Recognition*, 2008, Montreal. p. 19 - 24.
[20] Awal A.M, Mouchère H., Viard-Gaudin C., Towards handwritten mathematical expression recognition, *tenth International Conference on Document Analysis and Recognition*, 2009, Barcelona, Spain. P. 1046 – 1050.
[21] Cortes C. et Vapnik V., Support-vector networks. *Machine Learning* , 20(3) :273-297, 1995.
[22] Schenkel M., Guyon I. et Henderson D., Online cursive script recognition using time delay neural networks and hidden Markov models, Mach. Vis. Appl., *Special Issue on Cursive Script Recognition* 8 (1995) 215–223.
[23] C.M. Bishop, "Neural Networks for Pattern Recognition", *Oxford University Press*. ISBN 0-19-853849-9, pages, 1995