Un simulateur de classifieur pour évaluer les méthodes de combinaison

A classifier simulator for evaluating combination methods

H. Zouari[1, 2], L. Heutte[1], Y. Lecourtier[1], A. Alimi[2]

[1] Laboratoire Perception, Systèmes, Information (PSI), Université de Rouen, France
[2] Groupe de Recherche sur les Machines Intelligentes (REGIM), Université de Sfax, Tunisie

Laboratoire PSI, UFR des Sciences, Université de Rouen, F-76821 Mont-Saint-Aignan Cedex, France
Hela.Khoufi@univ-rouen.fr

Résumé

L'utilisation de données artificielles générées par un simulateur de classifieur est devenue récemment un moyen essentiel pour analyser expérimentalement le comportement des méthodes de combinaison parallèle de classifieurs.

Dans cet article, nous proposons une nouvelle méthode de simulation de classifieur permettant de générer des sorties (listes de propositions) en fonction de performances désirées (taux moyen de reconnaissance, taux moyen de rejet, ...) pour un problème de classification quelconque. La méthode proposée consiste tout d'abord à construire des matrices de confusion afin de fixer le comportement intrinsèque du classifieur à partir d'un jeu réduit de paramètres puis à générer la liste des sorties à partir de ces matrices. Nous présentons les algorithmes sur lesquels reposent la méthode. Nous montrons sur un exemple comment ce simulateur de classifieur peut être utilisé pour étudier le comportement de trois méthodes de combinaison de type rang.

Mots Clef

Simulateur de classifieur, génération des sorties de classifieur, performance de classifieur, matrice de confusion, évaluation des méthodes de combinaison.

Abstract

The use of artificial data generated by a classifier simulator has become recently an essential tool to analyse experimentally the behaviour of the parallel classifier combination methods.

In this paper, we propose a new method of classifier simulation aimed at generating outputs (lists of propositions) according to some desired performance (mean recognition rate, mean rejection rate, ...) for any classification problem. The proposed method consists firstly in constructing the confusion matrices in order to fix the intrinsic behaviour of the classifier from a reduced set of parameters and next in generating the lists of outputs from these matrices. We present the algorithms on which the method is based. We show on an example how this classifier simulator can be used to study the behaviour of three types of ranking combination methods.

Keywords

Classifier simulator, generation of classifier outputs, classifier performance, confusion matrix, evaluation of the combination methods.

1 Introduction

Récemment, la combinaison parallèle de classifieurs a été proposée comme une voie de recherche permettant d'améliorer les performances (taux de reconnaissance, fiabilité, …) des systèmes de reconnaissance en exploitant la complémentarité qui peut exister entre les classifieurs. A ce propos, plusieurs méthodes ont été développées et intensivement utilisées dans des problèmes de reconnaissance différents. Nous pouvons citer par exemple la reconnaissance du manuscrit [25, 7, 15], la reconnaissance de chiffres manuscrits [8, 10, 5, 11], la reconnaissance de visages [1, 3] et la vérification de signatures [2, 26]. Elles se distinguent principalement par leur capacité d'apprentissage et le type de sortie de classifieurs qu'elles combinent [25, 10, 27]. Les méthodes de type I combinent des classifieurs dont chacun attribue une classe unique à la forme à reconnaître [15, 25]. Parmi ces méthodes, on trouve le vote majoritaire [15, 25, 11, 5, 23], la théorie de Bayes [25], la théorie de Dempster-Shafer [25] et la méthode BKS (Behaviour Knowledge Space) [8]. Les méthodes de type II combinent des listes ordonnées de classes. Le Borda count appartient à cette catégorie [7, 19, 22]. Les méthodes de type III utilisent en plus une confiance associée à chacune des classes des listes proposées par les classifieurs. Elles incluent les réseaux de neurones [8] et les règles fixes comme le maximum, la somme, la moyenne et le produit [11, 5, 12, 21]. Le choix d'une méthode de combinaison reste cependant l'un des problèmes les plus difficiles rencontrés lors de la conception d'un système à plusieurs classifieurs. En effet, les travaux qui traitent ce problème testent généralement un ensemble de méthodes de combinaison pour retenir la plus adéquate en fonction de certains critères de

performance. Cependant, les résultats obtenus restent étroitement dépendants des applications traitées et par conséquent, difficiles à généraliser en dehors d'un contexte applicatif donné. D'autre part, il existe certains travaux intéressants qui traitent le problème de l'évaluation des méthodes de combinaison avec des données réelles en utilisant des bases de données différentes [9, 19, 21, 22]. Toutefois l'utilisation de données réelles ne permet pas d'avoir suffisamment de variabilité dans les performances des classifieurs à combiner pour analyser de façon fiable le comportement de ces méthodes de combinaison [23, 11, 10, 5, 9, 21].

Pour contourner cette difficulté, la simulation de données est devenue récemment un moyen pratique pour générer artificiellement de la variabilité dans les performances des classifieurs à combiner permettant ainsi d'évaluer de façon robuste le comportement des méthodes de combinaison. C'est l'objet de la présente communication dans laquelle nous proposons une nouvelle méthode de simulation de classifieur permettant de générer des sorties (listes de propositions) en fonction de performances désirées (taux moyen de reconnaissance, taux moyen de rejet, …) pour un problème de classification quelconque. La méthode proposée consiste à construire des matrices de confusion afin de fixer le comportement intrinsèque du classifieur à partir d'un jeu réduit de paramètres puis à générer la liste des sorties à partir de ces matrices.

Cet article est organisé comme suit. La section 2 présente une analyse critique de quelques travaux récents qui ont abordé le problème de la simulation de données dans le cadre de l'évaluation des méthodes de combinaison de classifieurs et nous justifions les choix retenus quant aux caractéristiques de notre propre simulateur. Nous présentons dans la section 3 son architecture et les différents paramètres utilisés pour la simulation. La section 4 décrit la construction des matrices de confusion nous permettant de fixer le comportement intrinsèque du classifieur à partir de ces paramètres. La section 5 présente les algorithmes de génération des sorties. A travers un exemple simple de simulation, nous montrons, dans la section 6, comment le simulateur peut être utilisé pour étudier le comportement des méthodes de combinaison. En conclusion, nous discutons du simulateur proposé et présentons quelques perspectives pour l'améliorer.

2 Simulation de données

Dans le cadre de la combinaison de classifieurs, on peut généralement simuler des données à trois niveaux :

- à l'entrée du classifieur : il s'agit d'introduire des modifications sur les données à reconnaître afin de produire une variabilité dans l'espace de caractéristiques utilisé par le classifieur [17]. On peut introduire, par exemple, du bruit sur les signaux ou des transformations (rotation, ajout de pixels) sur les images;
- au niveau de l'espace de caractéristiques : ceci consiste à générer de nouveaux vecteurs de caractéristiques à partir d'un ensemble de vecteurs en injectant par exemple du bruit (selon une moyenne et une variance donnée) sur chacune des caractéristiques [18];
- au niveau des sorties du classifieur : ceci consiste à utiliser un simulateur permettant de générer des données artificielles selon des performances désirées telles que le taux moyen de reconnaissance ou une matrice de confusion (réelle ou simulée) [19].

Parmi ces méthodes de génération de données, la simulation au niveau des sorties est la plus intéressante car elle permet de contrôler directement les entrées des méthodes de combinaison. A partir de là, certains chercheurs ont utilisé cette technique pour évaluer les méthodes de combinaison [18, 13, 16]. Dans [13] par exemple, l'auteur propose une méthode de génération séquentielle de sorties de classifieurs dépendants pour montrer l'importance de la dépendance de classifieurs dans la combinaison par vote majoritaire. En se basant sur la matrice de dépendance entre les classifieurs à simuler et leur taux de reconnaissance, le générateur produit le nombre de sorties désirées pour chaque classifieur. Dans [16], l'auteur génère aussi différents groupes de classifieurs dont chacun se caractérise par un taux de reconnaissance et un niveau de corrélation pour analyser le comportement de trois méthodes de combinaison de type classe. Pour montrer l'importance des méthodes de combinaison de type rang par rapport au type classe et au type mesure, Parker [19] simule des classifieurs basés sur des matrices de confusion construites selon le taux de reconnaissance fixé.

Ces travaux, aussi intéressants soient ils, sont cependant limités parce qu'ils utilisent des générateurs qui fournissent uniquement des sorties de type classe. Les classifieurs simulés ne sont de plus généralement contrôlés que par un seul paramètre comme le taux moyen de reconnaissance sans qu'il soit possible de générer une quelconque variabilité dans les taux de reconnaissance par classe. En d'autres termes, les simulateurs de classifieurs développés jusqu'à maintenant ([18, 13, 16]) sont très loin de générer des comportements proches de la réalité et sont en ce sens limités pour aborder l'évaluation des méthodes de combinaison.

Il convient maintenant de se poser la question de ce qu'on entend par « générer un comportement proche de la réalité ». Il suffit pour cela de revenir à ce que produit un classifieur réel lorsqu'il est amené à se prononcer sur une base de test donnée. Dans le cas le plus général, les sorties peuvent être représentées comme le montre la figure 1 : les listes de solutions peuvent être de taille variable (c'est-à-dire comporter une, deux ou N solutions), chaque solution étant associée à un score (probabilité, distance, confiance, …) ; le classifieur peut également avoir des capacités de rejet.

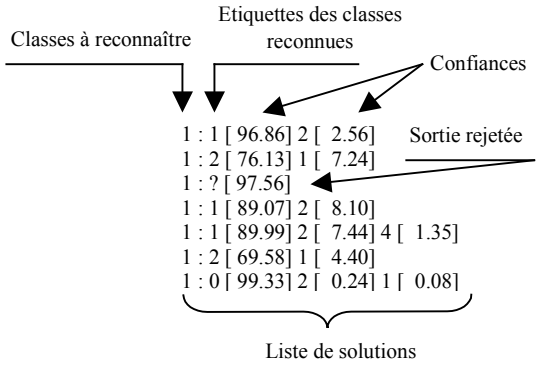


Figure 1 : *Exemples de sorties proposées par un classifieur réel*

Disposant d'une telle base de test, on procède généralement à l'évaluation du comportement (performance) du classifieur à partir de mesures globales comme le taux de reconnaissance ou de bonne classification (TL), le taux de rejet (TR) et/ou le taux de confusion (TC). Mais on peut également évaluer le classifieur en mesurant sa capacité à fournir la bonne solution en première proposition (on utilise alors des mesures du type TL^1, TC^1, TR^1) ou dans les deux premières propositions (TL^2, TC^2, TR^2), et de façon générale dans les K premières propositions (TL^k, TC^k, TR^k), TL^k étant le nombre de sorties dans lesquelles la bonne classe apparaît dans les k premières solutions, TR^k correspondant au nombre de sorties rejetées et TC^k au nombre de sorties dans lesquelles la bonne classe n'apparaît pas dans les k premières solutions. Enfin, une analyse plus précise peut être obtenue par des mesures locales comme les matrices de confusion MC^K (pour K=1 à N) qui décrivent le comportement du classifieur pour chacune des classes :

$$MC^K = \begin{bmatrix} TL_1^K & \cdots & TC_{1j}^K & \cdots & TC_{1N}^K & TR_1 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \\ TC_{i1}^K & \cdots & TL_i^K & \cdots & TC_{iN}^K & TR_i \\ \vdots & \ddots & \vdots & \ddots & \vdots & \\ TC_{N1}^K & \cdots & TC_{Nj}^K & \cdots & TL_N^K & TR_N \end{bmatrix}$$

Notons que MC^K ne peut être formellement appelée "matrice de confusion" que pour K = 1. Pour K entre 2 et N, il serait plus rigoureux de l'appeler "matrice de co-présence". Dans cette matrice, chaque élément TL_i^k correspond au nombre de sorties de la classe i pour lesquelles l'étiquette i apparaît dans les K premières solutions par rapport au nombre total des sorties de la classe i. La moyenne de ces taux donne le taux de reconnaissance global TL^K. La dernière colonne N+1 est réservée aux taux de rejet TR_i qui correspondent au nombre de sorties rejetées pour chaque classe i. La moyenne de ces taux est TR^K. Les éléments restants sont les taux de confusion $TC_{i,j}^k$ qui correspondent au nombre de sorties étiquetées i pour lesquelles la solution j (j≠i) apparaît dans les K premières solutions.

Simuler un classifieur le plus réaliste possible, c'est donc prendre en compte le maximum de ces indicateurs de performances globales et locales comme paramètres d'entrée de la simulation afin de produire artificiellement le type de sorties de la Figure 1 tout en respectant un comportement désiré dans les K premières propositions (au moyen de paramètres du type TL^K, TR^K et TC^K) et non pas seulement en première proposition (i.e. TL^1). D'autre part, le simulateur « idéal » doit pouvoir aussi générer des comportements différents (en reconnaissance et en rejet) pour chacune des classes : il faut donc disposer de paramètres permettant de contrôler la variabilité dans les performances sur chacune des classes, par exemple en limitant la marge de variation des taux de reconnaissance et de rejet par classe. Enfin, si l'on veut pouvoir utiliser le simulateur pour évaluer le comportement des méthodes de combinaison de classifieurs pour un problème de classification quelconque, on doit pouvoir également disposer en entrée du simulateur de paramètres permettant de fixer le nombre de classes du problème ainsi que le nombre de sorties à générer pour chacune des classes.

Nous proposons dans la section suivante un simulateur qui justement présente ces caractéristiques et permet de générer une grande variabilité de comportements à partir d'un jeu réduit de paramètres.

3 Méthode proposée

Dans cette section, nous proposons une nouvelle méthode de génération artificielle de sorties de classifieur de type mesure c'est-à-dire de listes de solutions dont chacune est composée d'une confiance associée à une étiquette. Notons que nous avons choisi de générer ce type de sorties parce qu'il peut être facilement transformé en type rang (en ne tenant compte que des étiquettes) ou en type classe (en ne retenant que la première solution de chaque liste). Pour générer ce type de sorties, le simulateur reçoit en entrée les principaux paramètres suivants :

TL^K : correspond au taux moyen de reconnaissance dans les K premières solutions du classifieur. Il s'agit plus précisément du rapport entre le nombre de sorties dans lesquelles la bonne classe apparaît dans les K premières solutions et le nombre total de sorties.

TR : correspond au taux de rejet qui représente le rapport entre le nombre de sorties rejetées et le nombre total de sorties. Notons ici que nous avons choisi de n'associer en sortie qu'une seule étiquette (rejet) lorque le classifieur rejette si bien que $TR = TR^1 = \ldots = TR^k = \ldots = TR^N$.

α^k : permet de limiter le champs de variation des taux de reconnaissance pour chaque classe TL_i^k qui doivent être tirés aléatoirement dans $[TL^k - \alpha^k ; TL^k + \alpha^k]$.

β : limite la marge des taux de rejet TR_i pour chacune des classes: chaque taux de rejet TR_i doit être tiré dans l'intervalle [TR - β ; TR + β].

De la même façon qu'à partir des sorties d'un classifieur réel, nous pouvons déterminer son comportement pour différentes tailles de la liste de solutions (en première proposition, dans les deux premières, …, dans les K premières), notre simulateur permet de contrôler non seulement le comportement dans les K premières solutions mais aussi, et en même temps, son comportement dans les K' premières (avec K'<K). Ceci nous permet ainsi de générer des listes de solutions à l'intérieur desquelles nous contrôlons la corrélation entre les solutions.

L'algorithme consiste alors à générer d'abord pour chaque sortie les K' solutions à partir de $MC^{K'}$ construite en fonction de $TL^{K'}$, $\alpha^{K'}$ et à ajouter ensuite les K-K' solutions restantes à la liste en utilisant MC^{K}. Cependant, pour construire $MC^{K'}$, tous les taux dans la matrice $MC^{K'}$ doivent être inférieurs à ceux de la matrice MC^{k} ; nous devons donc construire la matrice $MC^{K'}$ à partir de MC^{K} en respectant les contraintes suivantes : $TL^{K'} < TL^{K}$ et $TC^{K'} < TC^{K}$. Par conséquent, la structure du simulateur (figure 2) est constituée des étapes suivantes :

1. Construction de la matrice MC^{K} à partir de N (nombre de classes), K, TL^{K}, α^{K} , TR, β
2. Création de la matrice $MC^{K'}$ à partir de la matrice MC^{K}, $TL^{K'}$ et $\alpha^{K'}$.
3. Pour chaque sortie, génération d'une liste à K' solutions pour chaque sortie en utilisant la matrice $MC^{K'}$
4. Pour chaque sortie, génération des K-K' solutions restantes en utilisant la matrice MC^{K} et affectation des valeurs de confiance à chaque solution.

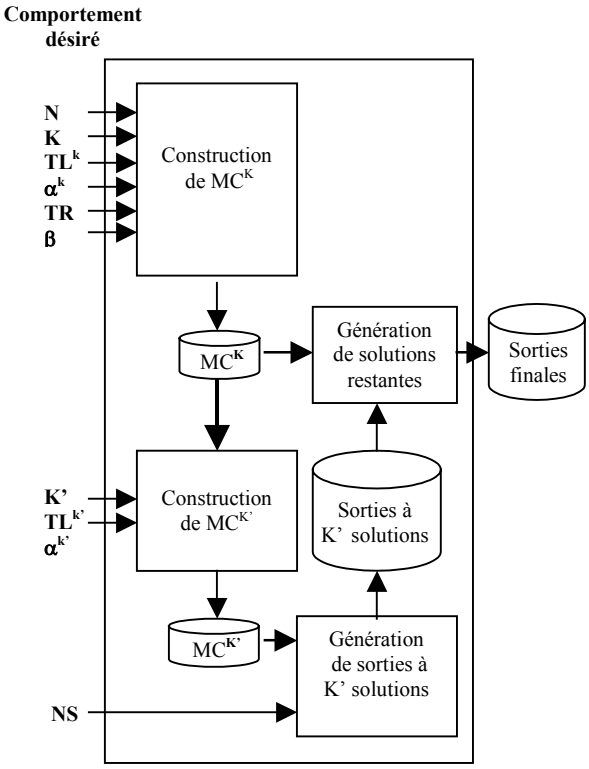


Figure 2 : *Génération de NS sorties en respectant un comportement fixé dans les K' premières solutions et un comportement fixé dans les K premières solutions*

Dans les sections suivantes, nous détaillons ces différentes étapes de simulation en présentant les contraintes à respecter et les algorithmes nécessaires à leur réalisation. Notons que notre simulateur peut fournir aussi une liste de sorties en respectant un seul comportement dans les K premières solutions. Cette génération est plus simple et n'est réalisée qu'en deux étapes. Elle consiste tout d'abord à construire la matrice MC^{K} en utilisant les paramètres considérés (N, K, TL^{K}, α^{K} , TR, β) et à générer par la suite des listes de solutions en utilisant la matrice MC^{K}. Ce mode est plus détaillé dans [28].

4 Construction des matrices de confusion

4.1 Construction de MC^{K}

La construction de la matrice MC^{K} (pour K variant de 1 à N) se fait en trois étapes : génération des taux de rejet pour chacune des classes puis génération des taux de reconnaissance et enfin génération des taux de confusion. Dans la première étape, on tire aléatoirement un taux de rejet par classe suivant la moyenne TR et la variance β. Respecter la moyenne TR revient à tirer aléatoirement des taux TR_i dont la somme est égale à TR*N. Respecter une variance β revient à tirer aléatoirement des taux de rejet entre TR - β et TR + β. Dans la deuxième étape, on

tire les taux de lecture par classe. Ce tirage se fait de la même façon que pour les taux de rejet c'est à dire en respectant une moyenne TL^K et une variance α^K. A chaque tirage, on doit respecter la contrainte suivante : $TL_i^k \leq 100 - TR_i$ pour tout K variant de 2 à N-1. Ayant fixé les taux de rejet et les taux de lecture pour chaque classe (ligne) dans la matrice de confusion, on remplit ensuite les colonnes restantes "confusion". Pour cela, on tire aléatoirement des valeurs comprises entre 0 et la somme des taux de confusion dont la valeur dépend du nombre de propositions K. Dans le cas où K=1, la somme des taux de chaque ligne doit être égale à 100% puisque les sorties à générer par la matrice MC^1 sont formées d'une seule solution. Donc, pour générer les taux de confusion de MC^1, nous devons respecter la contrainte suivante :

$$\sum_{j=1, i\neq j}^{N} TC_{i,j}^1 = 100\% - TR_i - TL_i^1 \qquad (1)$$

Dans le cas où K = N, les taux de confusion $TC_{i,j}^N$ de chaque ligne i de la matrice sont égaux à 100 - TR_i . Ceci est dû au fait que l'utilisation de la matrice MC^N permet d'avoir des sorties composées de N solutions chacune à l'exception des sorties rejetées. Par la suite, pour générer ce type de sorties, nous devons utiliser une matrice MC^N qui respecte la contrainte suivante pour chacune des classes i :

$$\sum_{j=1, i\neq j}^{N} TC_{i,j}^N = N*(100\% - TR_i) - TL_i^N \qquad (2)$$

Avec une matrice MC^K ($K \in [2..N-1]$, on peut générer des sorties dont chacune peut contenir une, deux, …, ou K solutions. Dans ce cas, la somme des taux pour chaque classe de la matrice doit être strictement inférieure à K*100. Pour cela, nous devons respecter la contrainte (3) pour calculer les taux de confusion dans la matrice MC^K

$$\sum_{j=1, i\neq j}^{N} TC_{i,j}^k < K*(100\% - TR_i) - TL_i^k \qquad (3)$$

4.2 Construction de $MC^{K'}$ à partir de MC^K

Nous avons vu précédemment que la construction de chaque matrice nécessite trois étapes : génération des taux de rejet puis des taux de reconnaissance et enfin des taux de confusion. Bien que la construction de la matrice $MC^{K'}$ respecte ces trois étapes, elle est différente de celle de MC^K. A part les taux de rejet qui sont les mêmes que ceux de la matrice MC^K (puisque la solution rejet est représentée toujours par une seule proposition), tous les autres taux doivent être inférieurs à ceux de la matrice MC^K. La génération des taux de confusion est réalisée de la même manière que celle présentée dans la section 3 en utilisant la contrainte (2) puisque la matrice $MC^{K'}$ sera utilisée pour générer des listes composées exactement de K' solutions. Chaque taux $TC_i^{k'}$ doit être aussi inférieur ou égal à TC_i^k. Cependant, le calcul des taux de reconnaissance se fait d'une manière différente de celui de la matrice de MC^K. Chaque taux de reconnaissance $TL_i^{k'}$ à calculer doit respecter les contraintes suivantes:

$$TL_i^{k'} \in [TL^{K'} - \alpha^{K'}; TL^{K'} + \alpha^{K'}]$$

$$TL_i^{k'} \leq 100 - TR_i$$

$$TL_i^{k'} \leq TL_i^k$$

5 Génération des sorties

Dans cette section, nous présentons les procédures nécessaires pour la génération de sorties du simulateur de classifieur à partir des matrices créées précédemment.
Rappelons ici que nous générons d'abord les sorties à K' solutions à partir du comportement fixé par $MC^{k'}$ puis nous générons les (K-K') solutions restantes à partir du comportement fixé par MC^K.

5.1 Génération de sorties à partir de $MC^{K'}$

Une fois la matrice $MC^{K'}$ construite, l'étape suivante consiste à générer la liste des solutions à partir de cette matrice. Pour générer ces sorties qui, en moyenne, doivent respecter un taux de reconnaissance et un taux de rejet dans les K' premières solutions, on doit connaître le nombre d'étiquettes à distribuer dans les listes de sorties de chacune des classes. Pour cela, la première étape à réaliser consiste à calculer les effectifs à partir de la matrice $MC^{K'}$ selon le nombre de sorties désiré NS. Cette phase permet de passer d'une matrice de probabilités $MC^{K'}$ à une matrice d'effectifs $MN^{K'}$. L'étape suivante consiste à générer NS sorties pour chacune des classes. Une sortie peut être composée d'une solution rejet ou d'une liste de K' solutions. Pour affecter l'un de ces deux types de solutions à chaque sortie, on tire aléatoirement une valeur X dans l'intervalle $[1..C_i + R_i]$ où C_i est le nombre d'étiquettes restantes à générer et R_i est le nombre d'étiquettes à rejeter pour la classe i. Il faut noter que chaque liste de K' solutions à générer doit contenir des étiquettes de classes différentes: une classe ne peut pas figurer deux fois dans une même liste de solutions. Pour cela, on doit tirer pour chaque sortie choisie aléatoirement, K' étiquettes dans les classes ayant les effectifs les plus élevés. L'algorithme de cette procédure peut être présenté comme suit :

```
Entrées:
            MN^K': matrice de nombres d'étiquettes à générer;
            NS: nombre de sorties à générer par classe
Sorties:    S_i^l pour i=1 à N et l=1 à NS : l^ème sortie pour la classe i
Début
  Pour chaque classe i de 1 à N faire
    Initialiser C_i à  Σ_{j=1}^{N} MN^K[i][j]
    Initialiser R_i à  MN^K[i][N+1]
    Tant que toutes les sorties ne sont pas traitées faire
      Choisir aléatoirement un nombre entre 1 et NS
      Tirer X aléatoirement dans [1, C_i+R_i]
      Si X ∈ ]C_i, C_i + R_i] alors
        Placer une solution rejet dans S_i^l
        Décrémenter R_i
      sinon
        Choisir K' étiquettes dans les classes ayant les effectifs les plus
élevés
        Placer ces étiquettes aléatoirement dans S_{i,j}^l
        C_i ← C_i - K'
Fin
```

Table 1. *Génération de listes de sorties à partir de $MC^{K'}$*

5.2 Génération des solutions restantes

Nous avons vu précédemment comment générer des sorties respectant le comportement donné dans les K' premières solutions. En tenant compte de ces solutions et de la matrice de présence MC^K, on génère les K-K' solutions restantes. Pour cela, la matrice des effectifs est construite à partir de la matrice de présence MC^K et la différence entre cette matrice et celle que nous avons construite précédemment à partir de $MC^{K'}$ (c'est à dire $MN^{K'}$) est obtenue. Soit MN^K la matrice contenant les effectifs restants. Les effectifs du rejet ne sont pas pris en compte dans cette étape puisque les solutions rejetées sont déjà générées dans la phase précédente.

Pour chaque sortie choisie aléatoirement, on tire une valeur $reslab_i^l$ indiquant le nombre d'étiquettes restantes à placer dans cette sortie. $reslab_i^l$ peut être égal à 0 mais ne doit pas dépasser la valeur K - K'. Bien évidemment, les solutions à tirer sont les étiquettes des classes ayant les effectifs les plus élevés et doivent être différentes de celles tirées déjà dans la liste de K' solutions. Cette génération peut être illustrée par l'algorithme suivant :

```
Entrées :
       MN^K : matrice des effectifs restants des classes
       NS: nombre de sorties par classe
Sorties :
       S_i^l : liste des solutions restantes à générer (i de 1 à N et l de 1 à NS)

Début
  Pour chaque classe i de 1 à N faire
    Initialiser C_i à  Σ_{j=1}^{N} MN^K[i][j]
    Tant que les sorties ne sont pas traitées faire
      Tirer aléatoirement un numéro l entre 1 et NS
      Si S_i^l ne contient pas une solution rejet alors
        Tirer reslab_i^l étiquettes dans les classes ayant les effectifs les plus élevés dans MN^K
        Placer ces étiquettes aléatoirement dans S_{i,j}^l (après les nlab_i^l étiquettes)
        Décrémenter le nombre des étiquettes choisies dans MN^K
Fin
```

Table 2 : *Génération des sorties respectant deux comportements dans les K solutions*

Finalement, on associe une confiance à chaque étiquette tirée dans la liste de solutions. Les confiances de chaque sortie sont ordonnées de manière décroissante. La confiance de la première solution de chaque liste doit être supérieure à 100/(nombre de solutions dans la liste). Cette contrainte n'est appliquée que pour les sorties qui sont composées par deux solutions au moins.

6 Résultats expérimentaux

Le but de cette section est de montrer comment on peut utiliser notre simulateur pour étudier le comportement des méthodes de combinaison. Généralement, ces méthodes sont classées selon leur complexité. Les méthodes simples (vote majoritaire [14, 15], Borda count [7, 22, 18], somme [11], moyenne et produit [21]) sont basées sur les règles fixes et n'utilisent aucune information en dehors des sorties des classifieurs à combiner. Les méthodes complexes (moyenne pondérée [20] ou régression logistique [6]) tiennent compte du comportement de chaque classifieur à combiner en leur affectant un poids déterminé le plus souvent par apprentissage. Les chercheurs sont d'accord sur le fait que les méthodes simples (ou fixes) sont plus robustes que les méthodes pondérées puisqu'il est difficile dans les applications réelles d'estimer de façon fiable les poids optimaux [20]. Toutefois, les conditions d'utilisation des méthodes fixes ne sont pas très claires en particulier pour les méthodes de type rang.

Nous comparons dans cette section trois méthodes de combinaison de type rang afin de montrer expérimentalement comment une méthode exploite l'information présente en première (Top1), dans les deux premières (Top2) ou dans les trois premières propositions (Top3) des sorties de classifieurs, Top*n* représentant le taux de reconnaissance d'un classifieur pour lequel la vraie classe est présente parmi les *n* étiquettes des solutions à combiner. Nous avons considéré trois méthodes de type rang : le Borda Count (BC), la médiane (M) et le meilleur rang (MR).

Soit un problème à N classes et L classifieurs dans lequel chaque classifieur propose une liste ordonnée de classes rangées par ordre décroissant de préférence.

Pour une classe j, le BC est la somme des rangs proposés par les L classifieurs [4, 6]:

$$BC_j = \sum_{i=1}^{L} (N + 1 - r_{ij})$$

où r_{ij} est le rang attribué par le i^{ème} classifieur à la j^{ème} classe, le rang 1 correspondant à la première place dans la liste et le rang N à la dernière place. Les classes sont ensuite triées par ordre décroissant selon leur BC. Le nouvel ordre détermine les rangs finaux.

La médiane est une variante du borda count. Elle prend en compte les rangs des indices du milieu de la liste [22]. Pour une classe j, la médiane est calculée comme suit:

$$M_j = \begin{cases} r_{\frac{L+1}{2}j} & \text{si } L \text{ est impair} \\ \dfrac{r_{\frac{L}{2}j} + r_{\frac{L+2}{2}j}}{2} & \text{si } L \text{ est pair} \end{cases}$$

La méthode du meilleur rang consiste à attribuer à chacune des classes le rang le plus élevé H_j parmi les rangs proposés par les classifieurs et d'ordonner la liste selon ces rangs [6, 7] :

$$H_j = \max_{i=1}^{L} r_{ij}$$

De par leur spécificité, ces trois méthodes de combinaison exploitent *a priori* de façons différentes les performances des classifieurs à combiner. Pour vérifier cette hypothèse, l'expérimentation menée consiste à envisager un problème de classification à N=5 classes et à combiner L=3 classifieurs dont les performances de reconnaissance en top K' (pour K'=1, 2 et 3) varient de 50% à 95% par pas de 5%, en fixant le taux de reconnaissance dans les N=5 premiers choix TL^{i5} à 100% (la bonne solution existe donc toujours dans la liste des 5 propositions).

Notons que dans la combinaison parallèle, l'ordre des classifieurs n'est pas pris en compte. Cela veut dire que les ensembles de classifieurs ayant les taux de reconnaissance (50%, 60%, 50%) ou (60%, 50%, 50%) sont les mêmes. Pour éviter de traiter les mêmes ensembles de classifieurs, nous simulons un premier classifieur avec un taux de reconnaissance $TL^{1k'}$ variant de 50% à 95% par pas de 5%, un deuxième classifieur avec un taux de reconnaissance $TL^{2k'}$ variant de $TL^{1k'}$ à 95% par pas de 5% et un troisième classifieur ayant un taux de reconnaissance $TL^{3k'}$ variant de $TL^{2k'}$ à 95% par pas de 5%. Nous obtenons donc au total 220 ensembles de 3 classifieurs et nous répétons l'évaluation 5 fois pour chaque ensemble (ce qui nous permet d'avoir pour chaque ensemble des résultats différents et d'estimer grossièrement la « variance » dans les résultats). Le comportement de chacune des méthodes de combinaison est évalué en terme de taux de reconnaissance moyen en sortie de combinaison.

Nous présentons dans la figure 3 les résultats obtenus par les 3 méthodes de combinaison en fonction de K'. La mesure *D* utilisée en abscisse est définie comme suit :

$$D = \frac{1}{L}\sum_{i=1}^{L} TL^{ik'} + \frac{\Delta}{\frac{1}{L}\sum_{i=1}^{L} TL^{ik'}}$$

où Δ est la différence entre le meilleur et le plus mauvais des classifieurs de chaque ensemble. Elle permet donc d'évaluer le taux de reconnaissance obtenu par chaque méthode de combinaison en fonction de la moyenne des performances des classifieurs à combiner pondérée par un terme qui permet de différencier les ensembles de classifieurs ayant la même moyenne en reconnaissance.

Si on considère les résultats en Top 1 de la figure 3a, on constate que les trois méthodes de combinaison se comportent différemment. La médiane produit la meilleure performance pour tous les ensembles de classifieurs. Le borda count améliore aussi la performance des classifieurs individuels. Alors que le meilleur rang donne des taux inférieurs à ceux des classifieurs combinés. Ce comportement persiste lorsque les taux des classifieurs augmentent. Ceci montre que le borda count et la médiane exploitent mieux la première solution que la méthode du meilleur rang. En Top 2 (figure 3b), le comportement de la médiane s'approche de celui du borda count. Une petite amélioration de performance du meilleur rang est obtenue dans le cas de la combinaison de classifieurs faibles (dont le taux de reconnaissance est inférieur à 55). Mais cette méthode échoue lorsque la performance des classifieurs augmente.

En regardant maintenant les résultats de la Figure 3c, on constate que le meilleur rang améliore le taux de reconnaissance seulement pour des ensembles de classifieurs dont le taux de reconnaissance moyen est inférieur à 60%. Le borda count dans ce cas se comporte mieux que la médiane. Mais lorsque les taux de reconnaissance des classifieurs sont supérieurs à 60%, les performances du borda count et de la médiane sont très similaires. Nous pouvons constater à travers ces résultats que la médiane et le borda count sont très sensibles aux erreurs dans les 3 premières rangs lorsqu'elles combinent des classifieurs faibles. Nous pouvons donc émettre l'hypothèse que lorsque le nombre de solutions augmente, le borda count exploite mieux la liste de solutions que la médiane.

Les résultats présentés dans cette section montrent bien que l'utilisation des sorties de classifieurs varie d'une méthode à une autre. En particulier, la médiane et le borda count exploitent bien l'information sur la classe correcte lorsqu'elle est placée en première ou en deuxième proposition quelque soit la performance des classifieurs à combiner. Cependant, le meilleur rang n'est intéressant qu'avec une liste de solutions plus grande avec des classifieurs de faible performance.

7 Conclusion et perspectives

Dans cet article, nous avons proposé une nouvelle méthode de génération artificielle de sorties de classifieur permettant de simuler n'importe quel problème de reconnaissance à partir d'un jeu réduit de paramètres tels que le taux moyen de reconnaissance et de rejet, la variation dans les taux de reconnaissance par classe, …. Cette méthode est basée sur deux étapes. La première étape consiste à construire des matrices temporaires selon les performances désirées (paramètres d'entrée du simulateur). La deuxième étape consiste à générer des sorties respectant en même temps deux comportements dans les K premières solutions selon ces matrices. Le contrôle de ces paramètres permet de produire une grande variabilité de performance.

Grâce à sa flexibilité et sa capacité de simuler n'importe quel comportement, notre simulateur peut être maintenant utilisé pour évaluer les méthodes de combinaison. Il peut nous aider à caractériser expérimentalement les situations optimales d'utilisation des méthodes de combinaison dans des problèmes différents de reconnaissance.

Pour approcher un peu plus le comportement d'un classifieur réel, nous envisageons de contrôler la corrélation à l'intérieur même de la liste de solutions c'est-à-dire en générant des sorties qui respectent à la fois un comportement en première proposition, dans les deux premières, …, dans les K premières. Il est à noter également que la dépendance entre les sorties de classifieurs a une influence particulière sur le comportement des méthodes de combinaison. C'est pourquoi nous envisageons de développer un module de génération des sorties dépendantes en se basant sur notre simulateur afin d'étudier l'effet de la corrélation de classifieurs sur la performance des méthodes de combinaison.

8 Bibliographie

[1] B. Achermann & H. Bunke, "Combination of classifiers on the decision level for face recognition", Rapport technique, university of Bern, 1996.

[2] R. Bajaj & S. Chaudhury, "Signature verification using multiple neural classifiers ", Pattern Recognition, Vol 30, no 1, pp. 1-7, 1997.

[3] R. Brunelli & D. Falavigna, "Person identification using multiple cues", IEEE Trans. Pattern Anal. Mach. Intell., vol 17, no 10, pp. 955-966, 1995.

[4] Jean-Charles de Borda. "Mémoire sur les Elections au Scrutin", Histoire de l'Academie Royale des Sciences, Paris, 1781.

[5] R.P.W. Duin and D.M.J. Tax, "Experiments with classifier combinig rules", in Proc. First Int. Workshop on Multiple Classifier System, MCS 2000, vol 1857, Springer, Berlin, pp. 16-29, 2000.

[6] T.K. Ho, "A theory of multiple classifier systems and its application to visual word recognition", Ph.D. dissertation, Dep. Of Computer Science, SUNY at Buffalo, 1992.

[7] T.K. Ho, J.J. Hull and S.N. Srihari, "Decision Combination In Multiple Classifier Systems", IEEE Transactions On Pattern Analysis And Machine Intelligence, Vol.16, No. 1, (1994) 66-75.

[8] Y.S. Huang & C.Y. Suen "A method of combining multipe experts for the recognition of unconstrained handwritten numerals", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 17, no 1, pp. 90-94, 1995.

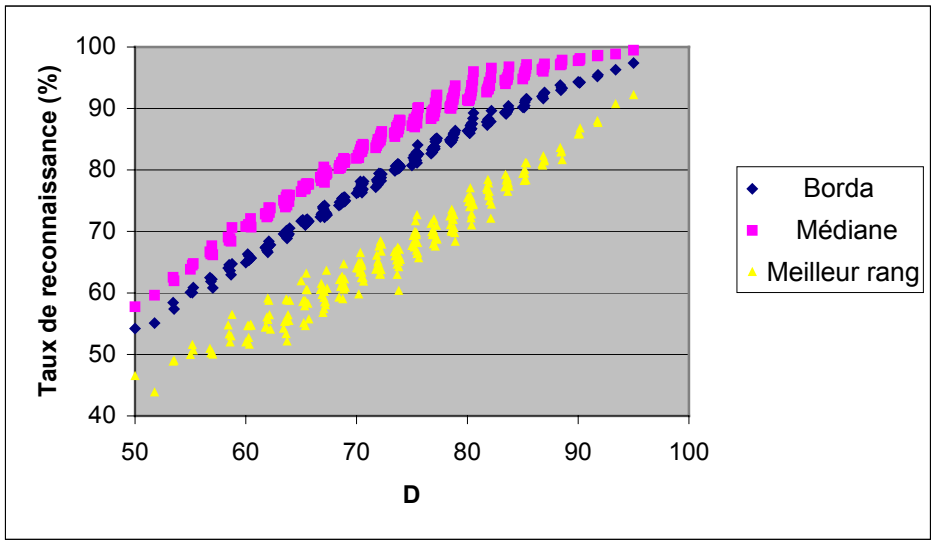
[9] S. Impedovo, A.Salzo, "Evaluation Of Combination Methods", Proc. ICDAR, Bangalore, India, (1999) 394-397

[10] A.K. Jain, R.P.W. Duin and J. Mao, "Statistical pattern recognition : A Review", IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 22, no. 1, 2000.

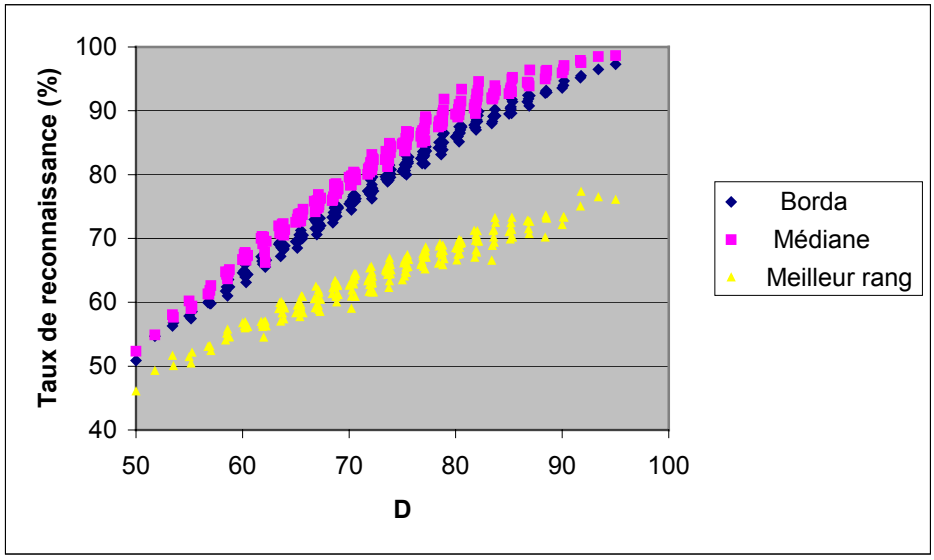
[11] J.Kittler, M. Hatef, R.P.W.Duin & J.Matas, "On Combining Classifiers", IEEE Transactions On Pattern Analysis And Machine Intelligence, Vol. 20, No. 3 (1998).

[12] L.I. Kuncheva, C.J. Whitaker, C.A. Shipp and R.P.W. Duin, "Is independence good for combining classifiers ?", in

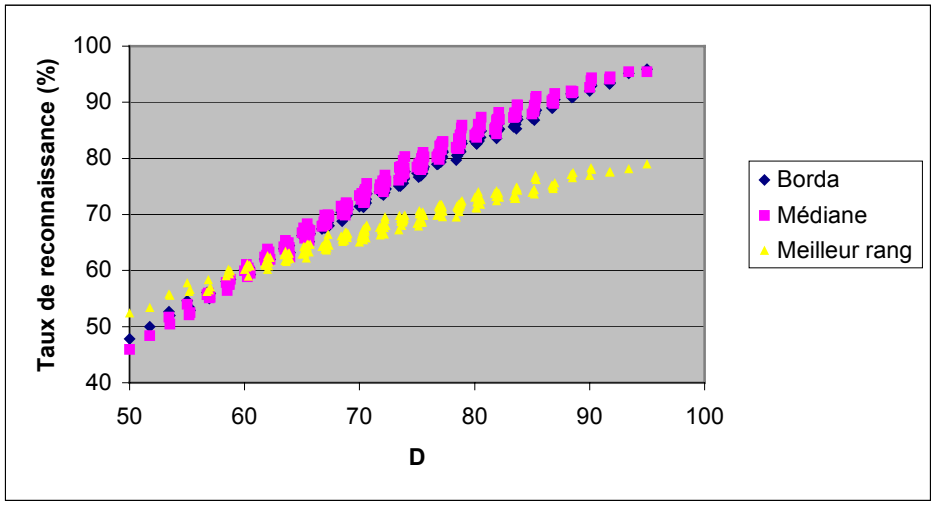
Proc. 15th International Conference on Pattern Recognition, Barcelona, Spain, vol. 2, pp. 169-171, 2000.
[13] L.I. Kuncheva and R.K. Kountchev, "Generating classifier outputs of fixed accuracy and diversity", Pattern Recognition Letters, vol. 23, pp. 593-600, 2002.
[14] L. Lam & C.Y. Suen, "A theoretical analysis of the application of majority voting to pattern recognition", International Conference on Pattern Recognition, pp. 418-420, Jerusalem, 1994.
[15] L. Lam & C.Y. Suen, "Application of majority voting to pattern recognition : an analysis of its behaviour and performance", IEEE Trans. On System, Man, and Cybernetics, Vol 27, No 5, Sepember 1997.
[16] V.D. Lecce, G. Dimauro, A. Guerrierro, S. Impedovo, G. Pirlo and A. Salzo, "Classifier combination : the role of a-priori knowledge", in Proc. of the 7th International Workshop on Frontiers in Handwriting Recognition, Amsterdam, The Netherlands, Sept. 11-13, pp. 143-152, 2000.
[17] R. Maclin and D. Opitz, "An empirical evaluation of bagging and boosting", in Proceedings of the 14th National Conference on Artificial Intelligence, Cambridge, MA., AAAI Press/MIT Press, pp. 546-551, 1997.
[18] J.R. Parker, "Evaluating classifier combination using simulated classifiers", Research Report #2000/659/11, Department of Computer Science, University of Calgary, Canada, 2000.
[19] J.R. Parker, "Rank and response combination from confusion matrix data", Information Fusion, vol. 2, pp. 113-120, 2001.
[20] F. Roli, G. Fumera, J. Kittler, Fixed And Trained Combiners For Fusion Of Imbalanced Pattern Classifiers. The Fifth International Conference On Information Fusion, Annapolis (Washington) USA (2002)
[21] D.M.J. Tax, M.V. Breukelen, R.P.W. Duin and J. Kittler, "Combining multiple classifiers by averaging or by multiplying? ", Pattern Recognition, vol. 33, pp. 1475-1485, 2000.
[22] M. Van Erp and L. Schomaker, "Variants of the borda count method for combining ranked classifier hypotheses", In: Proceedings of the seventh International Workshop on Frontiers in Handwriting Recognition, Amsterdam, 11-13 September, pp. 443-452, 2000.
[23] M. Van Erp, L. Vuurpijl and L. Schomaker, "An overview and comparison of voting methods for pattern recognition", 8th International Workshop on Frontiers in Handwriting Recognition, Niagara-on-the-Lake, Ontario, Aug. 6-8, pp. 195-200, 2002.
[24] B.H. Xiao, C.H. Wang and R.W. Dai, "Adaptive combination of classifiers and its application to handwritten chinese character recognition", in Proc. ICPR'00, Barcelona, Spain, vol. 2, pp. 2327-2330, 2000.
[25] L. Xu, A. Krzyzak & C.Y. Suen "Methods of combining multiple classifiers and their applications to handwriting recognition", IEEE Transaction on Systems, Man, And Cybernetics, vol 22, no 3, pp. 418-435, may/june 1992.
[26] E.N. Zois & V. Anastassopoulos, "Fusion of correlated decisions for writer verification", Pattern Recognition, vol 32, pp. 1821-1823, 1999.
[27] H. Zouari, L. Heutte, Y. Lecourtier & A. Alimi, "Un panorama des méthodes de combinaison de classifieurs en reconnaissance de formes", Reconnaissance de formes et Intelligence Artificielle (RFIA), pp. 499-508, 8-10 janvier 2002.
[28] H. Zouari, L. Heutte, Y. Lecourtier & A. Alimi, "Simulating Classifier Outputs for Evaluating Parallel Combination Methods", LNCS 2709, 4th International Workshop, Multiple Classifier Systems, Guildford, UK, 11-13 June, pp. 296-305, 2003.



(a)



(b)



(c)

Figure 3 *: Performances des méthodes de combinaison utilisant des ensembles de 3 classifieurs (a) en Top 1 (b) en Top 2 (c) en Top 3*