Apprentissage Incrémental à Base de Population des Réseaux de Neurones Appliqué à la Reconnaissance de la Parole Continue

TLEMSANI Redouane, AZIB Lamia et BENYETTOU Abdelkader

Laboratoire SIgnal-IMage-PArole (SIMPA)
Dépt. Informatique, Faculté des Sciences, Université des Sciences et de la Technologie d'Oran – USTO, Algérie
tlemsani_redouane@univ-usto.dz
azib_13@yahoo.fr
aek.benyettou@univ-usto.dz

ABSTRACT

Significant progress accomplished thanks to the development of powerful algorithms has been achieved in the field of the automatic speech recognition (ASR). Researchs based on a simplified biological model preserve the significant partial solutions lead to the appearance of new class called the Estimation of Distribution Algorithms (EDA). Population Based Incremental Learning (PBIL) is a technique whose stochastic research and the search for optimization are combined. It is a statistical approach with evolutionary calculation of the type EDA in order to adapt systems of recognition based on the artificial neural networks (NN).

Our real contribution lead to an improvement by PBIL of 16% vs Genetic Algorithms (AG) and 9.93% vs Evolutionnary Strategies (ES).

1. INTRODUCTION

Les systèmes de reconnaissance automatique de la parole (SRAPs) sont aujourd'hui bien connus dans le monde de l'intelligence artificielle et suscitent l'intérêt d'un public de plus en plus large.

Il existe une catégorie de problèmes pour lesquels il est difficile, voir impossible, de trouver une solution en un temps limité. Il est alors utile de trouver une technique permettant la localisation rapide de solutions optimales, sachant que l'espace de recherche a une taille et une complexité suffisamment importantes pour éliminer toute garantie d'optimalité [1]. Pour cela, un système capable de s'auto-modifier au cours du temps, tout en améliorant sa performance dans l'accomplissement des tâches auxquelles il est confronté, semble ouvrir la voie à une recherche intéressante.

Les algorithmes évolutionnaires sont basés sur des principes simples. En effet, peu de connaissances sur la manière de résoudre ces problèmes sont nécessaires, même si certaines peuvent être exploitées afin de rendre plus efficace l'évolution (en effet, il n'est pas réaliste d'espérer obtenir une méthode d'optimisation raisonnablement efficace sans aucune connaissance sur le domaine à traiter). C'est pourquoi, dans de nombreux domaines, les chercheurs ont été amenés à s'y intéresser. Dans cet article, nous nous sommes intéressés à un type d'algorithme, inspiré de l'évolution naturelle, qui propose de faire évoluer des populations d'individus dans un environnement en favorisant la survie des individus les mieux adaptés à cet environnement. On connaît l'efficacité de ce type d'approche pour l'optimisation de fonctions complexes [2], pour la recherche de solutions à des problèmes ayant un nombre de variables très grand. Nous cherchons à savoir si ce type d'algorithmes appelés Algorithmes d'Estimation de Distributions peut constituer une nouvelle approche pour l'adaptation des SRAPs à plusieurs locuteurs.

2. ALGORITHMES D'ESTIMATION DE DISTRIBUTIONS

Comme d'autres algorithmes évolutionnaires (Algorithmes Génétiques AG, stratégies d'évolution ES) EDAs maintiennent et améliorent successivement une population de solutions potentielles jusqu'à ce qu'un critère d'arrêt soit satisfiable. EDA remplace les opérateurs génétiques par deux étapes suivantes [3]:

- L'estimation de la probabilité de distribution des solutions prometteuses sélectionnées;
- Prélèvement de nouveaux individus selon l'estimation de modèle probabiliste.

Tout d'abord, EDA commence par la génération de la population initiale D_0 constituée de M individus (on dit aussi solutions). La génération de ces M individus est effectuée habituellement en assumant une distribution uniforme sur chaque variable. Ensuite, dans chaque génération, S meilleures solutions ($N \leq M$) sont sélectionnées à partir de D_0 selon les critères de sélection choisis (voir Fig.1). EDA extrait explicitement l'information statistique globale à partir de l'ensemble des individus sélectionnés de la génération précédente et construit un modèle de probabilité de distributions, qui est estimé pour représenter les interdépendances entre les variables en utilisant une des méthodes d'EDA.

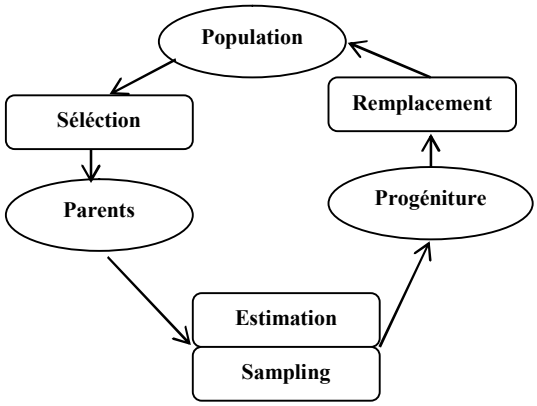

Figure 1 : Cycle d'évolution des EDAs

Cette étape est connue comme la plus cruciale, car la représentation convenable des dépendances entre les variables est essentielle pour une évolution appropriée vers les solutions optimales. Finalement, les nouvelles solutions sont prélevées du modèle probabiliste ainsi établi et avec le remplacement entier ou partiel des solutions de la population courante, en générant la nouvelle population (voir Fig.2).

$D_0 \leftarrow$ Générer M individus (la population initiale) aléatoirement
Répéter de $l = 1, 2, \ldots$ jusqu'a ce qu'un critère d'arrêt est rencontré
$D_{l-1}^{Se} \leftarrow$ Sélectionner $N \le M$ individus à partir de D_{l-1} selon une méthode de sélection
$p_l(x) = p(x / D_{l-1}^{Se}) \leftarrow$ Estimer la probabilité de distribution d'un individu parmi les individus sélectionnés
$D_l \leftarrow$ Prélever M individus (la nouvelle population) à partir de $p_l(x)$

Figure 2 : Pseudo-code d'EDA

2.1. Apprentissage Incrémental à Base de Population

PBIL est une technique dont la recherche stochastique est doublée à la recherche d'optimisation. C'est une approche statistique au calcul évolutionnaire qui combine les mécanismes d'un AG avec l'apprentissage concurrentiel. La combinaison de ces deux méthodes conduit à un outil plus simple, et meilleur qu'un AG standard sur un grand ensemble de problèmes d'optimisation en termes de vitesse et d'exactitude [4].

L'algorithme PBIL est un processus de recherche stochastique qui obtient son information directionnelle de la meilleure solution précédente. Le but de l'algorithme est de créer un vecteur de probabilité PV qui peut être considéré comme un prototype pour les prochains vecteurs d'évaluation dans l'espace de recherche exploré.

PBIL est une technique stochastique simple d'optimisation qui peut être appliquée rapidement à une variété de problèmes. Le domaine principal de l'application de la technique est consacré aux problèmes qui sont trop multimodaux ou discontinus pour le gradient ou les méthodes rétro, etc.

Règle de mise à jour de PV

La règle de mise à jour du vecteur de probabilité décrite en (1) est semblable à celle des poids dans un réseau d'apprentissage concurrentiel. Il est intéressant de noter que plusieurs méthodes de recherche examinent chaque gène indépendamment. Cette présentation a négligé une grande partie d'information que PBIL préserve.

La raison d'acceptation de l'indépendance n'est pas nuisible, PBIL n'essaie pas de représenter la population toute entière par le PV, mais plutôt, il représente un seul point (la meilleure solution), en se basant sur le vecteur d'évaluation le plus élevé autour duquel la prochaine population des points devrait être générée. Les valeurs dans le PV sont graduellement décalées vers celles binaires du meilleur vecteur [4]. La formule suivante décrit le processus de mise à jour du PV:

$$PV_i = ((1 - LR)PV_i) + (LR * vecteur_i) \qquad (1)$$

PV_i: la probabilité de générer 1 dans le bit de position i.
$vecteur_i$: la $i^{éme}$ position dans le meilleur vecteur des solutions vers lequel on déplace le vecteur de probabilité.
LR: taux d'apprentissage
L'équation (1) est alors appliquée au PV pour incorporer l'information directionnelle du meilleur chromosome.

Le taux de recherche SR

SR est la distance que le niveau final de convergence est atteint approximativement. Nous pouvons faire fonctionner l'algorithme après fixation de la taille de la population et le calcul de la valeur maximale de SR qui minimise le nombre d'évaluations de fonction. L'équation (2) donne une approximation empirique à la valeur optimale de SR [5].

$$SR_{opt} \approx 1 - (2/\sqrt{P})^{1/b} \qquad (2)$$

Où P : taille de population ;
b : nombre de bits dans un individu.
Un approximatif de la valeur optimale de SR ($0<SR<0.5$) est souvent un bon point de départ pour exécuter l'algorithme.
Si le SR est mis à zéro, le PV peut converger à 0 ou à 1. Une fois que cette valeur terminale a été atteinte, l'élément converge dans la fausse direction et il n'y a aucune méthode pour qu'elle soit corrigée. L'augmentation de SR empêche le PV de converger exactement à 0 ou à 1: plus la valeur SR est élevée, moins probable l'algorithme risque de tomber dans des optimums locaux.

Dans les dernières parties de la recherche, le décalage de mutation ou quantité de mutation (MS) joue un rôle important. D'une façon analogue aux AGs, un opérateur semblable est défini dans PBIL. Pour PBIL, il y a deux manières de définir un opérateur de mutation: exécuter la mutation directement sur les vecteurs générés ou sur le vecteur de probabilité; cette mutation peut-être définie

comme une petite probabilité de perturbation sur chacune des positions dans le PV. Toutes les deux formes de mutation ont le même effet que celui dans les AGs. Pour nos expériences, la deuxième forme de mutation a été mise en application.

Le rôle de la mutation est d'empêcher le vecteur de prototype, PV de converger trop rapidement à une valeur extrême (0 ou 1) dans chacune des positions de bit. La mutation est souvent utilisée dans PBIL pour aider à augmenter l'espace de recherche, elle peut cependant ne pas jouer un rôle crucial dans PBIL contrairement à la recherche génétique. La mutation change idéalement la direction du PV en le dirigeant vers les régions de recherche alternative. Les divers arrangements pour mettre en application la mutation existent cependant.

L'opérateur de mutation en équation (3) sera plus facile à généraliser que le taux d'apprentissage, parce qu'il exige seulement une direction vers une distribution aléatoire.

$$MS = \frac{2 * SR}{1 - 2 * SR(1 - LR)} \qquad (3)$$

Contrairement aux AGs qui nécessitent un niveau de connaissance élevé dans le domaine d'application afin de choisir les valeurs appropriées pour ces paramètres, PBIL a seulement deux paramètres à justifier: la taille de la population et le taux d'apprentissage. Tandis que, les autres paramètres sont obtenus par des calculs, ceci facilitera l'étude.

La figure 3 décrit les principales étapes de l'algorithme général contenant les constantes et les variables nécessaires pour la recherche de PBIL [6].

```
P ← initialiser le vecteur de probabilité (chaque position = 0.5).
Boucle # GENERATIONS
    # Générer les échantillons
    i ← boucle # SAMPLES
        sample_i ← générer le vecteur d'échantillon selon les probabilités dans P
        evaluation_i ← evaluer (sample_i)
    # Trouver le meilleur échantillon
    Best ← trouver le vecteur correspondant à l'évaluation maximum
    # Mise à jour du PV
        i ← boucle # LENGTH
        P_i ← P_i * (1.0 - LR) + Best_i * (LR)
    # Mutation de PV
        i ← boucle #LENGTH
            if (random (0,1] < MP)
                P_i ← P_i * (1.0 - MS) + random (0.0 or 1.0) * MS
```

Figure 3 : L'algorithme général de PBIL

3. MODÉLISATION DE L'HYBRIDATION

La première étape pour permettre la manipulation des réseaux de neurones NN par un PBIL est de définir sous quelle forme au sens structure de données, le PBIL verra le NN en tant qu'individu d'une population.
Considérons un réseau ayant trois couches de neurones avec deux neurones en entrée, deux en couche cachée et un en sortie. Soit W_{ij} le poids d'une connexion du neurone i vers le neurone j et Bj les biais. La figure 4 donne une représentation de ce réseau et le chromosome associé [7].

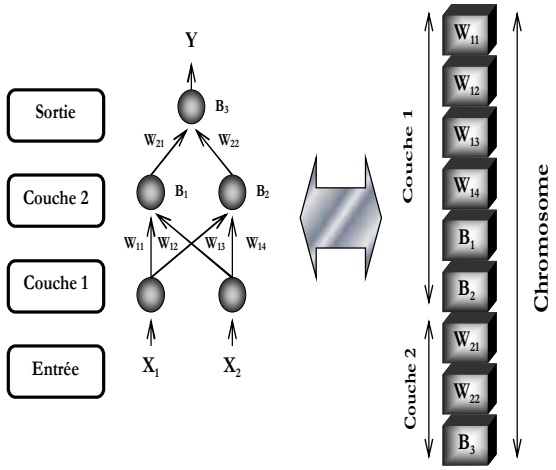

Figure 4 : La représentation évolutionnaire d'un réseau de neurones.

L'adaptation d'un individu (appelée fitness) décrit l'adéquation de celui-ci à son environnement, plus précisément, l'individu aura une adaptation d'autant moins élevée qu'il sera une bonne solution dans le problème de l'optimisation d'un NN. L'adaptation de l'individu sera donc l'erreur quadratique moyenne [8].

4. EXPERIENCES ET RESULTATS

Les signaux ont été échantillonnés à 16 kHz avec une analyse cepstrale sous l'échelle Mel, prise toutes les 20ms dans des fenêtres de Hamming de 25ms donnant chacune 12 coefficients MFCC (Mel Frequency Cesptral Coefficients) et l'énergie résiduelle correspondante. Le sous corpus de TIMIT est constitué de 6 voyelles (*ah, aw, axa x_h, uh uw*), 6 fricatives (*dh, f, sh, v, z, zh*) et 6 plosives (*b, d, g, p, q, t*) présentant un nombre important de données à traiter (voir Table 1).

Pour cela, nous avons fait une classification afin de compresser les données en utilisant l'algorithme LBG [9] avec la variante des k-moyennes relative à la quantification vectorielle.

Le sous corpus utilisé est traité globalement au niveau de la quantification vectorielle QV avec 64 prototypes. L'utilisation de la QV a eu l'effet de réduire le temps d'apprentissage en assignant à chaque intervalle de phonème traité un vecteur de 64 composantes binaires [10].

Table 1 : Taux de reconnaissance obtenus pour différents modèles implémentés appliqué au sous-corpus.

	Train	Test	NN-GA%	NN-ES%	NN-PBIL%
/ah/	2200	879	48.35	10.58	40.61
/aw/	700	216	43.52	86.11	61.11
/ax/	3352	1323	49.73	73.84	66.29
/ax-h/	281	95	65.26	68.42	64.21
/uh/	502	221	49.32	82.80	56.56
/uw/	536	170	31.17	67.05	69.41
/dh/	2058	822	67.51	68.24	54.01
/f/	2093	911	69.15	70.91	35.24
/sh/	2144	796	43.84	46.48	97.99
/v/	1872	707	56.58	63.93	71.71
/z/	3574	1273	67.87	69.59	97.10
/zh/	146	74	51.35	52.70	90.54
/b/	399	182	52.20	58.79	61.54
/d/	1371	526	41.25	65.77	73.95
/ g/	1337	546	53.30	45.97	65.20
/p/	2056	779	40.82	51.60	69.19
/q /	3307	1191	70.44	63.47	62.05
/t /	3586	1344	29.01	52.67	87.72
Taux global	31514	12055	52.98	59.24	69.17

Pour l'apprentissage, nous avons choisi l'utilisation d'un NN à une seule couche cachée, à qui nous avons fait varier la taille de la couche d'entrée, et la taille de la couche cachée, pour enfin garder le plus performant après plusieurs tentatives et qui était un NN à une couche d'entrée de 64 neurones (relatifs au nombre de prototypes), une couche cachée de 36 neurones et une couche de sortie de 18 neurones (relatifs au nombre de phonèmes).

Une nouvelle approche NN-PBIL nous a offert une performance remarquable qui se traduit par une augmentation de 9.93% par rapport au modèle NN-ES et une augmentation de 16.19% (voir Table 1) par rapport au Modèle NN-GA [10], [11], [12], et cela revient aux propriétés de l'algorithme PBIL. On remarque bien que les fricatives donnent un résultat acceptable par rapport aux plosives et aux voyelles. Pour les trois approches hybrides, les scores peuvent être amélioré en utilisant d'une part, toute la base de TIMIT et d'autre part, les dérivées première et seconde $\Delta MFCC$ et $\Delta^2 MFCC$.

5. CONCLUSION

La dépendance entre les variables d'un individu exprimée par un modèle probabiliste a permis d'obtenir des résultats plus performants en terme de taux de reconnaissance. Jusqu'ici nous voyons que le succès et l'échec d'EDAs dépendent seulement du modèle probabiliste utilisé. Ce dernier dépend entièrement de l'ensemble de solutions prometteuses, et utilise une distribution marginale simple ou conditionnelle comme estimation de probabilité de distribution. Les expériences futures porteront sur l'apprentissage à base évolutionnaire pour l'ensemble de la base de données TIMIT.

6. BIBLIOGRAPHIE

[1] K.A. de Jong. Learning with genetic algorithm : an overview. *Machine Learning* , vol. 3, pages 121-138, 1988.

[2] P. Larrañaga and J.A. Lozano. *A Review of Estimation Distributions Algorithms, Chapter 3*. Doct. Thesis, Univ. of the Basque Country, 2002.

[3] J. Očenášek. *Parallel of Estimation of Distributions Algorithms*. Thèse de Doct., Brno university of Technology, 2001.

[4] S. Baluja. *Population Based Incremental leaning*. Technical Report CMU-CS- 94-163, Computer Science Department, Carnegie Mellon University, Pittsburg, PA, USA,1994.

[5] J.L. Shapiro. *The Sensitivity of PBIL to the Learning Rate, and How Detailed Balance Can Remove It*. Dept. computer Science Univ. Manchester, 2002.

[6] J. Cuthbert and R.M. Braun. Multi-H Modulation Indexes Selection using the PBIL Algorithm. Univ. Cap Town, *Conference Publication* No.0-7803-3019-6, IEEE, 1996.

[7] U. Seiffert. MLP Training using GAs, *ESANN'2001,* Adelaide Knowledge-Based Intelligent Eng. Syst. Center (KES), Australia, 2001.

[8] R. Belew, J. Mc Inerney and N.N. Scharaudolph, Evolving Networks using the GAs with Connectionist Learning. CSE Tech. Rep. CS90-174, *Computer Science*, UCSD, 1990.

[9] Y. Linde, A. Buzo and R. M. Gray. An Algorithm for Vector Quantized Design, *IEEE Tran.Comp.,* 28(1), 1980.

[10] R. Tlemsani, N. Neggaz, A. Benyettou et N. Dehak. Un modèle neuro-évolutionnaire appliqué à la classification phonétique. *CNIE'04, Conf. Nat Ingénierie de l'Electronique,* page 55, Univ. USTOran –Algérie, 22-23 Nov. 2004.

[11] A. Benyettou, N. Neggaz et R. Tlemsani. Evolutionnary Algorithms Based Neural Networks Trainig for Phonetic Classification. *ACIT04, Int. Conf Information Technology*, pages 468-472, Mentouri Univ. Constantine, Algeria, December 12-15, 2004.

[12] R. Tlemsani, N. Neggaz et A. Benyettou, Amélioration de l'apprentissage des RNs par les algorithmes évolutionnaires. *3ème Int. Conf. : Sciences of Electronic, Technology of Information and Telecomm, SETIT05*, Tunisia, March 27-31, 2005.