Algorithme rapide de codage conjoint source-canal pour un canal de transmission non-stationnaire

B. Kövesi[1,2], S. Saoudi[1], J.M. Boucher[1], G. Horváth[2], N. Naja[3] et A. El Kesri[3]

[1] ENST-Br, Dépt. SC., BP.832., 29285 Brest cedex, France
[2] Université Technique de Budapest, Dépt. MMT., Műegyetem rkp. 9, 1521 Budapest, Hongrie
[3] INPT, Madinat Al Irfane, Rabat, Maroc

RÉSUMÉ

Dans cet article, nous proposons l'algorithme RGSKAε qui permet de construire un dictionnaire qui est optimal dans le cas où il n'y a pas de bruit de transmission et qui est aussi robuste vis-à-vis des erreurs de transmission de tous niveaux. Cet algorithme stochastique ne nécessite aucune connaissance a priori du signal à quantifier et du canal de transmission. Il est indépendant du dictionnaire initial et il est capable de sortir de minima locaux. L'exigence de temps de construction est comparable avec celle de l'algorithme de la k-moyenne (KMA). La mise en œuvre de l'algorithme RGSKAε est aussi simple que celle de l'algorithme KMA car tous ses paramètres sont fixés, ils ne dépendent pas du type ou de la taille des vecteurs à quantifier. L'algorithme RGSKAε possède des propriétés de convergence très avantageuses : il permet de construire un dictionnaire bien équilibré même si le nombre des vecteurs d'apprentissage est inférieur à la taille du dictionnaire.

ABSTRACT

In this paper, we present the RGSKAε algorithm that allows to construct a codebook that is optimal in case of a noiseless transmission and that is robust against different levels of channel noise as well. This stochastic algorithm does not need any knowledge of the signal to be quantified or the transmission channel in advance. The algorithme is independent from the initial codebook and it is able to leave local minima. Its computational time is close to that of the k-means algorithm (KMA). The implementation of the RGSKAε algorithm is as simple as that of the KMA algorithm as all its parameters are fixed, they do not depend on the type or the size of the vectors to be quantified. The RGSKAε algorithm has execelent convergency properties: it allows to construct a well-balanced codebook even if the number of the learning vectors is smaller to the size of the codebook.

1 Introduction

Dans cet article, nous traiterons de la quantification vectorielle dans le cadre de transmissions réelles où l'on transmet les indices via un canal plus ou moins bruité. A cause du bruit de transmission, le train binaire reçu n'est que rarement une réplique exacte de celui qui a été émis, certains éléments binaires seront différents. Ces modifications de bits se traduisent au niveau du signal de synthèse par des distorsions qui sont perçues comme un bruit. Pour protéger les indices transmis, on peut utiliser un code correcteur d'erreurs. Ici, nous avons choisi d'étudier le cas où le dictionnaire possède une robustesse intrinsèque vis-à-vis des erreurs de transmission. L'utilisation d'un tel dictionnaire optimisé à la fois pour la source et pour le canal à la place d'un dictionnaire traditionnel diminue la dégradation causée par des erreurs de transmission sans augmenter le débit.

2 Modélisation du canal

Nous allons considérer un canal binaire symétrique avec ε la probabilité d'erreur sur un bit. Supposons qu'on ait émis l'indice i, qui contient R éléments, la probabilité de recevoir l'indice j, après transmission sur le canal binaire symétrique, est donnée par :

$$P\,(j/i) = (1-\varepsilon)^{R-d_H(i,j)}\,\varepsilon^{d_H(i,j)} \qquad (1)$$

où $d_H(i,\,j)$ représente la distance de Hamming calculée entre les deux indices i et j. Dans ce modèle, lors de la réception d'un indice erroné, la distance de Hamming entre les indices émis et reçu est généralement égale à 1. On peut diminuer l'influence des erreurs de transmission, sans augmenter le débit, en construisant un dictionnaire ordonné. Dans un tel dictionnaire, les indices proches dans le sens de la distance de Hamming correspondent aux vecteurs proches dans l'espace.

3 Conditions d'optimalité

Lorsqu'on connait les statistiques du canal, on peut donner les deux conditions nécessaires que doit vérifier le quantificateur optimal. Pour un dictionnaire $C = \{c_1, \cdots, c_M\}$, la partition $S = \{s_1, \cdots, s_M\}$ est optimale si :

$$s_i = \left\{ x \in R^p \mid \sum_{j=1}^{M} P\,(j/i)\,d\left(x, c_j\right) \leqslant \sum_{j=1}^{M} P\,(j/k)\,d\left(x, c_j\right);\ \forall k \neq i \right\} \qquad (2)$$

Pour une partition de l'espace S, dans le cas de la distance euclidienne quadratique, en utilisant une base d'apprentissage,

les mots de code optimaux s'expriment comme :

$$c_i = \frac{\sum_{j=1}^{M} P(i/j) \sum_{x_n \in s_j} x_n}{\sum_{j=1}^{M} P(i/j) \left| s_j \right|}; \quad i = 1, \cdots, M \qquad (3)$$

4 Algorithme de la k-moyenne généralisé

L'algorithme de la k-moyenne généralisé (GKMA[1]) recalcule itérativement la partition de l'espace (étape 1) et les mots de code du dictionnaire (étape 2) en satisfaisant alternativement aux deux conditions nécessaires (2) et (3) (voir [1], [3]).

La généralisation de l'algorithme de la k-moyenne (KMA[2]) classique a deux effets différents :

- Le dictionnaire obtenu est plus ou moins ordonné. Le dictionnaire est mieux ordonné dans le cas où on suppose une probabilité d'erreur plus forte pendant la construction.
- Le dictionnaire obtenu est compressé par rapport au dictionnaire de l'algorithme KMA, c.à.d., la distance moyenne des mots de code diminue d'une manière proportionelle à la probabilité d'erreur supposée pendant la construction. On peut comprendre ce phénomène facilement : nous avons vu que les représentants sont les centroïdes pondérés de chaque vecteur d'apprentissage (voir l'expression (3)), c'est pourquoi il se déplace vers le centre de gravité de la base.

4.1 Inconvénients de l'algorithme GKMA

Lorsque la valeur de la probabilité d'erreur supposée pendant la construction du dictionnaire est petite, son influence n'est pas assez forte pour bien ordonner les mots de code. Dans le cas inverse, quand la probabilité d'erreur supposée est grande, le dictionnaire est mieux ordonné, mais il devient à la fois trop compressé entraînant une dégradation de ses performances.

Avant de lancer l'algorithme GKMA, il faut fixer la valeur de la probabilité d'erreur $\varepsilon\prime$ qui sera utilisée dans le calcul des termes $P(j/i)$ au cours de la construction du dictionnaire. Pour faire cela, la connaissance a priori des statistiques du canal est nécessaire.

En pratique, la probabilité d'erreur du canal de transmission est rarement connue à l'avance et, de plus, elle peut évoluer pendant la transmission. Pour un canal caractérisé par un taux d'erreur différent de celui qui était prévu initialement, les performances d'un dictionnaire de GKMA ne sont sûrement pas optimales et peuvent même être inférieurs aux performances d'un dictionnaire aléatoire de l'algorithme KMA dans le cas d'un taux d'erreur plus faible que celle qui était prévu.

L'algorithme GKMA est déterministe et converge vers un minimum généralement local déterminé par le choix du dictionnaire initial. Le but de l'algorithme GKMA est triple. Il essaie d'optimiser simultanement la partition en régions, le choix de leur représentant et l'affectation des indices binaires des mots de code. L'affectation des indices binaires dans le dictionnaire initial est aléatoire, l'algorithme GKMA doit les réarranger. Comme cet algorithme ne peut pas perturber les indices, le seul moyen de faire le réarrangement est de changer les positions des mots de code dans l'espace en les déplaçant petit à petit vers une constellation ordonnée au cours des itérations. Intuitivement, on sent que cette opération nécessiterait l'augmentation provisoire de la distorsion, ce que l'algorithme GKMA ne permet pas, il respecte toujours les deux conditions nécessaires (2) et (3). Donc, il est très probable que l'algorithme GKMA aboutisse dans un minimum local assez loin du minimum global.

	Add.	**Sous.**	**Div.**	**Mult.-acc.**	**Compar.**
KMA é-1	0	MLp	0	MLp	$(M-1)L$
KMA é-2	$2Lp$	0	Mp	0	0
GKMA é-1	0	MLp	0	$ML(p+M)$	$(M-1)L$
GKMA é-2	$2Lp$	0	Mp	$2MM$	0

TAB. 1 — Nombre des opérations par itération

L'algorithme GKMA est beaucoup plus complexe que l'algorithme de la k-moyenne. Le tableau 1 permet de comparer le nombre des opérations nécessaires lors des deux étapes principales des deux algorithmes KMA et GKMA dans chaque itération. Nous notons M la taille du dictionnaire, p le nombre d'éléments par vecteur et L la longueur de la séquence d'apprentissage.

On peut vérifier facilement que c'est surtout la partition des vecteurs d'apprentissage (étape 1) selon la formule (2) qui augmente la complexité des calculs de construction d'un quantificateur.

5 Algorithme RGSKAε

Dans l'algorithme RGSKAε [3] proposé ici, nous tenons compte des statistiques du canal de transmission seulement dans l'étape de réactualisation du dictionnaire pour réduire le temps de construction. Nous calculons donc les nouveaux mots de code selon (3) comme dans le cas d'algorithme GKMA. Mais nous ne fixons plus la valeur de la probabilité d'erreur $\varepsilon\prime$. Nous supposons une probabilité d'erreur forte au début qui diminue progressivement au cours des itérations selon la formule $\varepsilon_k\prime = \varepsilon_o \alpha^k + \gamma$ avec k le numéro d'itération. Nous préciserons les paramètres ε_o, α et γ plus bas.

Pour partitioner les vecteurs d'apprentissage, nous utilisont une méthode stochastique. D'abord nous calculons les probabilités d'appartenance du vecteur d'apprentissage x_n à la classe s_i représentée par le mot de code c_i. Ensuite, nous faisons un choix aléatoire correspondant aux probabilités. Naturellement, le mot de code le plus proche a la plus grande

[1]GKMA : Generalized K-Means Algorithm
[2]KMA : K-Means Algorithm
[3]RGSKA : Reduced complexity Generalized Stochastique K-means Algorithm

probabilité d'être le gagnant stochastique. Grâce à sa nature stochastique, l'algorithme peut sortir d'un minimum local.

5.1 Fonction de probabilité

Nous avons déterminé les probabilités d'appartenance selon la formule suivante :

$$P(i,n,k) = \frac{e^{-\beta(k)\frac{d\left(x_n,c_{k,i}\right)}{\widehat{d(x_n)}}}}{\sum_{m=1}^{M} e^{-\beta(k)\frac{d\left(x_n,c_{k,m}\right)}{\widehat{d(x_n)}}}} \quad (4)$$

où k est le numéro d'itération.

$\widehat{d\left(x_n\right)}$ est la distance moyenne du vecteur d'apprentissage x_n des mots de code du dictionnaire actuel. Nous avons introduit cette normalisation des distances par la valeur moyenne dans la fonction de probabilité classique pour que les rapports de probabilités ne dépendent que des rapports des distances $d\left(x_n, c_{k,i}\right)$, et que nous puissions utiliser la même fonction $\beta(k)$ pour différents types de variables. Cela facilite l'inplantation de l'algorithme.

$\beta(k)$ est l'inverse d'une température qui augmente après chaque itération tel que $\lim_{k\to\infty}\beta(k) = +\infty$. Son rôle est de diminuer l'aspect stochastique de l'algorithme. Quand l'indice d'itération k augmente, le système refroidit, l'algorithme devient de plus en plus déterministe. Nous proposons d'utiliser la fonction linéaire $\beta(k) = c(k-1)$, avec laquelle nous avons obtenu le meilleur résultat et qui est le plus robuste vis-à-vis du choix du parametre c.

5.2 Condition initiale

L'algorithme RGSKAε ne dépend pas de tout du dictionnaire initial. Dans la première itération ($k = 1$), $\beta(k) = 0$. Dans ce cas, toutes les probabilités $P(i,n,k)$ sont identiques. Après cette itération, tous les mots de code du dictionnaire seront proche du centre de gravité de la base d'apprentissage. Dans la suite, lorsque la température baisse, l'algorithme est capable d'éloigner progressivement les mots de code vers leurs positions optimales.

Au début de l'algorithme, quand la probabilité d'erreur supposée est forte, on obtient un dictionnaire bien ordonné mais en même temps compressé. Les performances d'un tel dictionnaire sont mauvaises dans le cas où il n'y a pas d'erreurs de transmission. A la fin des itérations, la valeur supposée de la probabilité d'erreur devient pratiquement négligeable, donc le dictionnaire peut s'étendre dans l'espace pour avoir une bonne performance pour une transmission de bonne qualité. Etant donné qu'à ce moment la température est déjà assez basse, l'ordre des mots de code obtenu dans la première partie des itérations sera conservé.

L'algorithme RGSKAε peut être lancé avec la valeur initiale ($\varepsilon_o = 0.5$) la plus défavorable quand la température est très élevée. La valeur de la probabilité d'erreur diminue rapidement à une valeur raisonnable quand la température est également refroidie. La valeur de ε_k tend vers la valeur de γ. Lorsque la probabilité d'erreur du canal n'était pas connue, nous avons pris $\gamma = 0.001$. Ce petit biais n'a pas d'influence sur la performance dans le cas non-bruité, mais il aide à mieux garder l'ordre du dictionnaire à la fin des itérations. Il faut choisir la valeur du paramètre α de manière à ce que la valeur du premier terme $\varepsilon_o\alpha^k$ devienne négligeable ($\varepsilon_o\alpha^k << \gamma$) à la fin des itérations. En générale, la valeur du paramètre α est dans l'intervalle [0.8, 0.95].

5.3 Propriétés de convergence

L'algorithme KMA nécessite une base d'apprentissage qui contient au moins 5 à 10 fois plus de vecteurs que le nombre des mots de code du dictionnaire à construire. Sinon, dans chaque itération, on risque d'avoir des classes vides.

Dans le cas de l'algorithme RGSKAε, chaque mot de code est le centroïde pondéré de tous les vecteurs d'apprentissage (voir la formule (3)). Comme les facteurs de pondérations $P(i/j)$ sont non nuls, nous pouvons toujours remettre à jour la valeur de chaque mot de code. Nous avons ainsi vérifié que l'algorithme RGSKAε converge bien même si on utilise moins de vecteurs d'apprentissage que le nombre des mots de code du dictionnaire à construire. De plus, les mots de code obtenus dans ce cas sont tous différents. Les performances du dictionnaire obtenu sont meilleures que celles de la séquence d'apprentissage en l'utilisant comme un dictionnaire.

5.4 Mise en œuvre de l'algorithme

Grâce à la normalisation des distances par la valeur moyenne $\widehat{d\left(x_n\right)}$ dans la fonction de probabilité et l'utilisation de la fonction $\beta(k)$ linéaire, nous avons obtenu des très bons résultats avec les paramètres $c = 2$; $\varepsilon_o = 0.5$; $\alpha = 0.9$; $\gamma = 0.001$ pour tous les types de données (coefficients CLSP et différents niveaux de bruit de quantification dans les quantificateurs) et pour toutes les dimensions du dictionnaire (taille 2,$\cdots$,16384 ; longueurs des vecteurs 4,$\cdots$,10). Ainsi, la mise en œuvre de l'algorithme RGSKAε est simple, elle ne demande pas la détermination expérimentale d'aucun de ces paramètres.

5.5 Version rapide

Dans [4], nous avons proposé un algorithme de recherche rapide pour trouver le plus proche mot de code au vecteur LSP[4] ou CLSP[5] dans le cas du codage de la parole à débit réduit. Nous avons ordonné les mots de code du dictionnaire selon le facteur d'ordre $v = \sum_{j=1}^{p} x_j$ et, au cours de codage, nous avons cherché l'indice du mot de code ayant le facteur d'ordre le plus proche de celui du vecteur à coder. Ce mot de code central de l'indice i et son voisinage dans le dictionnaire ordonné ont formé la zone de recherche de longueur l souhaitée.

Nous avons utilisé cette méthode de recherche rapide basée sur la somme de composantes des vecteurs dans l'algorithme proposé RGSKAε. Cela a deux avantages : nous réduisons considérablement le temps nécessaire pour la construction et

[4]LSP :Line Spectrum Pairs

[5]CLSP :Cosine of Line Spectrum Pairs

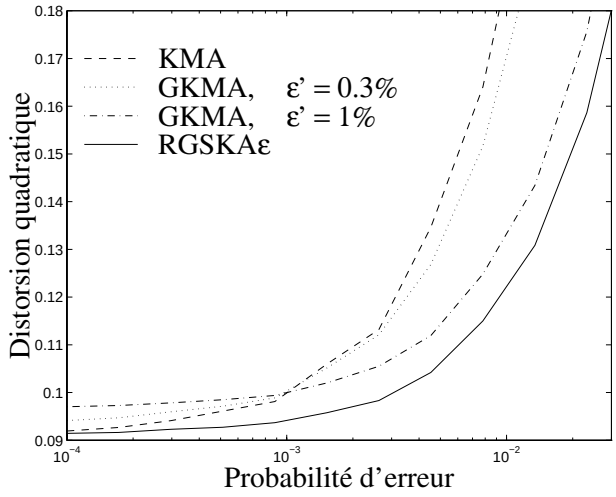

FIG. 1 — Distorsion euclidienne quadratique en fonction de la probabilité d'erreur de transmission. Dictionnaires CLSP de 256 mots de code

de plus, nous optimisons le dictionnaire pour la recherche rapide tout en gardant les avantages de l'algorithme RGSKAε [2].

6 Résultats de simulations

La figure 1 permet de comparer les performances des l'algorithmes KMA, GKMA et RGSKAε sur un canal bruité, dans le cadre de la quantification vectorielle des coefficients CLSP de l'ordre 10. Les dictionnaires de 256 mots de code ont été construits à partir d'une base d'apprentissage de 25600 vecteurs. Nous avons codé la séquence de test de 35000 vecteurs en simulant la transmission des indices via un canal binaire symétrique. Pendant cette simulation, nous avons utilisé la distance euclidienne quadratique. Sur la figure 1, nous avons tracé la distorsion euclidienne quadratique en fonction de la probabilité d'erreur de transmission.

Nos observations sont les suivantes :

- Dans le cas de l'absence de bruit de transmission, les performances du dictionnaire de l'algoritme RGSKAε sont aussi bonnes que celles de l'algorithme KMA.
- L'algoritme RGSKAε donne une distorsion plus faible que l'algorithme GKMA même dans le cas où la probabilité d'erreur du canal correspond à celle prévue à la construction de dictionnaire de l'algorithme GKMA.

Malgré que les dictionnaires de l'algorithme GKMA satisfont les deux conditions nécessaries d'optimalité, la solution obtenue est un minimum local qui est très loin de l'optimum global.

Lorsqu le canal est stationnaire et sa probabilité d'erreur sur un bit ε_c est connue lors de la construction du dictionnaire, on peut mieux approcher l'optimum global en utilisant l'algoritme RGSKAε et en choisissant $\gamma = \varepsilon_c$.

7 Conclusions

Nous avons développé un nouvel algorithme de codage conjoint source-canal, l'algorithme RGSKAε. Cet algorithme n'est pas influencé par le dictionnaire initial et il est capable de sortir des minima locaux. En diminuant progressivement la probabilité d'erreur supposée de la transmission au cours de l'apprentissage, pour un grand nombre de canaux (bruité ou non), les performances du nouvel algorithme sont supérieures à celles des précédents algorithmes étudiés.

En tenant compte des statistiques du canal seulement pendant la réactualisation des mots de code du dictionnaire au cours de l'apprentissage, la complexité de l'algorithme RGSKAε est devenue comparable à celle de l'algorithme KMA et donc beaucoup moins complexe que l'algorithme GKMA. Nous avons pu encore diminuer sa complexité sans perte de performance en utilisant une technique de codage rapide basée sur la somme des composantes des vecteurs au cours de la construction du dictionnaire.

Grâce à ces propriétés de convergence avantageuse, l'algorithme RGSKAε permet de construire un dictionnaire bien équilibré même si le nombre des vecteurs d'apprentissage est inférieur à la taille du dictionnaire. Sa mise en œuvre est aussi simple que celle de l'algorithme KMA car tous ses paramètres sont fixés, ils ne dépendent pas du type ou de la taille des vecteurs à quantifier.

Références

[1] N. FARVARDIN. A study of vector quantization for noisy channels. *IEEE Trans. on information theory*, 36(4) :799–809, July 1990.

[2] B. KÖVESI. *Quantification vectorielle des paires de raies spectrales pour la compression de la parole à débit réduit. Application à la téléphonie numérique sur canal acoustique sous-marin*. Thèse de doctorat, Université de Rennes I, Janvier 1997.

[3] B. KÖVESI, S. SAOUDI, J.M. BOUCHER, and Z. REGULY. A fast robust stochastic algorithm for vector quantizer design for nonstationary channels. In *Proc. ICASSP-95, IEEE International Conference of Acoustics, Speech, and Signal Processing*, volume 1, pages 269–273, Detroit, Michigan, USA, May 1995.

[4] B. KÖVESI, S. SAOUDI, J.M. BOUCHER, and Z. REGULY. Réduction de la complexité de la quantification vectorielle des coefficients LSP pour le codage de la parole. In *Proc. GRETSI-95, Quinzième Colloque sur le Traitement du Signal et des Images*, volume 1, pages 353–356, Juan-les-pins, France, Septembre 1995.