Proceedings of the VI Brazilian Conference on Neural Networks - VI Congresso Brasileiro de Redes Neurais
pp. 37–42, June 2–5, 2003 - Centro Universitário da FEI, São Paulo - SP - Brazil

Aprendizagem Competitiva versus Algoritmo LBG Quanto à Complexidade Computacional

Francisco Madeiro[1], Waslon T. A. Lopes[2], Benedito G. Aguiar Neto[2], Marcelo S. Alencar[2]
[1]Universidade Católica de Pernambuco – Recife, PE, Brasil
[2]Universidade Federal de Campina Grande – Campina Grande, PB, Brasil*
madeiro@dei.unicap.br, {waslon,bganeto,malencar}@dee.ufcg.edu.br

Abstract

Codebook design plays a crucial role in the performance of signal processing systems based on vector quantization (VQ). This paper is devoted to the comparison of a competitive learning algorithm with the traditional LBG (Linde-Buzo-Gray) algorithm in terms of computational complexity. From analytical expressions for the number of operations performed by those algorithms for VQ codebook design, the authors establish the conditions that may be satisfied so that the competitive algorithm be more efficient than the LBG algorithm in terms of the number of operations (multiplications, divisions, additions, subtractions and comparisons) performed.

1. Introdução

A quantização vetorial (QV) [1, 2] desempenha um papel importante em diversos sistemas de processamento de sinais. Uma questão central em QV é a representação adequada de uma fonte de informação por meio de um dicionário de tamanho finito. Em muitos casos, a função densidade de probabilidade da fonte não é conhecida *a priori*, mas um conjunto de treinamento $\boldsymbol{X} = \{\boldsymbol{x}_m \in \mathbb{R}^K : 1 \leq m \leq M\}$ é disponível. O projeto de dicionário lida com a tarefa de encontrar um dicionário $W = \{\boldsymbol{w}_i \in \mathbb{R}^K : 1 \leq i \leq N\}$ de vetores-código (também denominados vetores de reconstrução, protótipos, padrões de reconstrução, padrões de referência) $\boldsymbol{w}_i$ que seja o mais representativo possível do conjunto de vetores de treino $\boldsymbol{x}_m$.

Em quantização vetorial, um determinado vetor de entrada $\boldsymbol{x}$ é representado pela versão quantizada $\boldsymbol{w}_i = Q(\boldsymbol{x})$ se $d(\boldsymbol{x}, \boldsymbol{w}_i) < d(\boldsymbol{x}, \boldsymbol{w}_j), \forall j \neq i$, isto é se $\boldsymbol{w}_i$ for o vetor-código mais semelhante a $\boldsymbol{x}$ (de acordo com uma medida de distorção $d(\cdot)$, como por exemplo a distância euclidiana) dentre todos os vetores-código do dicionário.

Em se tratando de compressão de sinais baseada em QV, o alvo a ser alcançado é a obtenção de um dicionário ótimo, tal que a distorção introduzida ao se representarem os vetores do sinal por suas correspondentes versões quantizadas (isto é, pelos correspondentes vetores-código) seja minimizada. A taxa de codificação de um quantizador vetorial, expressa em bit/amostra em codificação de forma de onda de voz e em bit por *pixel* (bpp) em codificação de imagem, é dada por $R = \frac{1}{K}\log_2 N$, em que K é a dimensão dos vetores-códigos e N é o número de níveis do quantizador vetorial (tamanho do dicionário).

O algoritmo LBG [3] é a técnica mais utilizada para projeto de dicionários. Também conhecido como algoritmo de Lloyd generalizado (GLA, *generalized Lloyd algorithm*), o algoritmo LBG procura satisfazer as duas condições necessárias para a obtencão de um dicionário ótimo [2]: a condição de vizinho mais próximo para a partição do espaço de padrões introduzida pela QV e a condição de centróide para os vetores-código. No algoritmo LBG, a distorção introduzida ao se representarem os vetores do conjunto de treinamento pelos correspondentes vetores-códigos é monitorada a cada iteração. A regra de parada (teste de convergência) do algoritmo baseia-se nessa distorção monitorada – o treinamento do dicionário é encerrado quando $(D_{n-1} - D_n)/D_n \leq \epsilon$, em que ϵ denota um limiar de distorção pré-estabelecido e D_n denota a distorção total ao final da n-ésima iteração.

Algoritmos de aprendizagem não-supervisionada têm sido utilizados para projeto de dicionários [4–10]. Nessas abordagens, os pesos sinápticos dos neurônios correspondem às componentes dos vetores-código do dicionário a ser projetado. Em se tratando de aprendizagem competitiva simples aplicada ao projeto de dicionário, a cada apresentação de vetor de treino, os pesos sinápticos do *vencedor* (isto é, vetor-peso mais próximo do vetor de treino, ou seja, vetor-código de maior similaridade com o vetor de treino) são atualizados; os demais vetores-peso não são atualizados. Na aprendizagem de Kohonen [4, 5], o *vencedor* e uma vizinhança de vetores (em um arranjo topológico de nós) são atualizados na direção do vetor de treino.

No presente trabalho é apresentada uma avaliação comparativa de complexidade de um algoritmo de aprendizagem competitiva e do algoritmo LBG. São obtidas expressões analíticas (em função de K, N, M e do número de iterações desses algoritmos) que estabelecem as condições que devem ser obedecidas para que o algoritmo competitivo seja mais eficiente que o algoritmo LBG quanto à questão de complexidade computa-

*Os autores gostariam de expressar os agradecimentos à CAPES e ao CNPq pelo apoio financeiro ao trabalho.

cional, avaliada por meio da contabilização do número de divisões, multiplicações, comparações, adições e subtrações realizadas para o treinamento de dicionários. Os resultados apresentados ao final do artigo mostram que, em simulações envolvendo codificação de sinal com distribuição de Gauss-Markov, essas condições são obedecidas.

2. Algoritmo Competitivo

O algoritmo competitivo considerado neste trabalho é descrito a seguir.

Algoritmo Competitivo (AC):

Para $1 \leq n \leq n_{\mathrm{AC}}$

Para $1 \leq m \leq M_{\mathrm{vet}}$

Determine o vencedor $\boldsymbol{w}_{i^*}(n,m)$:

$$i^* = \arg\min_i d[\boldsymbol{x}(m), \boldsymbol{w}_i(n,m)]$$

Atualize o vencedor de acordo com

$$\tilde{w}_{i^*j}(n,m) = w_{i^*j}(n,m) + \Delta w_{i^*j}(n,m), \quad (1)$$

em que

$$\Delta w_{i^*j}(n,m) = \eta(n)[(x_j(m) - w_{i^*j}(n,m)]. \quad (2)$$

Na descrição acima, $\boldsymbol{x}(m)$ é o m-ésimo vetor do conjunto de treino, enquanto $\boldsymbol{w}_i(n,m)$ e $\boldsymbol{w}_{i^*}(n,m)$ denotam, respectivamente, o i-ésimo vetor-código e o vencedor quando da apresentação do m-ésimo vetor de treino na n-ésima iteração. Por sua vez,

$$d[\boldsymbol{x}(m), \boldsymbol{w}_i(n,m)] = \sum_{j=1}^{K} [x_j(m) - w_{ij}(n,m)]^2 \quad (3)$$

denota a distância euclidiana quadrática entre os vetores $\boldsymbol{x}(m)$ e $\boldsymbol{w}_i(n,m)$, em que $x_j(m)$ é a j-ésima componente do vetor $\boldsymbol{x}(m)$ e $w_{ij}(n,m)$ é a j-ésima componente do vetor $\boldsymbol{w}_i(n,m)$. Na expressão que descreve a atualização do vencedor, Δw_{i^*j} é a modificação introduzida na j-ésima componente do vencedor, $\eta(n)$ é a taxa de aprendizagem ou ganho de adaptação na n-ésima iteração (com $0 < \eta(n) < 1$), w_{i^*j} é a j-ésima componente do vencedor e $\tilde{w}_{i^*j}$ é a versão atualizada da j-ésima componente do vencedor.

A função taxa de aprendizagem decresce linearmente com n, permanecendo constante ao longo de cada iteração.

O algoritmo AC apresenta 5 parâmetros ajustáveis: dimensão do dicionário (K); número de vetores-código (N); número total de iterações (n_{AC}); taxa de aprendizagem inicial ($\eta(1)$); taxa de aprendizagem final ($\eta(n_{\mathrm{AC}})$).

3. Número de operações dos Algoritmos LBG e AC

Em [11], foram obtidas expressões para o número de operações (multiplicações, subtrações, adições, divisões e comparações) dos algoritmos LBG e AC, em função da dimensão (K), do tamanho (N) do dicionário, do número de vetores de treino (M) e do número de iterações dos algoritmos. A Tabela 1 apresenta o número total de operações dos algoritmos LBG e AC, em que n_{LBG} e n_{AC} denotam o número de iterações dos algoritmos LBG e AC, respectivamente.

A Tabela 1 mostra que, em termos de número de divisões, o algoritmo AC é mais eficiente que o algoritmo LBG. De fato, o algoritmo AC executa apenas uma operação de divisão.

4. Condições para as quais AC é mais Eficiente que LBG em Termos de Operações Realizadas

Nesta seção, são determinadas as condições para as quais o algoritmo AC requer um número de multiplicações, subtrações, adições e comparações inferior ao requerido pelo algoritmo LBG. Cada uma destas operações é considerada (por meio dos resultados apresentados na Tabela 1) separadamente, conforme segue.

4.1 Número de Multiplicações

Para que o número de multiplicações do algoritmo AC seja menor que o número de multiplicações do algoritmo LBG, deve-se ter

$$[1 + (1+N)KM]n_{\mathrm{AC}} - 1 < KNMn_{\mathrm{LBG}}, \quad (4)$$

ou seja,

$$n_{\mathrm{AC}} < \frac{1 + KNMn_{\mathrm{LBG}}}{1 + (1+N)KM}. \quad (5)$$

Em projetos típicos de dicionários, são utilizados grandes conjuntos de treino (elevados valores de M), de tal forma que $KNMn_{\mathrm{LBG}} \gg 1$ e $(1+N)KM \gg 1$. Assim, (5) se reduz a

$$n_{\mathrm{AC}} < \frac{N}{1+N} n_{\mathrm{LBG}}. \quad (6)$$

4.2 Número de Subtrações

Para que o número de subtrações do algoritmo AC seja menor que o número de subtrações do algoritmo LBG, deve-se ter

$$[1 + (1+N)KM]n_{\mathrm{AC}} + 1 < (1 + KNM)n_{\mathrm{LBG}}, \quad (7)$$

ou seja,

$$n_{\mathrm{AC}} < \frac{(1 + KNM)n_{\mathrm{LBG}} - 1}{1 + (1+N)KM}. \quad (8)$$

Tabela 1: Número total de operações requeridas pelos algoritmos LBG e AC.

Operação	Algoritmo	
	LBG	AC
Multiplicação	$KNMn_{\mathrm{LBG}}$	$[1+(1+N)KM]n_{\mathrm{AC}}-1$
Subtração	$(1+KNM)n_{\mathrm{LBG}}$	$[1+(1+N)KM]n_{\mathrm{AC}}+1$
Adição	$[(1+KN)(M-1)+(K-N+1)M]n_{\mathrm{LBG}}$	$[1+(K-1)NM+KM]n_{\mathrm{AC}}-1$
Divisão	$(1+KN)n_{\mathrm{LBG}}$	1
Comparação	$[1+(N-1)M]n_{\mathrm{LBG}}$	$(N-1)Mn_{\mathrm{AC}}$

Em projetos típicos de dicionários (em que são utilizados grandes conjuntos de treino, o que implica valores elevados de M), $(1+KNM)n_{\mathrm{LBG}} \gg 1$ e $(1+N)KM \gg 1$, de modo que (8) se reduz a

$$n_{\mathrm{AC}} < \frac{(1+KNM)n_{\mathrm{LBG}}}{(1+N)KM}. \quad (9)$$

Tendo em vista que $1+KNM \approx KMN$, (9) se reduz a

$$n_{\mathrm{AC}} < \frac{N}{1+N}n_{\mathrm{LBG}}. \quad (10)$$

4.3 Número de Adições

Para que o número de adições do algoritmo AC seja menor que o número de adições do algoritmo LBG, deve-se ter

$$[1+(K-1)NM+KM]n_{\mathrm{AC}}-1 < [(1+KN)(M-1)+(K-N+1)M]n_{\mathrm{LBG}}, \quad (11)$$

ou seja,

$$n_{\mathrm{AC}} < \frac{1+[(1+KN)(M-1)+(K-N+1)M]n_{\mathrm{LBG}}}{1+(K-1)NM+KM}. \quad (12)$$

Considerando projetos típicos de dicionários (isto é, considerando elevados valores de M), segue que $[(1+KN)(M-1)+(K-N+1)M]n_{\mathrm{LBG}} \gg 1$ e $(K-1)NM+KM \gg 1$, de modo que (12) se reduz a

$$n_{\mathrm{AC}} < \frac{[(1+KN)(M-1)+(K-N+1)M]n_{\mathrm{LBG}}}{(K-1)NM+KM}. \quad (13)$$

Tendo em vista que $M-1 \approx M$, (13) se reduz a

$$n_{\mathrm{AC}} < \frac{[(1+KN)M+(K-N+1)M]n_{\mathrm{LBG}}}{(K-1)NM+KM}. \quad (14)$$

Após algumas manipulações, (14) se reduz a

$$n_{\mathrm{AC}} < \frac{[(K-1)(1+N)+3]}{(K-1)N+K}n_{\mathrm{LBG}}. \quad (15)$$

4.4 Número de Comparações

Para que o número de comparações do algoritmo AC seja menor que o número de comparações do algoritmo LBG, deve-se ter

$$(N-1)Mn_{\mathrm{AC}} < [1+(N-1)M]n_{\mathrm{LBG}}, \quad (16)$$

ou seja,

$$n_{\mathrm{AC}} < \frac{[1+(N-1)M]n_{\mathrm{LBG}}}{(N-1)M}. \quad (17)$$

Como $1+(N-1)M \approx (N-1)M$ em projetos típicos de dicionários (ou seja, para elevados valores de M), a condição (17) se reduz a

$$n_{\mathrm{AC}} < n_{\mathrm{LBG}}. \quad (18)$$

4.5 Considerações Gerais

Considere as Inequações (6) e (10), referentes, respectivamente, ao número de multiplicações e subtrações. À medida que N aumenta, tem-se que $\frac{N}{1+N}$ tende a 1. Deste modo, à medida que N aumenta, as condições (6) e (10) tendem a se reduzir siplesmente à condição $n_{\mathrm{AC}} < n_{\mathrm{LBG}}$. Considere agora o pior caso, ou seja, o menor valor possível de N, isto é, $N = 1$. Para este valor de N as condições (6) e (10) são satisfeitas para

$$n_{\mathrm{AC}} < \frac{n_{\mathrm{LBG}}}{2}. \quad (19)$$

Portanto, é lícito afirmar que, uma vez satisfeita a condição (19), o número de multiplicações e subtrações realizadas pelo algoritmo AC é inferior ao número de multiplicações e subtrações do algoritmo LBG.

Considere a Inequação (15). Para $K \geq 1$ e $N \geq 1$, conforme se pode observar na Figura 1, segue que

$$\frac{[(K-1)(1+N)+3]}{(K-1)N+K} > 1, \quad (20)$$

de modo que

$$\frac{[(K-1)(1+N)+3]}{(K-1)N+K}n_{\mathrm{LBG}} > n_{\mathrm{LBG}}. \quad (21)$$

Assim, uma vez assegurado que

$$n_{\mathrm{AC}} < n_{\mathrm{LBG}}, \quad (22)$$

a condição (15) é automaticamente satisfeita. Portanto, uma vez satisfeita a condição (22), o número de adições (como também o número de comparações[1]) realizadas pelo algoritmo AC é inferior ao número de adições (comparações) do algoritmo LBG.

[1]Conforme mostra a Inequação (18).

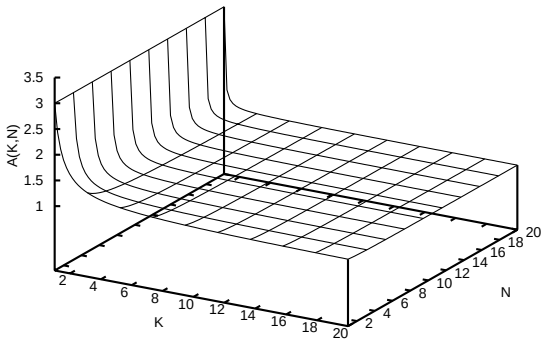

Figura 1: Gráfico de $A(K,N) = \frac{[(K-1)(1+N)+3]}{(K-1)N+K}$.

Tendo em vista que a condição (19) é mais severa que a condição (22) é lícito afirmar que a condição (19) deve ser satisfeita para que o número de operações do algoritmo AC seja inferior ao número de operações do algoritmo LBG.

Conforme mostra a Tabela 2, que apresenta resultados obtidos em [11], a condição (19) foi satisfeita nas simulações concernentes ao projeto de dicionários aplicados à QV de imagens. A tabela mostra que o algoritmo AC necessitou de um número de iterações menor que metade do número de iterações executadas pelo algoritmo LBG.

Tabela 2: Número de iterações realizadas pelos algoritmos LBG e AC para $K = 16$ (QV de imagem usando blocos de 4×4 pixels) e alguns valores de N (com a correspondente taxa de codificação R).

K	N	R	Iterações	
			LBG	AC
16	32	0,3125 bpp	18	3
16	64	0,375 bpp	13	3
16	128	0,4375 bpp	17	3
16	64	0,375 bpp	15	4
16	128	0,4375 bpp	15	5

Na seção a seguir são apresentados resultados concernentes ao projeto de dicionários aplicados à codificação de sinal com distribuição de Gauss-Markov [12, 13] com coeficiente de correlação igual a 0,8.

5. Resultados

Todos os projetos de dicionários foram realizados utilizando um conjunto de treino constituído de 120.000 amostras. Foram projetados dicionários com diversos valores de dimensão (K) e número de níveis (N). Em se tratando do projeto de dicionários com o algoritmo LBG, utilizou-se um limiar de distorção ϵ igual a 0,001 e foram testados 5 dicionários iniciais diferentes para cada combinação de K e N avaliada.

A Tabela 3 mostra que a velocidade de convergência (número de iterações) do algoritmo LBG depende fortemente do dicionário inicial utilizado. No algoritmo AC, por outro lado, o número de iterações é especificado *a priori*, como um parâmetro do algoritmo.

A diferença de desempenho entre os algoritmos LBG e AC, em termos de relação sinal-ruído (SNR, *signal to noise ratio*) [12] do sinal reconstruído, é apresentada na Tabela 4 – esta diferença em decibéis (dB) é denotada por $\mathrm{SNR_{AC}} - \mathrm{SNR_{LBG}}$. A tabela também mostra o número de iterações realizadas pelos algoritmos LBG e AC nas condições (parâmetros, inicialização) que levaram aos melhores dicionários para cada combinação de K e N. Conforme se observa na Tabela 4, para os valores de K e N considerados, o algoritmo AC necessitou de um menor número de iterações para produzir dicionários com praticamente a mesma qualidade dos dicionários LBG, tendo em vista que na Tabela são observados pequenos valores de $\mathrm{SNR_{AC}}$-$\mathrm{SNR_{LBG}}$. Em outras palavras, apesar de serem obtidos após um número de iterações menor que metade do número de iterações (passagens completas do conjunto de treino) realizadas pelo algoritmo LBG, os dicionários AC em geral levaram a sinais reconstruídos com praticamente o mesmo valor de SNR dos sinais reconstruídos com uso de dicionários LBG. De fato, a maior diferença pró AC observada na Tabela 4 foi de 0,09 dB, enquanto que a maior diferença pró LBG observada na Tabela 4 foi de 0,03 dB.

Tabela 3: Sensibilidade do algoritmo LBG a cinco dicionários iniciais diferentes (D1, D2, D3, D4 e D5) em termos de número de iterações (n_{LBG}).

K	N	R	n_{LBG}				
			D1	D2	D3	D4	D5
2	16	2,0	32	21	21	33	21
2	32	2,5	46	45	23	38	30
2	64	3,0	83	40	35	58	40
4	16	1,0	24	29	26	16	14
4	32	1,25	29	16	15	23	14
4	64	1,5	35	20	21	27	22
3	8	1,0	21	10	24	14	20
5	32	1,0	26	22	19	18	15
8	256	1,0	32	15	16	21	16

O último conjunto de avaliações realizadas neste trabalho teve como objetivo comparar o número de operações (multiplicações, subtrações, adições, divisões e comparações) realizadas pelos algoritmos LBG e AC quado aplicados ao projeto de dicionários destinados à codificação do sinal com distribuição de Gauss-Markov. As Tabelas 5 e 6 mostram que, para todos os valores de K e N (com a correspondente taxa de codificação R, expressa em bit/amostra) considerados, o algoritmo AC realiza um número de operações inferior ao número de

Tabela 4: Diferença de desempenho dos algoritmos LBG e AC em termos de relação sinal-ruído (SNR) do sinal reconstruído.

K	N	$\mathrm{SNR_{AC}} - \mathrm{SNR_{LBG}}$	Iterações	
			LBG	AC
2	8	0,00	14	2
2	16	0,00	21	2
2	32	0,09	46	2
2	64	0,08	83	3
2	128	0,06	51	4
2	256	0,00	50	5
4	8	-0,03	13	2
4	16	-0,02	14	2
4	32	0,03	16	2
4	64	0,04	20	3
4	128	0,01	19	4
4	256	0,01	18	4
3	8	-0,01	10	2
5	32	-0,01	18	2
6	64	0,01	16	3
8	256	0,01	15	5

operações realizadas pelo algoritmo LBG.

Finalmente, cumpre mencionar que a condição (19) também é satisfeita em se tratando de projeto de dicionários aplicados à codificação de forma de onda de voz. Para essa aplicação, os dicionários AC levaram, em geral, a sinais de voz reconstruídos com qualidade, (avaliada por meio da relação sinal-ruído segmental) próxima ou ligeiramente superior à apresentada pelos sinais reconstruídos com uso de dicionários LBG.

6. Conclusões

Este trabalho apresentou um abordagem da complexidade computacional de um algoritmo competitivo e do algoritmo LBG em projeto de dicionários para quantização vetorial. A abordagem foi desenvolvida com base em expressões analíticas para o número de operações (multiplicações, subtrações, adições, divisões e comparações) realizadas pelo algoritmo competitivo (AC) e pelo algoritmo LBG no projeto de dicionários. Mostrou-se que, para dimensão (K) e tamanho (N) de dicionário determinados, o algoritmo AC é mais eficiente que o algoritmo LBG em termos de número de multiplicações e subtrações caso a condição $n_{\mathrm{AC}} < \frac{n_{\mathrm{LBG}}}{2}$ seja satisfeita, ou seja, caso o número de iterações do algoritmo AC seja menor que a metade do número de iterações do algoritmo LBG. Além disso, foi demonstrado que o algoritmo AC executa um número de adições e comparações inferior ao executado pelo algoritmo LBG sempre que $n_{\mathrm{AC}} < n_{\mathrm{LBG}}$. Ressalte-se que o algoritmo AC requer a execução de apenas uma operação de divisão.

Avaliações realizadas neste trabalho, considerando alguns projetos de dicionários aplicados à codificação de sinal com distribuição de Gauss-Markov, mostraram que a condição $n_{\mathrm{AC}} < \frac{n_{\mathrm{LBG}}}{2}$ é satisfeita e, conseqüentemente, a condição $n_{\mathrm{AC}} < n_{\mathrm{LBG}}$ também é satisfeita. O algoritmo competitivo, deste modo, requer um número de operações inferior ao requerido pelo algoritmo LBG. Observou-se que, realizando um número de operações inferior ao apresentado pelo algoritmo LBG, ainda assim o algoritmo competitivo produziu dicionários que levaram a sinais reconstruídos com praticamente os mesmos valores de relação sinal-ruído dos sinais reconstruídos com uso de dicionários LBG.

Referências

[1] R. M. Gray. Vector quantization. *IEEE ASSP Magazine*, pages 4–29, April 1984.

[2] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Boston, MA, 1992.

[3] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1):84–95, January 1980.

[4] T. Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, 3 edition, 1989.

[5] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, September 1990.

[6] S. Carrato. Image vector quantization using ordered codebooks: Properties and applications. *Signal Processing*, 40:87–103, 1994.

[7] A. K. Krishnamurthy, S. C. Ahalt, D. E. Melton, and P. Chen. Neural networks for vector quantization of speech and images. *IEEE Journal on Selected Areas in Communications*, 8(8):1449–1457, October 1990.

Tabela 5: Número de operações requeridas pelo algoritmo LBG ao serem projetados dicionários destinados à codificação de sinal com distribuição de Gauss-Markov.

K	N	R	n_{LBG}	mult.	sub.	ad.	div.	comp.
2	8	1,5	14	$1,34 \cdot 10^7$	$1,34 \cdot 10^7$	$1,01 \cdot 10^7$	$2,38 \cdot 10^2$	$5,88 \cdot 10^6$
2	16	2,0	21	$4,03 \cdot 10^7$	$4,03 \cdot 10^7$	$2,52 \cdot 10^7$	$6,93 \cdot 10^2$	$1,89 \cdot 10^7$
2	128	3,5	51	$7,83 \cdot 10^8$	$7,83 \cdot 10^8$	$4,04 \cdot 10^8$	$1,31 \cdot 10^4$	$3,89 \cdot 10^8$
2	256	4,0	50	$1,54 \cdot 10^9$	$1,54 \cdot 10^9$	$7,80 \cdot 10^8$	$2,56 \cdot 10^4$	$7,65 \cdot 10^8$
4	8	0,75	13	$1,25 \cdot 10^7$	$1,25 \cdot 10^7$	$1,17 \cdot 10^7$	$4,29 \cdot 10^2$	$2,73 \cdot 10^6$
4	16	1,0	14	$2,69 \cdot 10^7$	$2,69 \cdot 10^7$	$2,27 \cdot 10^7$	$9,10 \cdot 10^2$	$6,30 \cdot 10^6$
4	128	1,75	19	$2,92 \cdot 10^8$	$2,92 \cdot 10^8$	$2,22 \cdot 10^8$	$9,75 \cdot 10^3$	$7,24 \cdot 10^7$
4	256	2,0	18	$5,53 \cdot 10^8$	$5,53 \cdot 10^8$	$4,18 \cdot 10^8$	$1,84 \cdot 10^4$	$1,38 \cdot 10^8$
5	32	1,0	18	$6,91 \cdot 10^7$	$6,91 \cdot 10^7$	$5,83 \cdot 10^7$	$2,90 \cdot 10^3$	$1,34 \cdot 10^7$
8	256	1,0	15	$4,61 \cdot 10^8$	$4,61 \cdot 10^8$	$4,05 \cdot 10^8$	$3,07 \cdot 10^4$	$5,74 \cdot 10^7$

Tabela 6: Número de operações requeridas pelo algoritmo AC ao serem projetados dicionários destinados à codificação de sinal com distribuição de Gauss-Markov.

K	N	R	n_{AC}	mult.	sub.	ad.	div.	comp.
2	8	1,5	2	$2,16 \cdot 10^6$	$2,16 \cdot 10^6$	$1,20 \cdot 10^6$	1	$8,40 \cdot 10^5$
2	16	2,0	2	$4,08 \cdot 10^6$	$4,08 \cdot 10^6$	$2,16 \cdot 10^6$	1	$1,80 \cdot 10^6$
2	128	3,5	4	$6,19 \cdot 10^7$	$6,19 \cdot 10^7$	$3,12 \cdot 10^7$	1	$3,05 \cdot 10^7$
2	256	4,0	5	$1,54 \cdot 10^8$	$1,54 \cdot 10^8$	$7,74 \cdot 10^7$	1	$7,65 \cdot 10^7$
4	8	0,75	2	$2,16 \cdot 10^6$	$2,16 \cdot 10^6$	$1,68 \cdot 10^6$	1	$4,20 \cdot 10^5$
4	16	1,0	2	$4,08 \cdot 10^6$	$4,08 \cdot 10^6$	$3,12 \cdot 10^6$	1	$9,00 \cdot 10^5$
4	128	1,75	4	$6,19 \cdot 10^7$	$6,19 \cdot 10^7$	$4,66 \cdot 10^7$	1	$1,52 \cdot 10^7$
4	256	2,0	4	$1,23 \cdot 10^8$	$1,23 \cdot 10^8$	$9,26 \cdot 10^7$	1	$3,06 \cdot 10^7$
5	32	1,0	2	$7,92 \cdot 10^6$	$7,92 \cdot 10^6$	$6,38 \cdot 10^6$	1	$1,49 \cdot 10^6$
8	256	1,0	5	$1,54 \cdot 10^8$	$1,54 \cdot 10^8$	$1,35 \cdot 10^8$	1	$1,91 \cdot 10^7$

[8] O. T.-C. Chen, B. J. Sheu, and W.-C. Fang. Image compression using self-organization networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 4(5):480–489, October 1994.

[9] C. Zhu and L. M. Po. Partial distortion sensitive competitive learning algorithm for optimal codebook design. *Electronics Letters*, 32(19):1757–1758, September 1996.

[10] E. Yair, K. Zeger, and A. Gersho. Competitive learning and soft competition for vector quantizer design. *IEEE Transactions on Signal Processing*, 40(2):294–309, February 1992.

[11] F. Madeiro, W. T. A. Lopes, B. G. Aguiar Neto, and M. S. Alencar. Complexidade computacional de um algoritmo competitivo aplicado ao projeto de quantizadores vetoriais. *Anais do VI Congresso Brasileiro de Redes Neurais (CBRN'03)*, pages 43–48, 2003.

[12] N. S. Jayant and P. Noll. *Digital Coding of Waveforms*. Prentice-Hall, Englewood Cliffs, NJ, 1984.

[13] R. M. Gray and Y. Linde. Vector quantizers and predictive quantizers for Gauss-Markov sources. *IEEE Transactions on Communications*, 30(2):381–389, February 1982.