Learning and Nonlinear Models - Revista da Sociedade Brasileira de Redes Neurais (SBRN), Vol. 2, No. 1, pp. 34–48, 2004

COMPLEXIDADE COMPUTACIONAL DE UM ALGORITMO COMPETITIVO APLICADO AO PROJETO DE QUANTIZADORES VETORIAIS

Francisco Madeiro
Departamento de Estatística e Informática
Universidade Católica de Pernambuco – Recife, PE, Brasil
madeiro@dei.unicap.br

Waslon Terllizzie Araújo Lopes
Departamento de Engenharia Elétrica
ÁREA1 – Faculdade de Ciência e Tecnologia – Salvador, BA, Brasil
waslon@area1.br

Benedito Guimarães Aguiar Neto, Marcelo Sampaio de Alencar
Departamento de Engenharia Elétrica
Universidade Federal de Campina Grande – Campina Grande, PB, Brasil
{bganeto,malencar}@dee.ufcg.edu.br

Resumo – O projeto de dicionários tem um papel crucial para o bom desempenho de sistemas de processamento de sinais baseados em quantização vetorial (QV). Neste trabalho é investigada a complexidade computacional de um algoritmo competitivo aplicado ao projeto de dicionários. São obtidas expressões analíticas (em função do tamanho do dicionário, da dimensão dos seus vetores-código, do número de vetores do conjunto de treino e do número de iterações realizadas) para o número de operações (divisões, multiplicações, comparações, adições e subtrações) realizadas pelo algoritmo competitivo bem como pelo algoritmo LBG (Linde-Buzo-Gray). A partir dessas expressões analíticas, são estabelecidas as condições que devem ser obedecidas para que o algoritmo competitivo seja mais eficiente que o algoritmo LBG no que diz respeito a cada uma das operações realizadas. Simulações referentes ao projeto de dicionários aplicados à codificação de imagem e à codificação de sinal com distribuição de Gauss-Markov corroboram as expressões analíticas obtidas.

Palavras-chave – Quantização vetorial, algoritmo competitivo, algoritmo LBG, complexidade computacional, codificação de imagens.

Abstract – Codebook design plays a crucial role in the performance of signal processing systems based on vector quantization (VQ). In the present paper, the computational complexity of a competitive learning algorithm applied to VQ codebook design is investigated. Analytical expressions (as a function of the codebook size, the dimension of the codevectors, the number of training vectors and the number of iterations performed) are derived for the number of operations (multiplications, divisions, additions, subtractions and comparisons) performed by the competitive algorithm. Analytical expressions are also derived for the LBG (Linde-Buzo-Gray) algorithm. ¿From the analytical expressions for the number of operations performed by those algorithms, the authors establish the conditions that may be satisfied so that the competitive algorithm be more efficient than the LBG algorithm concerning each operation performed. Simulations regarding image coding as well as coding of signal with Gauss-Markov distribution corroborate the analytical expressions derived.

Keywords – Vector quantization, competitive algorithm, algorithm LBG, computational complexity, image coding.

1. INTRODUÇÃO

A compressão de sinais, cujo objetivo fundamental é reduzir o número de bits necessários para representar adequadamente os sinais (voz, imagem, áudio, vídeo), desempenha um papel importante em aplicações que necessitam minimização dos requisitos de largura de faixa e/ou de capacidade de armazenamento [1, 2], tais como: sistemas multimídia, redes digitais de serviços integrados, videoconferência, sistemas de resposta vocal, telefonia móvel, sistemas de armazenamento de imagens médicas e de impressões digitais e transmissão de imagens de sensoriamento remoto obtidas por satélites. Nesse cenário, a quantização vetorial (QV) apresenta-se como uma técnica adequada, bastante utilizada em diversos sistemas de codificação de sinais.

A quantização vetorial [3, 4], que pode ser vista como uma extensão da quantização escalar em um espaço multidimensional, encontra-se fundamentada na Teoria da Distorção Versus Taxa [5], formulada por Shannon, segundo a qual um melhor desempenho é obtido codificando-se blocos de amostras (isto é, vetores) ao invés de amostras individuais (isto é, escalares). Em outras palavras, essa teoria ressalta a superioridade da quantização vetorial sobre a quantização escalar [6].

Em sistemas de reconhecimento de locutor, um dos méritos da quantização vetorial reside no fato de que essa técnica dispensa a necessidade de alinhamento temporal, uma vez que permite ao sistema de reconhecimento flexibilidade quanto ao tamanho das sentenças proferidas pelos locutores.

O projeto de dicionários desempenha um papel importante para o bom desempenho de sistemas de processamento de sinais baseados em quantização vetorial (QV) [3, 4]. Em codificação de voz e imagem baseada em QV (e.g. [7–14]), a qualidade dos sinais reconstruídos depende dos dicionários projetados. Em sistemas de identificação de locutor que utilizam QV paramétrica (e.g. [15–19]), as taxas de identificação dependem dos dicionários de padrões acústicos de referência projetados para cada locutor cadastrado pelo sistema. Dentre as diversas técnicas para projeto de dicionários, o algoritmo LBG (Linde-Buzo-Gray) [20] destaca-se por sua ampla utilização. Outras abordagens têm sido utilizadas em projeto de dicionários, como por exemplo: algoritmo de Kohonen [21] e outros algoritmos não-supervisionados [22–25]; relaxação estocástica [26]; algoritmos *fuzzy* [27, 28] e algoritmo genético [29].

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 do tamanho do dicionário, da dimensão dos seus vetores-código, do número de vetores do conjunto de treino e do número de iterações realizadas) que estabelecem as condições que devem ser obedecidas para que o algoritmo competitivo (AC) seja mais eficiente que o algoritmo LBG quanto ao número de operações (multiplicações, divisões, adições, subtrações e comparações) envolvidas no projeto de dicionários. Os resultados apresentados ao final do artigo, referentes a simulações envolvendo QV aplicada à codificação de imagem e à codificação de sinal com distribuição de Gauss-Markov, corroboram a avaliação de complexidade computacional dos algoritmos LBG e competitivo por meio meio das expressões obtidas.

O restante deste artigo encontra-se organizado de acordo com as seções a seguir. A Seção 2 apresenta uma abordagem sucinta da quantização vetorial. O algoritmo LBG é descrito na Seção 3. A Seção 4 apresenta uma breve descrição do algoritmo competitivo (AC). Na Seção 5, cujo foco é a complexidade computacional dos algoritmos LBG e AC, são obtidas expressões analíticas para o número de operações realizadas por esses algoritmos. A Seção 6 estabelece as condições que devem ser obedecidas para que o algoritmo AC seja mais eficiente que o algoritmo LBG em termos de número de operações realizadas. Resultados e conclusão são apresentados, respectivamente, nas Seções 7 e 8.

2. QUANTIZAÇÃO VETORIAL

Matematicamente, a quantização vetorial [3, 4] pode ser definida como um mapeamento Q de um vetor de entrada $\boldsymbol{x}$ pertencente ao espaço euclidiano K-dimensional, $\mathbb{R}^K$, em um vetor pertencente a um subconjunto finito W de $\mathbb{R}^K$, ou seja,

$$Q : \mathbb{R}^K \rightarrow W. \tag{1}$$

O dicionário $W = \{\boldsymbol{w}_i;\ i = 1,\ 2,\ \ldots,\ N\}$ é o conjunto de vetores de reprodução (também denominados vetores-código ou vetores de reconstrução), K é a dimensão do quantizador vetorial e N é o tamanho do dicionário, isto é, o número de vetores-código (ou número de níveis, em analogia com a quantização escalar).

O mapeamento Q introduz um particionamento de $\mathbb{R}^K$ em N células (denominadas regiões de Voronoi) $S_i, i = 1,\ 2,\ \ldots,\ N$, tais que

$$\bigcup_{i=1}^{N} S_i = \mathbb{R}^K \text{ e } S_i \cap S_j = \emptyset \text{ para } i \neq j, \tag{2}$$

em que cada célula S_i é definida por

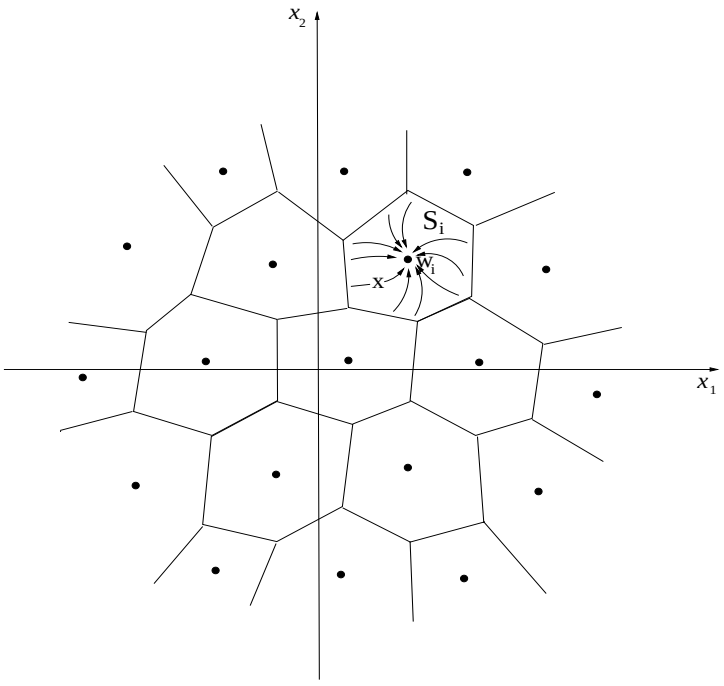
$$S_i = \{\boldsymbol{x} :\ Q(\boldsymbol{x}) = \boldsymbol{w}_i\}. \tag{3}$$

Mais precisamente,

$$S_i = \{\boldsymbol{x} :\ d(\boldsymbol{x}, \boldsymbol{w}_i) < d(\boldsymbol{x}, \boldsymbol{w}_j), \forall j \neq i\}, \tag{4}$$

em que $d(\cdot, \cdot)$ denota uma medida de distância. O vetor-código $\boldsymbol{w}_i$ constitui o vetor representativo de todos os vetores de entrada pertencentes à célula S_i, conforme ilustra a Figura 1. Como a quantização vetorial realiza um mapeamento de padrões de entrada (vetores de entrada $\boldsymbol{x}$) semelhantes em padrões de saída (vetores-código $\boldsymbol{w}_i$) semelhantes, ela pode ser vista como uma forma de reconhecimento de padrões, em que um padrão de entrada é "aproximado" por um padrão de referência, pertencente a um conjunto predeterminado (dicionário) de padrões (vetores-código) de referência [3, 30].

Em um sistema de codificação de sinais baseado em quantização vetorial, conforme apresentado na Figura 2, um quantizador vetorial pode ser visto como a combinação de duas funções: um codificador de fonte e um decodificador de fonte. Dado um vetor $\boldsymbol{x} \in \mathbb{R}^K$ da fonte a ser codificada, o codificador calcula a distorção $d(\boldsymbol{x}, \boldsymbol{w}_i)$ entre o vetor de entrada (vetor a ser quantizado) e cada vetor-código $\boldsymbol{w}_i,\ i = 1,\ 2,\ \ldots,\ N$ do dicionário W. A regra ótima para codificação é a regra do vizinho mais próximo, na qual uma representação binária do índice I, denotada por $\boldsymbol{b}_I$, é transmitida ao decodificador de fonte se o vetor-código $\boldsymbol{w}_I$ corresponder à menor distorção, isto é, se $\boldsymbol{w}_I$ for o vetor-código que apresentar a maior similaridade com $\boldsymbol{x}$ dentre todos os vetores-código do dicionário. Em outras palavras, o codificador usa a regra de codificação $\mathcal{C}(\boldsymbol{x}) = \boldsymbol{b}_I$ se $d(\boldsymbol{x}, \boldsymbol{w}_I) < d(\boldsymbol{x}, \boldsymbol{w}_i),\ \forall i \neq I$. Ao receber a representação binária $\boldsymbol{b}_I$, o decodificador de fonte, que dispõe de uma cópia do dicionário W, simplesmente procura pelo I-ésimo vetor-código e produz o vetor $\boldsymbol{w}_I$ como a reprodução (versão quantizada) de $\boldsymbol{x}$. Em outras palavras, é utilizada a seguinte regra de decodificação: $\mathcal{D}(\boldsymbol{b}_I) = \boldsymbol{w}_I$.



.

Figura 1: Partição do espaço euclidiano bidimensional, $\mathbb{R}^2$, introduzido pelo mapeamento dos vetores de entrada $\boldsymbol{x}$ nos vetores-código $\boldsymbol{w}_i$. As coordenadas x_1 e x_2 representam a primeira e a segunda componentes do vetor $\boldsymbol{x} \in \mathbb{R}^2$, respectivamente.

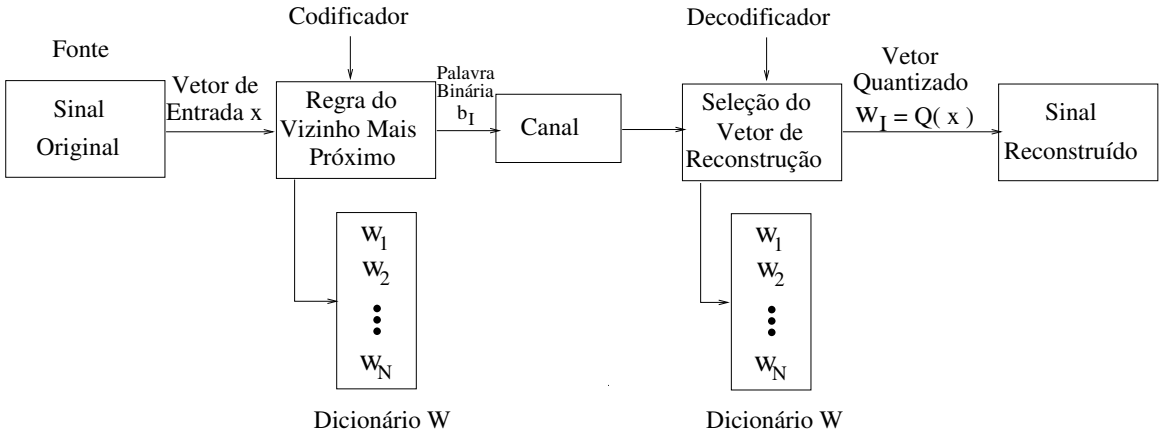


Figura 2: Sistema de codificação baseado em quantização vetorial.

No cenário de codificação digital de sinais, a quantização vetorial, portanto, constitui uma técnica de compressão com perdas, visto que o sinal reconstruído é uma versão degradada do sinal original. O erro médio de quantização ao se representar o sinal de entrada por sua versão quantizada é chamado distorção do quantizador. Por outro lado, a taxa de codificação do quantizador vetorial, que mede o número de bits por componente do vetor, é dada por $R = \frac{1}{K} \log_2 N$. Em codificação de forma de onda de voz (*e. g.* [7, 8]), R é expressa em bits/amostra. Em se tratando de codificação de imagens (*e. g.* [9, 10]), R é expressa em bits por pixel (bpp).

O objetivo das técnicas de projeto de dicionário é reduzir (para uma determinada taxa R) a distorção introduzida ao se representarem os vetores de entrada $\boldsymbol{x}$ por suas correspondentes versões quantizadas $Q(\boldsymbol{x})$.

3. ALGORITMO LBG

Seja a iteração do algoritmo LBG denotada por n. Dados K, N e um limiar de distorção $\epsilon \geq 0$, o algoritmo LBG [20] consiste da seguinte seqüência de passos:

- *Passo 1*) inicialização: dado um dicionário inicial W_0 e um conjunto de treino $\boldsymbol{X} = \{\boldsymbol{x}_m;\ m = 1,\ 2,\ \ldots,\ M\}$, faça $n = 0$ e $D_{-1} = \infty$;

- *Passo 2*) particionamento: dado W_n (dicionário na n-ésima iteração) aloque cada vetor de treino (vetor de entrada) na respectiva classe (célula de Voronoi) segundo o critério do vetor-código mais próximo; calcule a distorção

$$D_n = \sum_{i=1}^{N} \sum_{\boldsymbol{x}_m \in S_i} d(\boldsymbol{x}_m, \boldsymbol{w}_i); \tag{5}$$

- *Passo 3*) teste de convergência (critério de parada): se $(D_{n-1} - D_n)/D_n \leq \epsilon$ pare, com W_n representando o dicionário final (dicionário projetado); caso contrário, continue;

- *Passo 4*) atualização do dicionário: compute os novos vetores-código como os centróides das classes de vetores; faça $W_{n+1} \leftarrow W_n$; faça $n \leftarrow n + 1$ e retorne ao *Passo 2*.

No algoritmo LBG a função distorção decresce monotonicamente, uma vez que o dicionário é iterativamente atualizado visando satisfazer as condições de centróide e de vizinho mais próximo. No algoritmo LBG, a distorção introduzida ao se representarem os vetores do conjunto de treinamento pelos correspondentes vetores-código (centróides) 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 $(D_{n-1} - D_n)/D_n \leq \epsilon$. Existem alguns problemas apresentados pelo algoritmo LBG, comumente relatados na literatura [31]: alguns vetores-código podem ser sub-utilizados e, em casos extremos, até mesmo nunca serem utilizados, ou seja, o projeto do dicionário pode resultar em células de Voronoi pequenas ou até mesmo vazias; a velocidade de convergência e o desempenho do dicionário final dependem do dicionário inicial.

4. ALGORITMO COMPETITIVO

Na aprendizagem competitiva simples aplicada ao projeto de dicionários, a cada apresentação de um vetor de treino apenas o *vencedor* (isto é, o vetor-peso mais semelhante ao vetor de treino) tem suas componentes atualizadas.

Após a inicialização dos pesos (isto é, das componentes dos N vetores-código K-dimensionais), dicionários podem ser projetados utilizando o algoritmo competitivo descrito a seguir.

Algoritmo Competitivo (AC):

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

Para $1 \leq m \leq M$

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

$$\tilde{w}_{i^*j}(n, m) = w_{i^*j}(n, m) + \Delta w_{i^*j}(n, m), \tag{6}$$

em que

$$\Delta w_{i^*j}(n, m) = \eta(n)[(x_j(m) - w_{i^*j}(n, m)]. \tag{7}$$

Na descrição acima, n_{AC} é o número total de iterações do algoritmo competitivo, $\boldsymbol{x}(m)$ é o m-ésimo vetor do conjunto de treino[1], 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,

$$d[\boldsymbol{x}(m), \boldsymbol{w}_i(n, m)] = \sum_{j=1}^{K} [x_j(m) - w_{ij}(n, m)]^2 \tag{8}$$

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[2].

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})[3]; taxa de aprendizagem inicial ($\eta(1)$); taxa de aprendizagem final ($\eta(n_{\mathrm{AC}})$).

[1] É importante observar que, por questões de conveniência de notação, utilizou-se $\boldsymbol{x}(m)$ na descrição do algoritmo AC e $\boldsymbol{x}_m$ na descrição do algoritmo LBG.

[2] Note que se omitiram n e m de $\Delta w_{i^*j}(n, m)$, de $w_{i^*j}(n, m)$ e de $\tilde{w}_{i^*j}(n, m)$ para simplificar a notação.

[3] Portanto, em projeto de dicionários utilizando o algoritmo AC, o número de iterações é especificado *a priori*. Por outro lado, em se tratando do algoritmo LBG, o número total de iterações depende do parâmetro de convergência ϵ bem como do dicionário inicial utilizados.

5. AVALIAÇÃO DE COMPLEXIDADE

Neste trabalho, a distância entre um determinado vetor de treino $\boldsymbol{x}$ e o vetor-código $\boldsymbol{w}_i$ é avaliada por meio da distorção (distância euclidiana quadrática)

$$d(\boldsymbol{x}, \boldsymbol{w}_i) = \sum_{j=1}^{K} (x_j - w_{ij})^2, \tag{9}$$

em que w_{ij} é a j-ésima componente do vetor-código $\boldsymbol{w}_i$ e x_j é a j-ésima componente do vetor de treino $\boldsymbol{x}$.

A seguir é apresentada uma avaliação de complexidade dos algoritmos LBG e AC.

5.1 Algoritmo LBG

No *Passo 2* do algoritmo LBG, para que seja determinada a classe à qual pertence um vetor de treino, é necessário encontrar a distância entre este vetor e cada um dos N vetores-código do dicionário e depois comparar as distâncias de modo a encontrar o vetor-código mais semelhante: um vetor de treino $\boldsymbol{x}$ é alocado para a i-ésima classe (célula de Voronoi S_i) se $d(\boldsymbol{x}, \boldsymbol{w}_i) < d(\boldsymbol{x}, \boldsymbol{w}_j),\ \forall j \neq i$. A determinação da classe a que pertence um vetor de entrada (vetor de treino) requer, portanto, N cálculos de distância e $N-1$ comparações. Ao ser utilizada a distância euclidiana, cada cálculo de distância requer K multiplicações, K subtrações e $K-1$ adições. A alocação de um vetor de entrada em uma das N classes requer, portanto, KN multiplicações, KN subtrações, $(K-1)N$ adições e $N-1$ comparações. Tendo em vista que o número de vetores de treino é M, a cada iteração do algoritmo LBG o processo de alocação dos vetores de treino nas respectivas classes requer KNM multiplicações, KNM subtrações, $(K-1)NM$ adições e $(N-1)M$ comparações. Em n_{LBG} iterações, este processo de alocação no *Passo 2* do algoritmo LBG requer $KNMn_{\mathrm{LBG}}$ multiplicações, $KNMn_{\mathrm{LBG}}$ subtrações, $(K-1)NMn_{\mathrm{LBG}}$ adições e $(N-1)Mn_{\mathrm{LBG}}$ comparações.

O número de operações requeridas para a determinação de D_n no *Passo 2* do algoritmo LBG é determinado a seguir. A distorção D_n é calculada somando as distâncias entre cada vetor de treino e o correspondente vetor-código da classe à qual foi alocado, o que requer $M-1$ somas de distâncias. Considerando n_{LBG} iterações, a determinação de D_n corresponde a $(M-1)n_{\mathrm{LBG}}$ adições.

Considere agora o *Passo 3*. O teste de convergência do algoritmo LBG precisa realizar uma comparação de $(D_{n-1} - D_n)/D_n$ com ϵ. Para que a comparação seja realizada, são necessárias uma subtração e uma divisão. Considerando n_{LBG} iterações, o número total de operações requeridas para a realização do teste de convergência do algoritmo LBG corresponde, portanto, a n_{LBG} comparações, n_{LBG} subtrações e n_{LBG} divisões.

O centróide da i-ésima ($1 \leq i \leq N$) classe é um vetor $\widetilde{\boldsymbol{w}}_i$ tal que sua j-ésima componente corresponde à média aritmética das j-ésimas componentes de todos vetores de treino alocados (no *Passo 2*) em S_i. Seja M_i o número de vetores de treino alocados em S_i. Assim,

$$\sum_{i=1}^{N} M_i = M. \tag{10}$$

A j-ésima ($1 \leq j \leq K$) componente $\widetilde{w}_{ij}$ do centróide $\widetilde{\boldsymbol{w}}_i$ é dada pela média

$$\widetilde{w}_{ij} = \frac{1}{M_i} \sum_{\boldsymbol{x}_m \in S_i} x_{mj}, \tag{11}$$

em que x_{mj} representa a j-ésima componente do vetor de treino $\boldsymbol{x}_m$. A determinação do centróide da i-ésima classe requer K cálculos de média de componentes (cada vetor tem K componentes), o que requer $K(M_i - 1)$ adições e K divisões. O número total de adições para a determinação dos N centróides é

$$\sum_{i=1}^{N} K(M_i - 1) = K\left(\sum_{i=1}^{N} M_i - \sum_{i=1}^{N} 1\right). \tag{12}$$

Pela Equação (10) segue que para cada iteração do algoritmo LBG, o *Passo 4* requer $K(M-N)$ adições. O número total de divisões no *Passo 4*, por sua vez, é KN. Admitindo n_{LBG} iterações, o número total de operações do *Passo 4* é igual a $K(M-N)n_{\mathrm{LBG}}$ adições e KNn_{LBG} divisões. Além destas operações, é importante ressaltar que o número (M_i, necessário para cálculo do centróide) de vetores de treino alocados para a i-ésima classe é determinado mediante uso de um contador[4], o que implica M_i adições. Considerando as N classes, são utilizados N contadores (um para cada classe), que consomem $\sum_{i=1}^{N} M_i = M$ adições. Portanto, ao final de n_{LBG} iterações, o *Passo 4* do algoritmo LBG requer, também, Mn_{LBG} adições.

5.2 Algoritmo Competitivo

No algoritmo competitivo o número de operações requeridas para a determinação do vencedor, isto é, para a determinação de $\boldsymbol{w}_{i^*} : d(\boldsymbol{x}, \boldsymbol{w}_{i^*}) < d(\boldsymbol{x}, \boldsymbol{w}_j), \forall j \neq i^*$, é igual a $KNMn_{\mathrm{AC}}$ multiplicações, $KNMn_{\mathrm{AC}}$ subtrações, $(K-1)NMn_{\mathrm{AC}}$ adições e $(N-1)Mn_{\mathrm{AC}}$ comparações.

[4]Para ser mais preciso, esta contagem é feita no *Passo 2*, na etapa de alocação dos vetores de treino nas respectivas classes.

Para cada vetor de treino, o vencedor (vetor-código mais semelhante a $\boldsymbol{x}$ segundo o critério de distância mínima) é atualizado de acordo com as Equações (6) e (7). Esta atualização requer K adições (uma adição para cada componente de vetor, conforme Equação (6)), K multiplicações (uma multiplicação para cada componente de vetor, conforme Equação (7)) e K subtrações (uma subtração para cada componente de vetor, conforme Equação (7)). Ao final de n_{AC} iterações e considerando M vetores de treino, a atualização dos vencedores requer KMn_{AC} adições, KMn_{AC} multiplicações e KMn_{AC} subtrações.

Ao final de cada iteração, a taxa de aprendizagem é calculada de acordo com

$$\eta(n) = \eta(1) + (n-1)\frac{\eta(n_{\mathrm{AC}}) - \eta(1)}{n_{\mathrm{AC}} - 1}. \tag{13}$$

Observe que o cálculo da taxa de aprendizagem não é realizado ao final da última iteração. Vale lembrar que $\eta(1)$, $\eta(n_{\mathrm{AC}})$ e n_{AC} são parâmetros do algoritmo AC. O valor $\frac{\eta(n_{\mathrm{AC}})-\eta(1)}{n_{\mathrm{AC}}-1}$ é calculado uma única vez (requerendo duas subtrações e uma divisão), ao final da primeira iteração, sendo armazenado para utilização ao final das demais iterações. Ao final de cada iteração (com exceção da última), para que a taxa de aprendizagem seja calculada são necessárias uma adição, uma subtração e uma multiplicação $\left(\text{por } \frac{\eta(n_{\mathrm{AC}})-\eta(1)}{n_{\mathrm{AC}}-1}\right)$. Assim, o número total de operações envolvidas nos $n_{\mathrm{AC}} - 1$ cálculos de $\eta(n)$ corresponde a $2 + n_{\mathrm{AC}} - 1$ subtrações, uma divisão, $n_{\mathrm{AC}} - 1$ adições e $n_{\mathrm{AC}} - 1$ multiplicações.

5.3 LBG versus AC

As Tabelas 1 e 2 apresentam um resumo do número de adições (ad.), subtrações (sub.), divisões (div.), multiplicações (mult.) e comparações (comp.) requeridas pelos algoritmo LBG e AC, respectivamente.

Tabela 1: Número de operações requeridas pelo algoritmo LBG.

	Passo 2
mult.	$KNMn_{\mathrm{LBG}}$
sub.	$KNMn_{\mathrm{LBG}}$
ad.	$[(K-1)NM + (M-1)]n_{\mathrm{LBG}}$
comp.	$(N-1)Mn_{\mathrm{LBG}}$
	Passo 3
comp.	n_{LBG}
sub.	n_{LBG}
div.	n_{LBG}
	Passo 4
ad.	$[(K+1)M - KN]n_{\mathrm{LBG}}$
div.	KNn_{LBG}

Tabela 2: Número de operações requeridas pelo algoritmo AC.

	Determinação do vencedor
mult.	$KNMn_{\mathrm{AC}}$
sub.	$KNMn_{\mathrm{AC}}$
ad.	$(K-1)NMn_{\mathrm{AC}}$
comp.	$(N-1)Mn_{\mathrm{AC}}$
	Atualização do vencedor
ad.	KMn_{AC}
mult.	KMn_{AC}
sub.	KMn_{AC}
	Cálculo da taxa de aprendizagem
sub.	$n_{\mathrm{AC}} + 1$
div.	1
ad.	$n_{\mathrm{AC}} - 1$
mult.	$n_{\mathrm{AC}} - 1$

A Tabela 3 apresenta o número total de operações dos algoritmos LBG e AC.

Tabela 3: 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	$[1+(N-1)M]n_{\mathrm{LBG}}$	$(N-1)Mn_{\mathrm{AC}}$

6. CONDIÇÕES PARA AS QUAIS O ALGORITMO AC É MAIS EFICIENTE QUE O ALGORITMO LBG EM TERMOS DE NÚMERO 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 3) separadamente, conforme segue.

6.1 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

$$[1+(1+N)KM]n_{\mathrm{AC}}-1<KNMn_{\mathrm{LBG}}, \tag{14}$$

ou seja,

$$n_{\mathrm{AC}}<\frac{1+KNMn_{\mathrm{LBG}}}{1+(1+N)KM}. \tag{15}$$

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, (15) se reduz a

$$n_{\mathrm{AC}}<\frac{N}{1+N}n_{\mathrm{LBG}}. \tag{16}$$

6.2 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

$$[1+(1+N)KM]n_{\mathrm{AC}}+1<(1+KNM)n_{\mathrm{LBG}}, \tag{17}$$

ou seja,

$$n_{\mathrm{AC}}<\frac{(1+KNM)n_{\mathrm{LBG}}-1}{1+(1+N)KM}. \tag{18}$$

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 (18) se reduz a

$$n_{\mathrm{AC}}<\frac{(1+KNM)n_{\mathrm{LBG}}}{(1+N)KM}. \tag{19}$$

Tendo em vista que $1+KNM\approx KMN$, (19) se reduz a

$$n_{\mathrm{AC}}<\frac{N}{1+N}n_{\mathrm{LBG}}. \tag{20}$$

6.3 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

$$[1+(K-1)NM+KM]n_{\mathrm{AC}}-1<[(1+KN)(M-1)+(K-N+1)M]n_{\mathrm{LBG}}, \tag{21}$$

ou seja,

$$n_{\mathrm{AC}}<\frac{1+[(1+KN)(M-1)+(K-N+1)M]n_{\mathrm{LBG}}}{1+(K-1)NM+KM}. \tag{22}$$

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 (22) se reduz a

$$n_{\mathrm{AC}} < \frac{[(1+KN)(M-1)+(K-N+1)M]n_{\mathrm{LBG}}}{(K-1)NM+KM}. \tag{23}$$

Tendo em vista que $M-1 \approx M$, (23) se reduz a

$$n_{\mathrm{AC}} < \frac{[(1+KN)M+(K-N+1)M]n_{\mathrm{LBG}}}{(K-1)NM+KM}. \tag{24}$$

Após algumas manipulações, (24) se reduz a

$$n_{\mathrm{AC}} < \frac{[(K-1)(1+N)+3]}{(K-1)N+K} n_{\mathrm{LBG}}. \tag{25}$$

6.4 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

$$(N-1)Mn_{\mathrm{AC}} < [1+(N-1)M]n_{\mathrm{LBG}}, \tag{26}$$

ou seja,

$$n_{\mathrm{AC}} < \frac{[1+(N-1)M]n_{\mathrm{LBG}}}{(N-1)M}. \tag{27}$$

Como $1+(N-1)M \approx (N-1)M$ em projetos típicos de dicionários (ou seja, para elevados valores de M), a condição (27) se reduz a

$$n_{\mathrm{AC}} < n_{\mathrm{LBG}}. \tag{28}$$

6.5 Considerações Gerais

Considere as Inequações (16) e (20), referentes, respectivamente, ao número de multiplicações e subtrações. À medida que N aumenta, $\frac{N}{1+N}$ tende a 1. Deste modo, à medida que N aumenta, as condições (16) e (20) tendem a se reduzir simplesmente 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 (16) e (20) são satisfeitas para

$$n_{\mathrm{AC}} < \frac{n_{\mathrm{LBG}}}{2}. \tag{29}$$

Portanto, é lícito afirmar que, uma vez satisfeita a condição (29), o número de multiplicações e subtrações realizadas pelo algoritmo AC é inferior ao número de multiplicações e subtrações do algoritmo LBG.

Considere a Inequação (25). Para $K \geq 1$ e $N \geq 1$, conforme se pode observar na Figura 3, segue que

$$\frac{[(K-1)(1+N)+3]}{(K-1)N+K} > 1, \tag{30}$$

de modo que

$$\frac{[(K-1)(1+N)+3]}{(K-1)N+K} n_{\mathrm{LBG}} > n_{\mathrm{LBG}}. \tag{31}$$

Assim, uma vez assegurado que

$$n_{\mathrm{AC}} < n_{\mathrm{LBG}}, \tag{32}$$

a condição (25) é automaticamente satisfeita. Portanto, uma vez satisfeita a condição (32), o número de adições (como também o número de comparações[5]) realizadas pelo algoritmo AC é inferior ao número de adições (comparações) do algoritmo LBG.

Tendo em vista que a condição (29) é mais severa que a condição (32) é lícito afirmar que a condição (29) deve ser satisfeita para que o número de operações do algoritmo AC seja inferior ao número de operações do algoritmo LBG.

7. RESULTADOS

Os resultados apresentados neste trabalho encontram-se divididos em duas seções. A Seção 7.1 apresenta resultados referentes ao projeto de dicionários aplicados à codificação de imagem. Na Seção 7.2 são apresentados resultados concernentes ao projeto de dicionários aplicados à codificação de sinal com distribuição de Gauss-Markov [32, 33] com coeficiente de correlação igual a 0,8.

[5]Conforme mostra a Inequação (28).

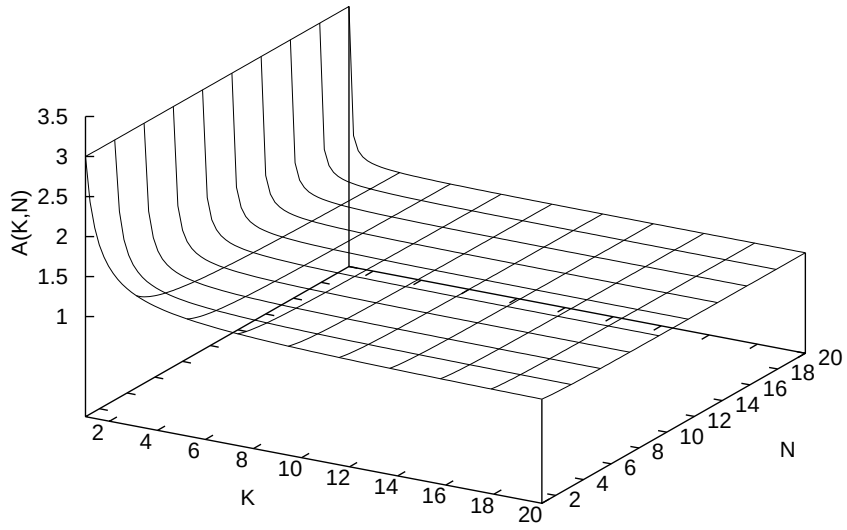


Figura 3: Gráfico de $A(K,N) = \frac{[(K-1)(1+N)+3]}{(K-1)N+K}$.

7.1 Imagem

Os resultados apresentados nesta seção foram obtidos com dicionários projetados utilizando um conjunto de treino constituído de 7 imagens (256×256 pixels). Foi considerada quantização vetorial com dimensão $K = 16$, usando blocos de 4×4 pixels. Utilizou-se, portanto, um conjunto de treino com 28672 vetores de dimensão 16. Foram projetados dicionários com $N = 32$, 64, 128, 256 e 512 vetores-código. Foram consideradas, por conseguinte, as respectivas taxas de codificação $R = 0{,}3125$ bpp, 0,375 bpp, 0,4375 bpp, 0,5 bpp e 0,5625 bpp. O algoritmo LBG foi executado até que que a modificação na distorção introduzida ao se representar os vetores de treino pelo dicionário fosse inferior a 0,1% ($\epsilon = 0,001$).

As Tabelas 4 e 5 apresentam o número total de iterações e o correspondente número total de operações dos algoritmos LBG e AC, obtidos com as condições (parâmetros, inicialização) que levaram aos dicionários de melhor qualidade, ou seja, aqueles que resultaram nos maiores valores de relação sinal-ruído de pico (PSNR – *Peak Signal-to-noise Ratio*) das imagens reconstruídas, definida como 10 vezes o logaritmo na base 10 da razão entre o quadrado do valor de pico da amplitude do sinal de entrada, v_p^2, e o erro médio quadrático (MSE, *mean square error*):

$$\text{PSNR} = 10 \log_{10} \left[\frac{v_p^2}{\text{MSE}} \right]. \tag{33}$$

Para o caso de uma imagem original codificada a 8,0 bpp (256 níveis de cinza),

$$\text{PSNR} = 10 \log_{10} \left[\frac{255^2}{\text{MSE}} \right], \tag{34}$$

em que o erro médio quadrático entre as imagens original e reconstruída é definido como

$$\text{MSE} = \frac{1}{L \cdot C} \sum_{l=1}^{L} \sum_{c=1}^{C} [F(l,c) - \widehat{F}(l,c)]^2, \tag{35}$$

em que $F(l,c)$ e $\widehat{F}(l,c)$ representam os valores de pixels das imagens original e reconstruída, l designa a l-ésima linha e c denota a c-ésima coluna de uma imagem (matriz) $L \times C$.

Em se tratando do algoritmo LBG, para cada N, foram testados cinco dicionários iniciais diferentes. Os valores de n_{tot} da Tabela 4 correspondem ao dicionário inicial que levou aos melhores dicionários em termos de PSNR. Conforme mencionado anteriormente, o desempenho do algoritmo LBG e sua velocidade de convergência (número total de iterações) dependem do dicionário inicial: para $N = 128$, por exemplo, os cinco dicionários iniciais diferentes levaram a 43, 59, 17, 26 e 14 iterações.

As Tabelas 4 e 5 mostram que para todos os valores de tamanho do dicionário (N) avaliados, o algoritmo AC requer um número de iterações inferior à metade do número de iterações requeridas pelo algoritmo LBG: para $N = 256$, por exemplo, o dicionário LBG realiza 15 iterações, enquanto o algoritmo AC requer 4 iterações para projetar o dicionário. Observa-se também nas Tabelas 4 e 5 que o algoritmo AC realiza um número de operações inferior ao número de operações realizadas pelo algoritmo LBG.

A Figura 4 apresenta o desempenho dos algoritmos LBG e AC na codificação da imagem Mandrill, apresentada na Figura 5. Conforme se pode observar na Figura 4, apesar de o algoritmo competitivo utilizar um número de operações inferior ao apresentado pelo algoritmo LBG, os dicionários obtidos com o algoritmo competitivo levam a imagens reconstruídas com valores de PSNR ligeiramente superiores aos apresentados pelas imagens reconstruídas obtidas com uso de dicionários LBG.

Tabela 4: Número de operações requeridas pelo algoritmo LBG ao serem projetados dicionários destinados à codificação de imagem baseada em QV.

N	R	n_{tot}	mult.	sub.	ad.	div.	comp.
32	0,3125	18	$2,64 \cdot 10^8$	$2,64 \cdot 10^8$	$2,57 \cdot 10^8$	$9,23 \cdot 10^3$	$1,60 \cdot 10^7$
64	0,375	13	$3,82 \cdot 10^8$	$3,82 \cdot 10^8$	$3,65 \cdot 10^8$	$1,33 \cdot 10^4$	$2,35 \cdot 10^7$
128	0,4375	17	$9,98 \cdot 10^8$	$9,98 \cdot 10^8$	$9,45 \cdot 10^8$	$3,48 \cdot 10^4$	$6,19 \cdot 10^7$
256	0,5	15	$1,76 \cdot 10^9$	$1,76 \cdot 10^9$	$1,66 \cdot 10^9$	$6,15 \cdot 10^4$	$1,10 \cdot 10^8$
512	0,5625	15	$3,52 \cdot 10^9$	$3,52 \cdot 10^9$	$3,31 \cdot 10^9$	$1,23 \cdot 10^5$	$2,20 \cdot 10^8$

Tabela 5: Número de operações requeridas pelo algoritmo AC ao serem projetados dicionários destinados à codificação de imagem baseada em QV.

N	R	n_{tot}	mult.	sub.	ad.	div.	comp.
32	0,3125	3	$4,54 \cdot 10^7$	$4,54 \cdot 10^7$	$4,27 \cdot 10^7$	1	$2,67 \cdot 10^6$
64	0,375	3	$8,95 \cdot 10^7$	$8,95 \cdot 10^7$	$8,40 \cdot 10^7$	1	$5,42 \cdot 10^6$
128	0,4375	3	$1,78 \cdot 10^8$	$1,78 \cdot 10^8$	$1,67 \cdot 10^8$	1	$1,09 \cdot 10^7$
256	0,5	4	$4,72 \cdot 10^8$	$4,72 \cdot 10^8$	$4,42 \cdot 10^8$	1	$2,92 \cdot 10^7$
512	0,5625	5	$1,18 \cdot 10^9$	$1,18 \cdot 10^9$	$1,10 \cdot 10^9$	1	$7,33 \cdot 10^7$

É oportuno mencionar que a superioridade do algoritmo AC sobre o algoritmo LBG, em termos de número de iterações e número total de operações também foi constatada ao serem utilizados conjuntos de treino com outros tamanhos.

7.2 Gauss-Markov

Em diversos trabalhos [4, 25, 26, 33–36], o desempenho de algoritmos para projeto de quantizadores vetoriais é avaliado utilizando fontes com distribuições conhecidas, dentre as quais destacam-se a fonte gaussiana e a fonte de Gauss-Markov de $1^{\underline{a}}$ ordem, também conhecida como fonte de Markov de $1^{\underline{a}}$ ordem, fonte de Gauss autoregressiva de $1^{\underline{a}}$ ordem ou fonte AR(1). Para efeito de simplicidade, essa fonte será referenciada simplesmente como fonte de Gauss-Markov ao longo do presente trabalho.

O processo discreto de Gauss-Markov $\{X(n)\}$ é definido pela equação

$$X(n) = aX(n-1) + W(n), \ \ \forall n, \tag{36}$$

em que a denota o coeficiente de correlação e $\{W(n)\}$ é uma seqüência de amostras independentes de uma variável aleatória gaussiana com média nula.

O processo gaussiano (descorrelacionado) é obtido fazendo $a = 0$ na Equação (36).

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 cinco dicionários iniciais diferentes para cada combinação de K e N avaliada.

A Tabela 6 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*) [32] do sinal reconstruído, é apresentada na Tabela 7 – 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 7, 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 a favor de AC observada na Tabela 7 foi de 0,09 dB, enquanto que a maior diferença a favor de LBG observada na Tabela 7 foi de 0,03 dB.

No que diz respeito ao número de multiplicações, subtrações, adições, divisões e comparações realizadas pelos algoritmos LBG e AC quando aplicados ao projeto de dicionários destinados à codificação do sinal com distribuição de Gauss-Markov, as Tabelas 8 e 9 mostram que, para todos os valores de K e N (com a correspondente taxa de codificação R, expressa em bits/amostra) considerados, o algoritmo AC realiza um número de operações inferior ao número de operações realizadas pelo algoritmo LBG.

Finalmente, cumpre mencionar que a condição (29) 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

Tabela 6: 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

Tabela 7: Diferença de desempenho dos algoritmos LBG e AC em termos de relação sinal-ruído (SNR), expressa em decibéis (dB), do sinal reconstruído.

K	N	$\mathrm{SNR}_{\mathrm{AC}} - \mathrm{SNR}_{\mathrm{LBG}}$	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

Tabela 8: 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 9: 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	$1,91 \cdot 10^7$

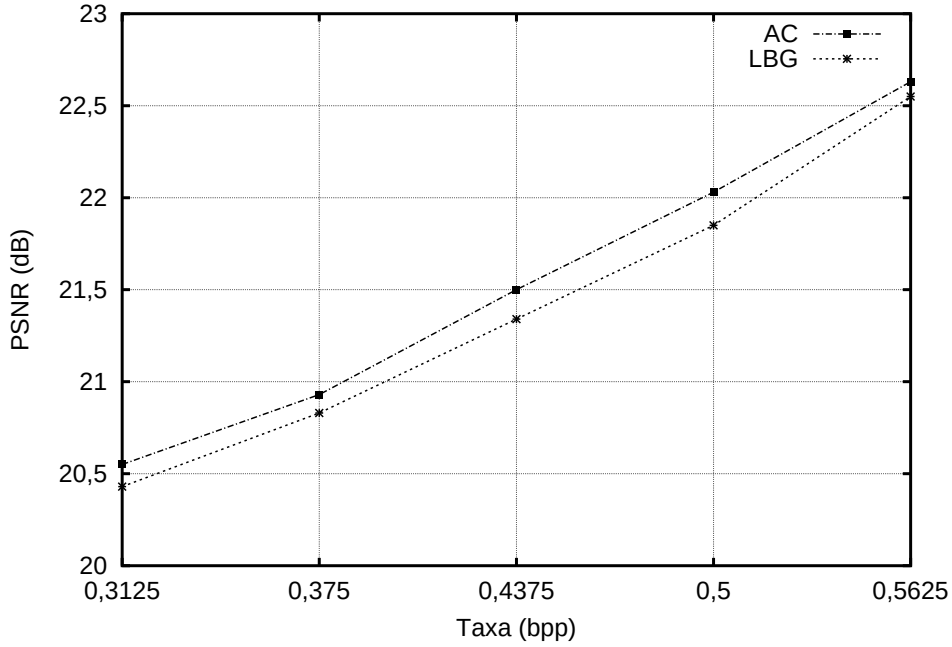


Figura 4: Desempenho dos algoritmos AC e LBG em QV de imagem: PSNR da imagem Mandrill reconstruída versus taxa de codificação para QV com dimensão $K = 16$.

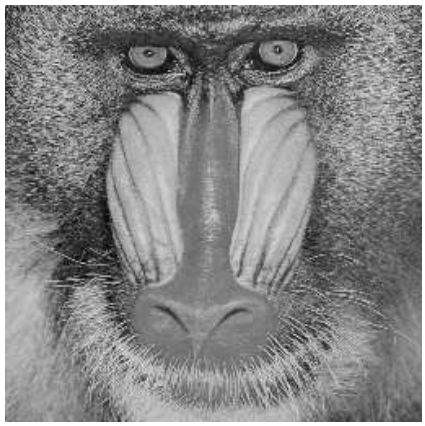


Figura 5: Imagem Mandrill.

com qualidade (avaliada por meio da relação sinal-ruído segmental) próxima ou ligeiramente superior à apresentada pelos sinais reconstruídos com uso de dicionários LBG.

8. CONCLUSÃO

Este trabalho apresentou uma 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 (em função da dimensão dos vetores-código, do tamanho do dicionário, do número de vetores do conjunto de treino e do número de iterações realizadas) 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 projetos de dicionários aplicados à codificação de imagem e à codificação de sinal com distribuição de Gauss-Markov, mostraram que a condição $n_{\mathrm{AC}} < \frac{n_{\mathrm{LBG}}}{2}$ 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 a mesma qualidade (avaliada por meio da relação sinal-ruído de pico para o caso de imagens e por meio da relação sinal-ruído para o caso de sinais com distribuição de Gauss-Markov) obtida com uso de dicioná-

rios LBG.

AGRADECIMENTOS

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

REFERÊNCIAS

[1] N. Jayant. "Signal Compression: Technology Targets and Research Directions". *IEEE Journal on Selected Areas in Communications*, vol. 10, no. 5, pp. 796–818, June 1992.

[2] N. Jayant, J. Johnston and R. Safranek. "Signal Compression Based on Models of Human Perception". *Proceedings of the IEEE*, vol. 81, no. 10, pp. 1385–1422, October 1993.

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

[4] R. M. Gray. "Vector Quantization". *IEEE ASSP Magazine*, pp. 4–29, April 1984.

[5] T. Berger. *Rate Distortion Theory: A Mathematical Basis for Data Compression*. Prentice-Hall, Englewood Cliffs, NJ, 1971.

[6] R. M. Gray and D. L. Neuhoff. "Quantization". *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2325–2383, October 1998.

[7] H. Abut, R. M. Gray and G. Rebolledo. "Vector Quantization of Speech and Speech-Like Waveforms". *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 30, no. 3, pp. 423–435, June 1982.

[8] A. Gersho and V. Cuperman. "Vector Quantization: A Pattern-Matching Technique for Speech Coding". *IEEE Communications Magazine*, pp. 15–20, December 1983.

[9] B. Ramamurthi and A. Gersho. "Classified Vector Quantization of Images". *IEEE Transactions on Communications*, vol. 34, no. 11, pp. 1105–1115, November 1986.

[10] N. M. Nasrabadi and R. A. King. "Image Coding Using Vector Quantization: A Review". *IEEE Transactions on Communications*, vol. 36, no. 8, pp. 957–971, August 1988.

[11] P. C. Cosman, R. M. Gray and M. Vetterli. "Vector Quantization of Image Subbands: A Survey". *IEEE Transactions on Image Processing*, vol. 5, no. 2, pp. 202–225, February 1996.

[12] M. Antonini, M. Barlaud, P. Mathieu and I. Daubechies. "Image Coding Using Wavelet Transform". *IEEE Transactions on Image Processing*, vol. 1, no. 2, pp. 205–220, April 1992.

[13] A. Kjoelen, S. E. Umbaugh and M. Zuke. "Compression of Skin Tumor Images – Wavelet/Vector Quantization Methods for Reducing the Time, Cost and Bandwidth of Storing and Transmitting Data". *IEEE Engineering in Medicine and Biology*, pp. 73–80, May/June 1998.

[14] K. K. Paliwal and B. S. Atal. "Efficient Vector Quantization of LPC Parameters at 24 Bits/Frame". *IEEE Transactions on Speech and Audio Processing*, vol. 1, no. 1, pp. 3–14, January 1993.

[15] F. K. Soong, A. E. Rosenberg, B.-H. Juang and L. R. Rabiner. "A Vector Quantization Approach to Speaker Recognition". *AT&T Technical Journal*, vol. 66, no. 2, pp. 14–26, March/April 1987.

[16] R. P. Ramachandran, M. S. Zilovic and R. J. Mammone. "A Comparative Study of Robust Linear Predictive Analysis Methods with Applications to Speaker Identification". *IEEE Transactions on Speech and Audio Processing*, vol. 3, no. 2, pp. 117–125, March 1995.

[17] L. Xu, J. Oglesby and J. S. Mason. "The Optimization of Perceptually-based Features for Speaker Identification". In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'89)*, pp. 520–523, 1989.

[18] J. He, L. Liu and G. Palm. "A Discriminative Training Algorithm for VQ-based Speaker Identification". *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 3, pp. 353–356, May 1999.

[19] M. S. Zilovic, R. P. Ramachandran and R. J. Mammone. "Speaker Identification Based on the Use of Robust Cepstral Features Obtained from Pole-zero Transfer Functions". *IEEE Transactions on Speech and Audio Processing*, vol. 6, no. 3, pp. 260–267, May 1998.

[20] Y. Linde, A. Buzo and R. M. Gray. "An Algorithm for Vector Quantizer Design". *IEEE Transactions on Communications*, vol. 28, no. 1, pp. 84–95, January 1980.

[21] T. Kohonen. "The Self-Organizing Map". *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, September 1990.

[22] 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*, vol. 8, no. 8, pp. 1449–1457, October 1990.

[23] 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*, vol. 4, no. 5, pp. 480–489, October 1994.

[24] C. Zhu and L. M. Po. "Partial Distortion Sensitive Competitive Learning Algorithm for Optimal Codebook Design". *Electronics Letters*, vol. 32, no. 19, pp. 1757–1758, September 1996.

[25] E. Yair, K. Zeger and A. Gersho. "Competitive Learning and Soft Competition for Vector Quantizer Design". *IEEE Transactions on Signal Processing*, vol. 40, no. 2, pp. 294–309, February 1992.

[26] K. Zeger, J. Vaisey and A. Gersho. "Globally Optimal Vector Quantizer Design by Stochastic Relaxation". *IEEE Transactions on Signal Processing*, vol. 40, no. 2, pp. 310–322, February 1992.

[27] N. B. Karayiannis and P.-I. Pai. "Fuzzy Vector Quantization Algorithms and Their Applications in Image Compression". *IEEE Transactions on Image Processing*, vol. 4, no. 9, pp. 1193–1201, September 1995.

[28] N. B. Karayiannis, P.-I. Pai and N. Zervos. "Image Compression Based on Fuzzy Algorithms for Learning Vector Quantization and Wavelet Image Decomposition". *IEEE Transactions on Image Processing*, vol. 7, no. 8, pp. 1223–1230, August 1998.

[29] J. S. Pan, F. R. McInnes and M. A. Jack. "VQ Codebook Design Using Genetic Algorithms". *Electronics Letters*, vol. 31, no. 17, pp. 1418–1419, August 1995.

[30] B. Kosko. *Neural Networks and Fuzzy Systems*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1992.

[31] D. Lee, S. Baek and K. Sung. "Modified K-means Algorithm for Vector Quantizer Design". *IEEE Signal Processing Letters*, vol. 4, no. 1, pp. 2–4, January 1997.

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

[33] R. M. Gray and Y. Linde. "Vector Quantizers and Predictive Quantizers for Gauss-Markov Sources". *IEEE Transactions on Communications*, vol. 30, no. 2, pp. 381–389, February 1982.

[34] L. Wu and F. Fallside. "Source Coding and Vector Quantization with Codebook Excited Neural Networks". *Computer Speech and Language*, pp. 243–276, 1992.

[35] H. A. Kaouri and J. V. McCanny. "Reduced-Memory Multirate Vector Quantisation". *Electronics Letters*, vol. 25, no. 7, pp. 471–473, March 1989.

[36] A. Makur and K. P. Subbalakshmi. "Variable Dimension VQ Encoding and Codebook Design". *IEEE Transactions on Communications*, vol. 45, no. 8, pp. 897–899, August 1997.