ACELERANDO A BUSCA PELO ELEMENTO ÓTIMO EM ALGORITMOS BASEADOS EM QUANTIZAÇÃO VETORIAL

Jacqueline S. Pereira[1]
Murilo B. de Carvalho[1]
Alexandre S. de la Vega[1]

Resumo: Técnicas como a quantização vetorial e o algoritmo de *k-means* têm sido amplamente usadas em aplicações de processamento digital de sinais. Tais métodos fazem uso de um conjunto de vetores, chamado de dicionário, sobre o qual é realizada uma busca, com o objetivo de representar os vetores de uma seqüência de dados de entrada pelos vetores do dicionário, de modo a minimizar a distorção medida de acordo com uma métrica específica. A criação do dicionário ótimo é não trivial. Porém, para um dado dicionário, é possível acelerar o processo de busca, de forma a torná-lo mais rápido e eficiente. Este trabalho propõe uma nova técnica para acelerar o processo de busca do melhor vetor do dicionário, baseada na partição do dicionário original em subdicionários menores. A melhor estratégia de partição é estudada, considerando-se o número de partições efetuadas e a quantidade de redundância adotada nos subdicionários.

Palavras-chave: algoritmos de *clustering*, *k-means*, quantização vetorial, subdicionários.

Abstract: Vector quantization and the k-means clustering algorithm are methods widely used in digital signal processing and data compression applications. In these techniques, a set of vectors, called codebook or dictionary, is searched for the best vectors to replace each vector of an input data sequence. The problem of finding the optimum codebook is non-trivial, but for a given dictionary it is possible to speed up the search for the best vector, a very computational intensive task. This work proposes a new approach for the search of the best vector in the codebook, based on dictionary partition. The best strategy is studied, considering the number of partitions and the redundancy created on the subdictionaries set.

Keywords: clustering algorithms, k-means, vector quantization, subdictionaries.

1. INTRODUÇÃO

Algoritmos de *clustering* têm sido amplamente empregados em aplicações de processamento digital de sinais, tais como compressão com perda, detecção de bordas, reconhecimento de objetos e casamento de padrões [1, 2, 3]. Entre as técnicas mais utilizadas estão a quantização vetorial [4] e o algoritmo *k-means* [5]. Nessas técnicas, é utilizado um conjunto de vetores chamado de *codebook* ou dicionário, cujos elementos são comparados a vetores da seqüência de dados de entrada. O objetivo é substituir os vetores de entrada pelos vetores do dicionário, com o melhor desempenho, segundo algum critério de otimização.

Normalmente, a geração de um dicionário é muito demorada e não trivial. Porém, uma vez criado o dicionário, é possível acelerar o processo de busca, tornando a codificação/decodificação de informações mais rápida e eficiente. Uma das formas de acelerar o processo é dividir o *codebook* original em dicionários menores, pois a complexidade computacional do processo de busca depende do tamanho do dicionário. De fato, caso

[1] Universidade Federal Fluminense - Escola de Engenharia. Departamento de Engenharia de Telecomunicações. Rua Passo da Pátria, 156 / Bl. D / Sala 504, São Domingos – Niterói – RJ. CEP: 24.210-240. *E-mail*: {jac,murilo,delavega}@telecom.uff.br.

a distância euclideana seja adotada como critério de distorção, quando utiliza-se um dicionário $D = \{\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_L\}$ contendo L vetores de dimensão M, necessita-se efetuar M subtrações, M multiplicações e $M-1$ somas, para determinar-se o módulo ao quadrado da diferença entre um vetor $\boldsymbol{v}_i$ do dicionário e o vetor de entrada $\boldsymbol{x}$. Assim sendo, como o dicionário contém L vetores, gasta-se um total de LM subtrações, LM multiplicações e $L(M-1)$ somas, para determinar-se qual dos vetores $\boldsymbol{v}_i$ do dicionário é o mais próximo de $\boldsymbol{x}$. Se o critério de proximidade não for a distância euclideana, verifica-se que a complexidade é proporcional a $L\,f(M)$, onde $f(M)$ depende da métrica adotada. Algumas técnicas que se baseiam neste princípio procuram otimizar um conjunto de dicionários que são utilizados de forma hierárquica [6].

Neste trabalho, é proposta uma técnica alternativa que permite repartir um dicionário qualquer em diversos subdicionários, de modo a acelerar a busca. É importante destacar que este método pode ser aplicado a qualquer dicionário, mesmo àqueles que não tenham sido originalmente otimizados para busca hierárquica.

O artigo é dividido da seguinte forma: na Seção 2 é discutida a técnica de partição proposta. O algoritmo proposto é descrito na Seção 3. Na Seção 4 são definidas as simulações realizadas e apresentados os seus resultados. As conclusões encontram-se na Seção 5. Finalmente, na Seção 6 são indicados alguns trabalhos futuros.

2. PROPOSTA DE PARTIÇÃO

Desconsiderando-se os passos de criação do dicionário e estabelecendo-se como critério de otimização a medida de distorção entre o vetor de entrada e os vetores do dicionário, subdivide-se o dicionário em N partições. Para isso, inicialmente são gerados N vetores uniformemente distribuídos em um hipercubo, chamados centróides. A cada centróide $\boldsymbol{c}_i$ associa-se um subdicionário, composto pelos vetores $\boldsymbol{v}_j$, do dicionário original, mais próximos do centróide.

Para realizar uma busca, verifica-se qual o centróide mais próximo do vetor de entrada $\boldsymbol{x}$. Este centróide determina qual subdicionário será utilizado para codificar $\boldsymbol{x}$, que deverá ser representado pelo vetor mais próximo pertencente a este subdicionário.

Uma vez que o dicionário original é dividido em grupos menores, o vetor $\boldsymbol{v}_j$ que irá representar o vetor de entrada $\boldsymbol{x}$ é buscado **apenas** no subdicionário que possui o centróide mais próximo a $\boldsymbol{x}$. Desse modo, diminui-se o número de vetores a ser comparado, aumentando-se a velocidade e, conseqüentemente, diminuindo-se o tempo de busca.

Contudo, deve-se ter cuidado ao se construir os subdicionários. Uma abordagem é colocar no i-ésimo subdicionário SD_i apenas aqueles vetores do dicionário original D que sejam mais próximos do centróide $\boldsymbol{c}_i$. Porém, dependendo da configuração geométrica dos vetores, pode acontecer que a busca em duas etapas (primeiro escolhendo o centróide mais próximo e depois escolhendo o vetor mais próximo no subdicionário selecionado) não seja equivalente à busca direta no dicionário original. Esta situação é ilustrada na Figura 1, onde observa-se que o centróide mais próximo de $\boldsymbol{x}$ é $\boldsymbol{c}_2$, enquanto o vetor mais próximo é $\boldsymbol{v}_2$.

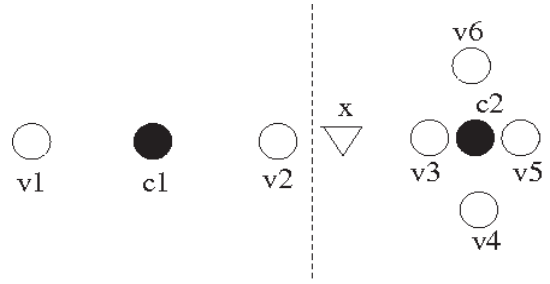

Figura 1 – Vetores e centróides.

Uma forma de solucionar este problema é acrescentar um dado vetor $\boldsymbol{v}_j$ do dicionário original D não apenas ao subdicionário SD_i, cujo centróide $\boldsymbol{c}_i$ é o mais próximo de $\boldsymbol{v}_j$, mas também ao subdicionário associado ao segundo centróide mais próximo, ao terceiro mais próximo e assim por diante.

Definindo-se **redundância** V como o número de subdicionários aos quais um dado vetor $\boldsymbol{v}_i$ do dicionário original pertence, este trabalho se propõe a estudar o número de partições do *codebook* e a redundância necessária em uma dada partição para que a distorção na codificação seja a menor possível.

3. ALGORITMO PROPOSTO

Pretende-se mostrar que, com a divisão do dicionário em partes menores e a inclusão de re-

dundância, é possível, além de diminuir o tempo da busca, fazer com que a distorção seja igual à de *D*, qualquer que seja a dimensão do vetor de entrada. É necessário, então, encontrar um número de partições *N* e a redundância *V*, que faça com que isto ocorra. Com tal objetivo, foi implementado o algoritmo a seguir.

Algoritmo:

- Com base em uma distribuição gaussiana, criar um vetor de entrada $\boldsymbol{x}$, contendo *M* linhas (dimensão do espaço vetorial).
- Com base em uma distribuição gaussiana, criar um dicionário (*codebook*) *D*, com *M* linhas (dimensão do espaço vetorial) e *L* colunas (número de vetores do dicionário).
- Calcular a distorção entre o dicionário original *D* e o vetor de entrada $\boldsymbol{x}$, armazenando o resultado.
- Calcular os *N* centróides $\boldsymbol{c}_i$, um para cada partição.
- Dividir *D* nas *N* partições, preenchendo cada subdicionário SD_i do seguinte modo: calcula-se os V centróides $\mathbf{c}_k$ mais próximos de cada vetor $\boldsymbol{v}_j$. Em seguida, inclui-se $\boldsymbol{v}_j$ em cada um dos subdicionários SD_k associados aos centróides $\boldsymbol{c}_k$.
- Excluir os subdicionários com zero elementos.
- Buscar o centróide mais próximo do vetor de entrada $\boldsymbol{x}$.
- Calcular a distorção entre os vetores do subdicionário de centróide mais próximo e o vetor de entrada. O vetor de menor distorção será usado para codificar o vetor de entrada.
- Calcular o erro relativo percentual entre a distorção do dicionário completo e a distorção do subdicionário.

O algoritmo descrito acima foi implementado em MATLAB® e simulado para vários vetores de teste e vários dicionários.

4. SIMULAÇÕES E RESULTADOS

Para justificar a utilidade da idéia de particionar o dicionário em conjuntos menores, reduzindo o tempo de busca, é necessário verificar se tal divisão leva a resultados com distorções iguais às do dicionário original.

Como critério de avaliação, foi utilizado o erro relativo percentual entre: i) a distorção média entre o dicionário original *D* e os vetores de entrada $\boldsymbol{x}$ e ii) a distorção média entre os vetores $\boldsymbol{x}$ e os subdicionários de *D*.

Para verificar a redundância *V* e o número de partições *N* ótimos, que fazem com que a distorção seja idêntica à do dicionário original, o algoritmo descrito na seção anterior foi executado para vetores de entrada $\boldsymbol{x}$ de dimensões *M*=2, 3 e 4. Para cada dimensão *M*, foi criado um dicionário *D*, com 8192 (2^{13}) vetores, e foram utilizados 2048 (2^{11}) vetores de entrada $\boldsymbol{x}$.

Sendo *n* o número de partições em cada dimensão do vetor de entrada, cabe ressaltar que o número total de subdicionários é dado por $N=M^n$, enquanto o número de subdicionários efetivamente utilizado pode ser $L_c \leq N$. Isto ocorre porque alguns subdicionários podem resultar vazios, após o processo de mapeamento de vetores do dicionário original em cada subdicionário.

Foram testadas duas situações distintas: I) Caso 1 - o estudo da distorção com a variação do número de partições *n* em cada dimensão do dicionário, e II) Caso 2 - o estudo da distorção com a variação da redundância *V*. Ambos os casos são discutidos a seguir.

CASO 1: TESTE DA DISTORÇÃO COM A VARIAÇÃO DO NÚMERO DE PARTIÇÕES N.

Neste caso, para cada dimensão, admitiu-se que os subdicionários poderiam possuir redundância máxima de $V_{max}=5$.

Uma vez escolhido o valor da redundância a ser utilizado, apenas o número de partições foi variado, podendo o dicionário ser dividido em até $n=8$.

Desta forma, foi possível verificar, também, se a inclusão de redundância reduz ou não a distorção.

Os valores dos erros relativos percentuais obtidos estão apresentados na Tabela 1.

Tabela 1 - Erros relativos percentuais (Caso 1).

M	*V*	*n*						
		2	3	4	5	6	7	8
2	1	5.3	10.7	6.7	10.5	21.0	32.2	12.2
	2	0.0	0.0	0.0	0.3	3.7	0.6	0.0
	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1	6.8	17.0	20.1	25.3	39.0	28.5	29.2
	2	0.3	1.9	6.3	7.4	8.5	15.3	13.5
	3	0.1	0.4	2.3	3.4	2.9	7.7	6.3
	4	0.0	0.0	0.9	0.4	1.5	4.0	4.8
	5	0.0	0.0	0.3	0.5	0.2	3.4	3.8
4	1	11.1	23.7	29.1	31.3	43.4	48.2	52.7
	2	1.6	7.3	8.7	7.7	17.0	23.4	27.5
	3	0.2	3.0	4.4	3.9	11.1	11.0	14.7
	4	0.0	1.4	2.1	1.4	5.6	7.2	10.9
	5	0.0	0.8	0.7	0.7	3.6	4.9	8.7

Analisando-se a Tabela 1, pode-se verificar que, à medida que aumentamos a redundância *V*, menor será o erro relativo percentual para todas as dimensões testadas. Além disto, é possível notar que *n=2* gera o menor erro percentual, para todas as dimensões testadas. Os resultados mostram ainda que, à medida que a dimensão aumenta, deve-se aumentar a redundância para manter o erro relativo percentual nulo. Tomando-se, como exemplo, o caso de *n=2*, mostrado na Tabela 2, pode-se notar que, para manter o erro relativo percentual nulo, para dimensão *M=2* basta ter *V=2*, enquanto que para dimensão *M=4*, é necessário *V=4*.

Tabela 2 - Erro percentual para o caso de *n=2* (Caso 1).

M	*V*				
	1	2	3	4	5
2	5.3	0.0	0.0	0.0	0.0
3	6.8	0.3	0.1	0.0	0.0
4	11.1	1.6	0.2	0.0	0.0

Sendo L_i o número de vetores do subdicionário SD_i, pode-se definir **eficiência** como $E = L / (L_c + \Sigma_i p_i L_i)$, onde p_i é a probabilidade de usar o subdicionário SD_i para codificar $\boldsymbol{x}$. Cabe ressaltar que p_i foi aproximada pela freqüência relativa.

Os valores da eficiência *E* estão apresentados na Tabela 3. Nota-se que *E* diminui com o aumento da redundância, para todas as dimensões testadas. Observa-se, também, que *n=3* gerou uma eficiência numericamente menor que no caso de *n*=2, considerado ótimo do ponto de vista da distorção.

Tabela 3 - Eficiência (Caso 1).

M	*V*	*n*						
		2	3	4	5	6	7	8
2	1	3.2	2.6	4.4	6.4	8.4	10.7	13.2
	2	1.7	1.7	2.5	3.5	4.5	5.6	7.2
	3	1.2	1.4	1.9	2.5	3.2	4.1	5.0
3	1	7.4	3.9	10.4	14.3	19.0	20.8	20.5
	2	3.8	2.5	5.8	8.2	11.3	13.3	14.5
	3	2.5	2.0	4.1	5.9	8.3	10.2	11.2
	4	1.9	1.8	3.3	4.7	6.6	8.3	9.4
	5	1.6	1.6	2.8	4.0	5.6	7.1	8.2
4	1	14.3	6.2	17.2	17.4	14.3	10.7	8.1
	2	7.3	3.8	10.5	11.9	10.6	8.3	6.3
	3	4.2	3.1	7.9	9.3	8.9	7.0	5.3
	4	3.7	2.6	6.4	7.8	7.8	6.3	4.7
	5	3.0	2.4	5.4	6.9	7.0	5.7	4.4

CASO 2: TESTE DA DISTORÇÃO COM A VARIAÇÃO DA REDUNDÂNCIA V.

Para esse caso, foram mantidos constantes a dimensão *M* e o número de partições *n*, enquanto foi variada a redundância *V*.

Foram realizadas simulações com os seguintes parâmetros:

- *M=3*; *n=2, 3, 4*.
- *M=4*; *n=2, 3*.

Para *M=3* e *n=2*, a redundância máxima foi naturalmente limitada em $V_{max}=8$. Para os demais casos, forçou-se $V_{max}=10$.

Como no caso anterior, em cada simulação foram usados um dicionário *D* com 8192 vetores e um conjunto de entradas $\boldsymbol{x}$ com 2048 vetores.

As simulações foram repetidas 6 vezes para cada configuração.

Os valores (média de 6 simulações) dos erros relativos percentuais obtidos estão apresentados na Tabela 4.

Analisando-se a Tabela 4, pode-se verificar que:

- Aumentando-se a dimensão *M*, a distorção aumenta.
- Aumentando-se o número de partições *n*, a distorção aumenta.
- O aumento da redundância *V*, diminui a distorção.

Tabela 4 - Erros relativos percentuais (Caso 2).

M	*n*	*V*									
		1	2	3	4	5	6	7	8	9	10
3	2	5.0625	0.1778	0.0530	0.0000	0.0000	0.0000	0.0000	0.0000	----	----
	3	12.9604	2.5762	0.4601	0.0087	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	4	18.1652	3.3693	0.8984	0.2586	0.0161	0.0000	0.0000	0.0000	0.0000	0.0000
4	2	10.5346	1.6011	0.6019	0.0607	0.0231	0.0087	0.0067	0.0000	0.0000	0.0000
	3	17.1632	4.1260	1.7074	0.7115	0.3671	0.1297	0.0300	0.0085	0.0085	0.0000

5. CONCLUSÕES

Os resultados apresentados na Seção 4 mostram que a idéia de dividir o dicionário original em subdicionários, com o acréscimo de redundância, é eficiente para minimizar o tempo de busca, mantendo a distorção.

No caso de vetores criados com base em uma distribuição gaussiana e assumindo-se a segmentação proposta para o dicionário original, as simulações efetuadas permitem concluir que:

- Aumentando-se a dimensão *M*, a distorção aumenta.
- Aumentando-se o número de partições *n*, a distorção aumenta.
- O aumento da redundância *V*, diminui a distorção.

6. TRABALHOS FUTUROS

O desempenho alcançado com a técnica proposta justifica estudos posteriores, onde seja possível verificar, por exemplo: i) o uso de subdicionários circulares e hexagonais, ii) o uso de um número de partições *n* diferentes em cada dimensão e iii) dicionários e vetores de teste uniformemente distribuídos, a fim de garantir o desempenho do algoritmo para distribuições e partições genéricas.

7. BIBLIOGRAFIA

[1] K. L. Oehler and R. M. Gray, "Combining image classification and image compression using vector quantization", in Proceedings of the 1993 IEEE Data Compression Conference (DCC), J.A. Storer and M. Cohn, Eds., Snowbird, Utah, IEEE Computer Society Press, pp. 02-11, March 1993.

[2] Huilian Liao, Zhen Ji and Q.H.Wu , "A Novel Genetic Particle-Pair Optimizer for Vector Quantization in Image Coding," IEEE Congress on Evolutionary Computation 2008, pp. 708-713, June 2008.

[3] M. Atallah, Y. Genin, W. Szpankowiski, "Pattern Matching Image Compression: Algorithmic and Empirical Results", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, pp. 618-627, 1999.

[4] A. Gersho, "On the Structure of Vector Quantizers", IEEE Transactions on Information Theory, IT-28, pp. 157-165, March 1982.

[5] T. Kanungo, D. M. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, pp. 881-892, 2002.

[6] Ping Yu and A.N Venetsanopoulos,. "Improved hierarchical vector quantization for image compression," Proceedings of the 36th Midwest Symposium on Circuits and Systems, pp. 92-95, vol.1, August 1993.