KUANTISASI VEKTOR : DEFINISI, DISAIN DAN KINERJA

Ikhwana Elfitri
Jurusan Teknik Elektro Unand
E-mail : ikhwana@ft.unand.ac.id

ABSTRAK

Kuantisasi vektor telah digunakan secara luas pada berbagai bidang diantaranya adalah pada speech dan image processing. Karena aplikasinya yang luas, maka pada makalah ini ditampilkan studi literatur tentang prinsip dasar kuantisasi vektor, algoritma dan beberapa parameter yang harus dipertimbangkan dalam memilih vector quantizer. Dua tipe dasar kuantisasi vektor disampaikan beserta kelebihan dan kekurangan masing-masing tipe tersebut.

1. PENDAHULUAN

Kuantisasi vektor (*vector quantisation*) yang biasa disingkat dengan VQ telah terbukti secara teoritis mempunyai keunggulan dibandingkan dengan kuantisasi skalar (SQ). Shanon telah menunjukkan bahwa VQ mampu menghasilkan distorsi yang lebih rendah dibandingkan dengan SQ [1]. Beberapa penelitian telah memperlihatkan bahwa untuk dimensi kecil ataupun dimensi yang lebih besar, VQ lebih optimal dibandingkan dengan SQ [2,3].

Dalam berbagai bidang seperti speech dan image processing, VQ telah diaplikasikan secara luas dan berhasil digunakan untuk mendapatkan rate kuantisasi yang lebih rendah, tanpa menurunkan kualitas data. Beberapa pengkode suara yang telah distandarkan oleh ITU seperti pada [4,5,6] juga telah menggunakan VQ untuk kuantisasi parameter LPC. Karena itulah, pada makalah ini ditampilkan kajian literatur tentang prinsip dasar, keunggulan dan tipe VQ.

2. DEFINISI KUANTISASI VEKTOR

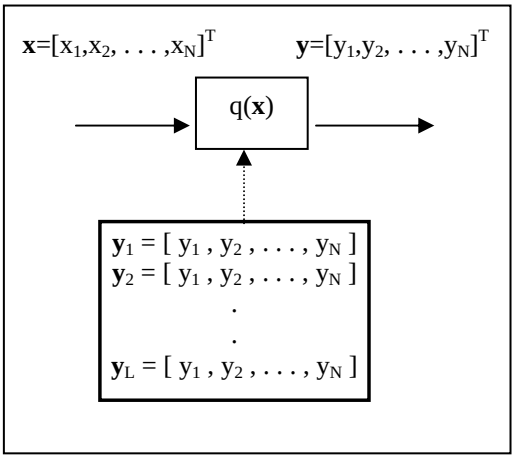
Sebuah vektor $\mathbf{x} = [\, x_1 , x_2 , \ldots , x_N \,]^T$ dimensi-N, akan dikuantisasi secara vektor. Setiap komponen dari vektor $\mathbf{x}$ $\{ x_k , 1 \le k \le N \}$ adalah variabel acak yang bernilai riil dengan amplituda kontinyu (*real-valued, continuous-amplitude random variables*). Dengan kuantisasi vektor maka vektor **x** akan dipetakan ke sebuah vektor **y** dimensi-N yang komponennya juga bernilai riil dengan amplituda diskrit. Ini berarti bahwa **x** dikuantisasi sebagai **y** dan **y** adalah nilai kuantisasi dari **x**. Proses kuantisasi (pemetaan) ini dapat dituliskan sebagai berikut [7] :

$$\mathbf{y} = q(\mathbf{x}) \qquad \ldots\ldots(1)$$

dimana :

- **y** vektor rekonstruksi (*reconstruction vector*) atau vektorkode (*codevector*).
- **X** adalah vektor masukan (*input vector*)
- q merupakan operator kuantisasi.

Seperti terlihat pada gambar 1 vektorkode **y** merupakan salah satu dari satu set vektor $\mathbf{Y} = \{ \mathbf{y}_i , 1 \le i \le L \}$ yang telah tersedia pada memori quantizer, dimana $\mathbf{y}_i = [\mathbf{y}_{i1}, \mathbf{y}_{i2} , \ldots, \mathbf{y}_{iN}]^T$. Set vektor **Y** disebut dengan bukukode (*codebook*). Parameter L adalah ukuran bukukode dan $\{\mathbf{y}_i\}$ adalah satu set vektorkode.



Gambar-1 Ilustrasi cara kerja vektor quantizer

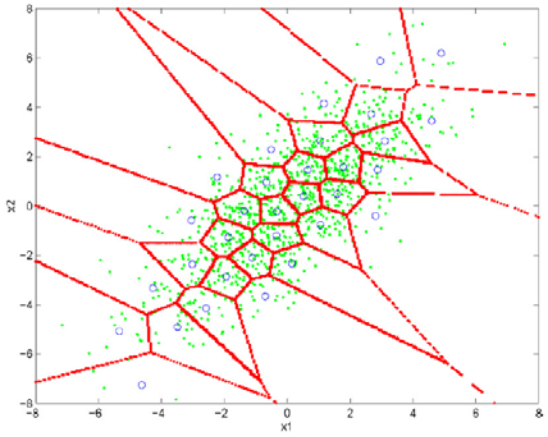
Agar dapat melakukan kuantisasi terhadap vektor masukan maka kita harus mempunyai bukukode yang sesuai dengan karakteristik vektor masukan. Sehingga quantizer vektor yang kita punya dapat mengeluarkan nilai aproksimasi yang baik untuk setiap vektor masukan. Baik buruknya nilai aproksimasi ini akan terlihat dari besar kecilnya kesalahan kuantisasi yang dihasilkan.

3. BUKU KODE

Dalam mendisain sebuah quantizer vektor, langkah pertama adalah mendisain sebuah bukukode. Setelah bukukode ditentukan barulah dilakukan disain quantizer. Jenis dan cara kerja quantizer akan sangat tergantung kepada jenis bukukode yang digunakan.

Untuk mendisain sebuah bukukode, ruang vektor dimensi-N dari vektor masukan **x** dipartisi menjadi

L sel (*region* atau *cell*) { C_i , $1 \le i \le L$ }. Untuk setiap sel C_i dipilih sebuah vektor $\mathbf{y}_i$ yang juga berada didalam sel tersebut sebagai representasi dari sel C_i . Proses partisi ruang vektor masukan menjadi sel-sel akan menentukan baik buruknya kualitas hasil kuantisasi.



Gambar-2 Sebuah ruang vektor dimensi 2, memperlihatkan vektor input (titik), vektorkode (lingkaran) dan sel (hyperplane) [8].

Jika bukukode telah ditentukan maka quantizer bekerja sebagai berikut. Langkah pertama quantizer adalah menentukan pada sel manakah sebuah vektor masukan berada. Quantizer akan mengeluarkan nilai kuantisasi $\mathbf{y}_i$ apabila vektor $\mathbf{x}_i$ berada didalam sel C_i .

4. DISAIN BUKU KODE

Disain bukukode dilakukan dengan menggunakan sejumlah besar data empirik (*emprical data*) berupa vektor masukan yang dinamakan vektor training. Vektor training ini haruslah memiliki karakteristik yang sama (menyerupai) vektor masukan quantizer. Proses disain dengan vektor training ini dinamakan dengan training bukukode (*training the codebook*).

Seperti telah dinyatakan sebelumnya bahwa untuk mendisain L-level bukukode maka ada 2 pekerjaan utama yang harus dilakukan yaitu [7] :

1. Partisi ruang vektor dimensi N menjadi L sel {Ci , $1 \le i \le L$}dan kemudian menempatkan vektor training kedalam sel-sel yang ada.
2. Menentukan vektor $\mathbf{y}_i$ yang merupakan representasi (centroid) dari sel tersebut .

Kedua proses diatas dilakukan secara berulang (iterasi). Karena itu proses disain bukukode bagi quantizer tertentu, biasanya dilakukan dengan algoritma iterasi.

Training bukukode membutuhkan sebuah bukukode sebagai kondisi awal. Sehingga untuk quantizer tertentu, bukukode yang dihasilkan dapat berbeda nilai vektorkode dan nilai distorsinya, tergantung kondisi awal yang digunakan.

Untuk itu proses disain harus menghasilkan bukukode yang optimal. Ada 2 jenis bukukode optimal yaitu :

- *Global optimal codebook*
- *Local optimal codebook*

Sebuah bukukode dikatakan sebagai *global optimal codebook* (distorsi minimum) jika parameter distorsinya bernilai minimum untuk semua L-level quantizer. Untuk mendapatkan bukukode yang bersifat *global optimal* maka dilakukan beberapa kali proses disain dengan menggunakan bukukode awal yang berbeda-beda.

Dengan proses disain yang seperti ini maka dapat dipilih sebuah bukukode yang menghasilkan distorsi paling kecil. Bukukode inilah yang dianggap sebagai *global optimal codebook*. Training bukukode dengan algoritma iterasi dapat menghasilkan sebuah bukukode yang memiliki distorsi minimal untuk bukukode awal tertentu. Setiap proses disain dengan algoritma iterasi seharusnya menghasilkan *local optimal codebook*.

5. ALGORITMA LBG

Salah satu metode dasar yang digunakan untuk disain bukukode adalah algoritma iterasi yang dikenal sebagai algoritma Lloyd (*K-means algorithm* atau *Lloyd algorithm*). Belakangan algoritma ini dikembangkan dan kemudian dikenal dengan LBG algorithm. Algoritma Lloyd membagi satu set vektor training menjadi L sel .

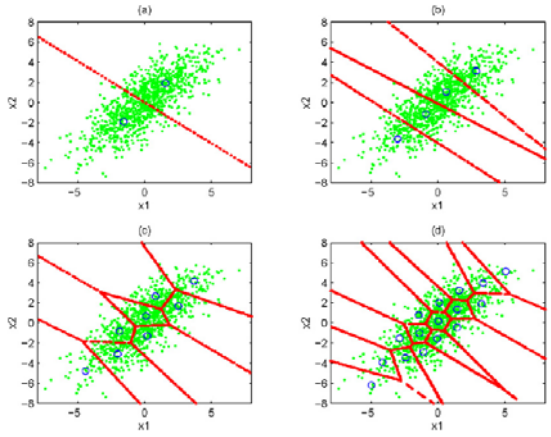
Secara sederhana, algoritma Lloyd dapat dijelaskan sebagai berikut [7] :

1. *Inisialisasi*. Set m=0 (iterasi ke-m). Pilih satu set vektorkode $\mathbf{y}_i$ (0), $1 \le i \le L$. (Sebuah bukukode awal).
2. *Klasifikasi*. Kelompokkan satu set vektor training { $\mathbf{x}$(n), $1 \le n \le M$ } kedalam L sel dengan menggunakan aturan nearest neighbor :
   $\mathbf{x} \in C_i$ (m), jika d[$\mathbf{x}$,$\mathbf{y}_i$ (m)] $\le$ d[$\mathbf{x}$,$\mathbf{y}_j$ (m)], untuk semua $j \ne i$.
3. *Perbaharui vektorkode*. Ganti m menjadi m+1. Hitung vektorkode yang baru untuk setiap sel dengan menggunakan prinsip centroid.
   $\mathbf{y}_i$ (m) = **centroid**[C_i (m)], $1 \le i \le L$.
4. *Tes untuk terminasi iterasi*. Jika penurunan distorsi D(m) pada iterasi ke-m relatif terhadap D(m-1) lebih kecil dari nilai treshold tertentu maka iterasi dihentikan. Jika tidak maka kembali ke langkah nomor dua.

Untuk menjamin agar setiap algoritma iterasi dapat menghasilkan sebuah bukukode yang optimal (*local optimal*) maka harus dipenuhi dua keadaan yang dirumuskan dengan aturan Nearest Neighbor dan aturan Centroid.

Algoritma LBG menyempurnakan algoritma Lloyd dengan menambahkan proses splitting untuk mendapatkan buku kode awal. Centroid dari semua vektor input di-split menjadi dua vektorkode. Selanjutnya satu set vektor training dibagi dua berdasarkan aturan nearest neighbor. Centroid dari

kedua cluster ini kemudian diiterasi dengan algoritma Lloyd sehingga diperoleh dua vektorkode untuk 1 bit quantizer. Proses ini diiterasi kembali sehingga didapatkan vector quantizer yang diinginkan.



Gambar-3 Disain 4 bit vector quantizer dengan LBG : (a) setelah split pertama dan iterasi dg Lloyd; (b) setelah split kedua dan iterasi dg Lloyd; (c) setelah split ketiga dan iterasi dg Lloyd; (d) setelah split terakhir dan iterasi dg Lloyd [8].

6. KINERJA VECTOR QUANTIZER

Proses kuantisasi dilakukan dengan menghitung distorsi antara vektor masukan **x** dengan setiap codevector dan kemudian memilih codevector dengan nilai distorsi yang terkecil sebagai nilai kuantisasi bagi **x**. Proses kuantisasi seperti ini dikenal dengan *full search* karena setiap codevector harus dihitung nilai distorsinya terhadap nilai vektor masukan. Untuk L-level quantizer dibutuhkan sebanyak L perhitungan distorsi, sedangkan perhitungan nilai distorsi untuk setiap codevector membutuhkan sebanyak N perkalian. Sehingga untuk setiap vektor masukan dibutuhkan sebnyak N × L perkalian. Besaran ini dinamakan dengan *computational cost.* Jika setiap kodevector dialokasikan B = RN = $\log_2$ L bit untuk transimisi maka computational cost menjadi :

$$\wp = N.\ 2^{RN} \text{.....(2)}$$

Ini berarti bahwa computational cost akan naik secara eksponensial seiring dengan pertambahan dimensi vektor dan jumlah bit per dimensi vektor.

Besaran lain yang perlu diperhatikan adalah *memory cost* yang menggambarkan banyaknya memory yang dibutuhkan untuk menyimpan codevector. Dengan asumsi satu memory dibutuhkan untuk menyimpan satu parameter vektor, maka untuk vektor dimensi-N dibutuhkan L × N memory sehingga memory cost adalah :

$$\aleph = N.L = N.\ 2^{RN} \text{.....(3)}$$

Sama seperti computational cost, memory cost juga akan bertambah secara eksponensial seiring dengan pertambahan dimensi dan jumlah bit per dimensi vektor.

6.1 Binary Search VQ

Untuk mengatasi kompleksitas perhitungan dan memory pada full-search codebook telah dikembangkan beberapa algoritma (*fast-search algorithm*). Tentu saja hal ini akan sedikit mengurangi performansi (kinerja) quantizer. Diantara algoritma tersebut akan dibahas disini beberapa diantaranya yaitu binary search dan multistage VQ.

Dengan menggunakan K-means algorithm dibutuhkan full-search algorithm pada L codevector untuk mengkuantisasi setiap vektor masukan. Pada binary search, dilakukan partisi cell secara bertingkat (hierarki). Sehingga proses searching untuk mendapatkan codevector yang memiliki distorsi minimum tidak dilakukan pada semua codevector tetapi hanya dilakukan pada $\log_2$ L codevector.

Ruang vektor dimensi N terlebih dahulu dipartisi menjadi 2 region atau cell menggunakan K-means algorithm dengan nilai K=2. Kemudian masing-masing cell dibagi menjadi 2 sub cell sehingga terbentuk 4 sub cell. Masing-masing sub cell dibagi lagi menjadi dua dan demikian seterusnya sehingga akan didapat sejumlah L cell. Ini berarti jumlah cell akan berharga pangkat dua yaitu $L=2^p$, dengan p adalah jumlah tingkat partisi. Untuk setiap cell pada setiap partisi dicari nilai centroid-nya.

Pada tahap pertama $\mathbf{v}_1$ dan $\mathbf{v}_2$ adalah centroid cell 1 dan cell 2. Pada tingkat kedua terdapat 4 sub cell dengan masing-masingnya memiliki centroid $\mathbf{v}_3$ sampai $\mathbf{v}_6$. Pada tingkat ketiga telah terdapat 8 cell sehingga centroid masing-masing sub cell akan berfungsi sebagai codevector $\mathbf{y}_i$.

Sebuah vektor masukan **x** akan dikuantisasi dengan cara memilih lintasan pada skema pohon (tree) diatas sehingga akan menghasilkan distorsi terkecil. Sebagai contoh misalkan sebuah vektor **x** dibandingkan terhadap $\mathbf{v}_1$ dan $\mathbf{v}_2$. Jika $d(\mathbf{x},\mathbf{v}_2) < d(\mathbf{x},\mathbf{v}_1)$ maka akan dipilih lintasan menuju $\mathbf{v}_2$. Kemudian **x** dibandingkan terhadap $\mathbf{v}_5$ dan $\mathbf{v}_6$. Jika, misalkan, $d(\mathbf{x},\mathbf{v}_5) < d(\mathbf{x},\mathbf{v}_6)$ maka dipilih lintasan yang menuju $\mathbf{v}_5$. Terakhir **x** dibandingkan terhadap $\mathbf{y}_5$ dan $\mathbf{y}_6$. Jika $d(\mathbf{x},\mathbf{y}_6) < d(\mathbf{x},\mathbf{y}_5)$ maka $\mathbf{y}_6$ dipilih sebagai nilai kuantisasi dari **x**.

Dari proses diatas dapat dilihat bahwa jumlah perhitungan distorsi adalah sebanyak 2 $\log_2$ L . Jika diasumsikan terdapat N perkalian untuk setiap perhitungan distorsi maka akan diperoleh computational cost sebesar :

$$\wp = 2 \log_2 L \times N = 2N \log_2 L = 2NB \text{.....(4)}$$

B = jumlah bit / codevector.

Pada algoritma binary search ini akan diperoleh pengurangan cukup besar pada computational cost

namun memory cost justru akan meningkat. Karena setiap vektor centroid pada setiap tingkat ($\mathbf{v}_1$ sampai $\mathbf{v}_6$) juga akan membutuhkan memory. Sehingga akan diperoleh memory cost :

$$\aleph = 2N (L–2).…….....……(5)$$

Binary search codebook sebagaimana dijelaskan diatas termasuk dalam jenis *uniform binary search*. Karena setiap cell pada setiap tingkat selalu dipartisi menjadi dua sub cell. Sehingga akan didapat bentuk skema tree yang balance (seimbang). Pada *nonuniform binary search*, cell yang hanya mengandung sedikit vektor training tidak akan dipartisi lebih lanjut sehingga bentuk skema tree bisa menjadi non uniform (non balance).

6.2 Multistage VQ

Multistage VQ merupakan suatu algoritma yang dapat memperkecil computational cost dan memory cost. Namun demikian harga computational cost yang dihasilkan masih lebih besar dari harga computational cost pada binary search codebook [2,5].

Multistage VQ (dinamakan juga dengan cascaded VQ) terdiri dari beberapa urutan codebook yang masing-masingnya bekerja pada residual dari codebook sebelumnya.

Vektor masukan $\mathbf{x}$ pertama kali dikuantisasi dengan menggunakan B_1 bit (L_1 -level) quantizer vektor. Residual atau error $\mathbf{e}$ antara $\mathbf{x}$ dan nilai kuantisasinya $\mathbf{z}_i$ kemudian dijadikan sebagai masukan bagi B_2 bit (L_2 -level) quantizer vektor kedua yang memberi keluaran vektor $\mathbf{w}_j$. Hasil akhir kuantisasi dari 2-tahap quantizer vektor ini adalah penjumlahan dari kedua vektor $\mathbf{z}_i$ dan $\mathbf{w}_j$:

$$q(\mathbf{x}) = \mathbf{y} = \mathbf{z}_i + \mathbf{w}_j \text{ ……...……(6)}$$

Computational cost dan Memory cost untuk 2-tahap quantizer ini adalah sebagai berikut :

$$\wp = N (L_1 + L_2)……...….(7)$$

$$\aleph = N (L_1 + L_2)……...……..(8)$$

7. KESIMPULAN

Pada makalah ini telah ditampilkan teori dasar kuantisasi vektor dilengkapi dengan proses disain menggunakan algoritma LBG. Beberapa tipe bukukode yang digunakan untuk vector quantizer juga dibahas dilengkapi dengan pembahasan tentang kinerjanya.

DAFTAR PUSTAKA

[1] Shannon, C.E., 1959, Coding Theorems for A Discrete Source with A Fidelity Criterion, IRE National Convention Record, Part 4, pp. 142-163

[2] Gray, R.M., Neuhoff D.L., 1998, Quantization, IEEE Trans. Inform. Theory 44 (6), pp. 2325-2383

[3] Lookabaugh, T.D., Gray, R.M., 1989, High-Resolution Quantisation Theory and The Vector Quantizer Advantage, IEEE Trans. Inform. Theory 35 (5), pp. 1020-1033

[4] ITU-T Recommendation G.722, 1988, 7 kHz audio coding within 64 kbits/s

[5] ITU-T Recommendation G.722.1, 1995, Coding at 24 and 32 kbits/s for hands-free operation in systems with low frame loss.

[6] ITU-T Recommendation G.722.2, 2002, Wideband coding of speech at around 16 kbit/s using Adaptive Multi-Rate Wideband (AMR-WB)

[7] Elfitri I., 2002, Pengkode Suara dengan Kuantisasi Vektor, Thesis Magister, Institut Teknologi Bandung

[8] So, S., Paliwal, K.K., 2007, Efficient product code vector quantization using the swicthed split vector quantiser, Digital Signal Processing, Vol. 17, No. 1, pp. 138-171