INDEXING AND ENTROPY CODING OF LATTICE CODEVECTORS

Adriana Vasilache and Ioan Tăbuş

Tampere University of Technology
Signal Procesing Laboratory
PO Box 553, 33101 Tampere, Finland
email: adrianav@cs.tut.fi, tabus@cs.tut.fi

ABSTRACT

We present here two methods of entropy coding for the lattice codevectors. We compare our entropy coding methods with one method previously presented in the literature from the point of view of rate-distortion as well as of the computation complexity and memory requirements. The results are presented for artificial Laplacian and Gaussian data, as well as for LSF parameters of speech signals. In the latter case, the multiple scale lattice VQ (MSLVQ) is used for quantization, which reduces the rate gain of the entropy coding method when compared with the fixed rate case, but allows a dynamic allocation of the bits in the whole speech coding scheme.

1. INTRODUCTION

Lattice vector quantization (LVQ) has attracted interest as an alternative to optimal vector quantization (VQ) due to its reduced memory requirements for storage of the codebook and low complexity of the encoding and decoding. To make the LVQ more competitive from the rate-distortion point of view the lattice codevectors should be entropy coded.

Practical methods of lattice codevectors entropy coding have been proposed, with applications mainly in image coding [1], [2]. These methods use a partitioning of the lattice codevectors into classes and then encode separately the class index sequence with Huffman or arithmetic coding and the position of codevectors within classes with fixed rate. Due to the fixed rate encoding part, there always exists a gap between the entropy of the codevectors and the average code-length of the resulting code, even though there have been attempts to reduce it [3].

Besides a new method of entropy coding using codevectors partitioning, we also propose the use of the canonical Huffman coding of the lattice codevectors with reduced memory requirements and having a low computational complexity for the entropy encoding and decoding. The second proposed method is based on the canonical Huffman coding and on a suitably indexing method for the lattice codevectors. Several methods of entropy coding of lattice codevectors (including the proposed ones) are tested on Laplacian and Gaussian data as well as on LSF coefficients of speech signals.

2. LATTICE VECTOR QUANTIZATION

A n-dimensional lattice, Λ, can be considered as a set of vectors that form a group under ordinary addition in $\mathbb{R}^n$. Geometrically, the lattice Λ can be thought of as an infinite regular array of points that uniformly fills the n-dimensional space.

When used as a VQ, the lattice has to be truncated to the finite number of vectors allowed by the finite rate, R. Our goal is to deal with sources whose distribution possesses a certain symmetry property (e.g spherical, pyramidal) and, therefore, we consider truncations reflecting those symmetries. A truncated lattice is defined as the set of lattice points having the norm less or equal to a given value, K:

$$\overline{\Lambda} = \{x \in \Lambda | N(x) \leq K\} \tag{1}$$

where $N(x)$ is the selected norm of x. If $N(x)$ is the Euclidean norm the truncation will be spherical and in the case of the l_1 norm the truncation will be pyramidal [4]. The lattice points will be grouped on spherical or pyramidal shells. The number of points on each shell can be calculated by means of the θ or other equivalent series, for both spherical and pyramidal [4] types of truncations. For truncations corresponding to other types of norms (e.g. $l_{1/2}$ norm) the enumeration can be done using the leader class cardinalities [5]. The codebook, $\mathcal{C}$, is a scaled truncated lattice:

$$\mathcal{C} = \{s \cdot c_j | c_j \in \bar{\Lambda}, s \in \mathbb{R}\} \tag{2}$$

where s is the scaling factor.

3. ENTROPY CODING OF LATTICE CODEVECTORS

3.1. Entropy coding of lattice codevectors using codevectors partitioning

The principle that has been used in several methods for lattice codevectors entropy coding is to group the codevectors into some classes. The indexes of the classes are entropy coded using for instance Huffman coding and the position of a codevector within a class is encoded using fixed rate enumerative encoding. The use of the fixed rate encoding is justified only if the elements of a class are decorrelated. The best code-length that can be obtained with a memoryless method is given by the entropy of the classes indexes, $H_{\mathcal{I}}$, plus the average code length used for the positions of the codevectors within a class:

$$L_{min} = H_{\mathcal{I}} + \sum_{l=1}^{n} p_l L_l \tag{3}$$

where n is the number of classes, I_l is the index and p_l the probability of the class l. $\mathcal{I} = \{I_l, l = \overline{1,n}\}$ is the set of class indexes and L_l is the number of bits needed to encode the position of

This work has been supported by Academy of Finland, project No. 44876 (Finnish Centre of Excellence Program (2000-2005))

one codevector belonging to the class l, within that class. As the positions of the codevectors are fixed-rate encoded and the cardinalities of the classes aren't integer powers of 2, there, inevitably, exist some gap between the average length of an optimal entropy code of the lattice codevectors and the average code-length given by (3).

The most common method is to group the codevectors into classes according to their norm [1], [2]. As the class corresponding to the norm zero is quite frequent it is customary to apply run-length coding and entropy coding on the sequence of norm indexes. This method has been improved by Kim [3] in the sense of reducing the gap between the optimal code-length and the real one, by redistributing radius and index sequences. Another way to group the lattice codevectors is according to their weight (number of non-zero components) [6].

Entropy coding using leader classes

We propose here, as an alternate approach, the use of leader classes as the method of grouping the lattice codevectors. A leader or leader vector is a vector belonging to the lattice and whose components are unsigned and sorted in decreasing order from left to right [5]. The leader class of the leader vector v is composed of all the vectors obtained by signed permutations (with some possible constraints) of the leader vector. As mentioned in [5], the leader class can be this way defined only if the considered lattice is invariant under permutation.

The performance of all these entropy coding methods for lattice codevectors is highly dependent on the distribution of the codevectors in classes. We will propose in what follows a method that attains the optimal average code-length for the lattice codevectors and also has low memory requirements and low complexity.

3.2. Canonical Huffman coding of lattice codevectors

The canonical Huffman coding (CHC) method [7] is known for its low complexity of decoding and low memory requirements. For instance, regarding the memory requirements, the data that has to be stored consists of a table of nL integers where nL is the number of different code lengths in the code and a permutation giving the index of the input alphabet symbols in the decreasing probability order. For large input alphabets the storage of this permutation is a limitation.

In the case of the lattice codevectors, as the number of codevectors can be very large, this coding method seems to be out of the question. However, with a suitably chosen indexing method, that would give the codevectors indexes in their decreasing probability order, we could still apply the canonical Huffman using a small amount of memory. We will present first the indexing method and then, how the canonical Huffman encoding procedure is simplified by using the proposed indexing.

3.2.1. Lattice codevectors indexing

Using the definition of a leader class presented in the section 3.1, an enumeration algorithm for the codevectors of a truncated lattice can be rapidly developed. For this we count the permutations used to obtain each vector from a leader class as well as the signs distribution among the non-zero components of the vectors.

Enumeration algorithm

Stored data:

$base\_index\_table$ - a table of L integers containing the indexes of the first vectors of each leader class (L is the number of leader classes)

C - a triangular matrix $[(N+1) \times (N+1)]$ containing the binomial coefficients (N is the space dimension).

Input: x - n dimensional lattice vector,
ℓ - leader class index to which x belongs.

Output: I - index of codevector x.

1. **if** $(x == (0, 0, \ldots, 0))$ $I = 0$, **stop**
2. $base\_index = base\_index\_table[\ell]$
3. $idx\_sign = code\_sign(x)$
4. $idx\_max = number(x)$
5. $index = code\_pos(x, ...)$
6. $I = base\_index + idx\_sign * idx\_max + index$

where:
$code\_sign(x)$ is a function that assigns to each non-zero component of x 0 or 1 with respect to its sign and transforms the resulting sequence of bits into an integer.
$number(x)$ counts how many vectors can be obtained through permutation of the absolute values of the components of x.
$code\_pos(x, \ldots)$ encodes the position of the absolute values of non-zero components of x.

Retrieval of the codevector from an index

Stored data:

$base\_index\_table$ - a table of L integers containing the indexes of the first vectors of each leader class (L is the number of leader classes)

C - a triangular matrix $[(N+1) \times (N+1)]$ containing the binomial coefficients.

$leaders\_table$ - a $[L \times N]$ matrix containing the leader vectors.

Input: I - index of codevector x.

Output: x - n dimensional lattice vector.

1. Find $base\_index$ and ℓ from $base\_index\_table$.
2. $I = I - base\_index$.
3. $idx\_max = number(leaders\_table(\ell))$
4. $idx\_sign = \text{round}\left(\frac{I}{idx\_max\_table[\ell]}\right)$.
5. $index = \text{rem}\left(\frac{I}{idx\_max\_table[\ell]}\right)$.
6. Determine unsigned vector
   $\hat{x} = decode\_vec(leaders\_table(\ell), index)$
7. Determine signed vector
   $x = decode\_sign(\hat{x}, idx\_sign)$

where rem is the remainder function, $decode\_vec$ is the inverse of the $code\_vec$ function and $decode\_sign$ is the inverse of the $code\_sign$ function.

3.2.2. Entropy coding using canonical Huffman

The code obtained with the canonical Huffman procedure has the property that when listed in the increasing order of the symbols probability, the corresponding code-lengths are increasing and for the same code-length the symbols are listed such that their codewords are consecutive binary numbers [7].

For a source with independent components it is natural to consider that the codevectors of a lattice codebook, that belong

No. lead.	EC on R			EC on L		
	L_{HR}	L_{idxR}	L_R	L_{HL}	L_{idxL}	L_L
3	1.00	7.98	8.98	1.23	7.31	**8.54**
8	1.08	12.59	13.67	2.02	11.10	**13.20**
9	1.51	13.00	14.51	2.41	11.42	**13.83**
12	1.52	14.81	16.33	2.83	12.94	**15.77**
13	1.43	15.19	16.62	2.92	13.25	**16.17**
20	1.54	15.06	16.60	3.23	13.16	**16.39**
21	1.74	15.73	17.47	3.44	13.60	**17.04**

Table 1. Code length distribution between the class index and the position of codevectors within a class for the entropy coding with partitioning on norms (R) and on leaders (L) for Laplacian data.

to the same leader class in the lattice have the same probability. Thus if we know how the leader classes are ordered with respect to their codevector probabilities, we can use the indexing method described in the previous subsection, to have the codevectors indexed in their increasing probability order. For this we must adapt the variable $base\_index\_table$ such that the leader classes are ordered with respect to their vectors probabilities. Therefore, besides the memory requirements for indexing, only an additional table of integers, of length equal to the number of different code length in the resulting code is needed.

4. RESULTS

We compare the results, in terms of the average code length, for two methods of entropy coding of lattice codevectors using partitioning (on norms and on leader classes) and for the canonical Huffman coding. We have considered first some generated data (Gaussian and Laplacian) after which the LSF prediction errors have been used as input for the lattice quantizer.

The samples in the artificial data set are independent and the results for these cases are reported over 500000 samples of data. The optimal scale (giving the minimal mean squared quantization error) is considered for each of the codebooks. It is obtained with an empiric optimization method [5].

4.1. Laplacian data

The lattice D_{10} is considered for quantization. It is pyramidally truncated up to a given number of leader classes. The average code-length for the lattice codevectors entropy coding with partitioning on norms and on leader classes are given in the Table 1. $L_{HR(L)}$ is the average code-length obtained with Huffman coding for the class index sequence and $L_{idxR(L)}$ is the average code-length corresponding to the sequence of codevectors positions within a class. $L_{R(L)}$ is the resulting average code-length for the two coding methods. From the results of Table 2 it is clear that the canonical Huffman coding of lattice codevectors is, as it was expected, the best method in terms of average code-length.

4.2. Gaussian data

The lattice D_{10} is considered for quantization. It is spherically truncated up to a given number of leader classes. The average code-length for the lattice codevectors entropy coding with partitioning on norms and on leader classes are given in the Table 3. $L_{HR(L)}$, $L_{idxR(L)}$, $L_{R(L)}$ have the same significance as for the

No. lead.	H	CHC L_H	ECR L_R	ECL L_L	Enum. coding
3	7.59	**7.65**	8.98	8.54	8
8	11.53	**11.85**	13.67	13.20	13
9	13.38	**13.41**	14.51	13.83	15
12	15.07	**15.10**	16.33	15.77	16
20	15.56	**15.59**	16.60	16.39	17
21	16.18	**16.20**	17.47	17.04	18

Table 2. Comparison of canonical Huffman coding (CHC) with entropy coding with partitioning of lattice codevectors on norms (ECR) and on leaders (ECL) for different sizes of a pyramidal LVQ on Laplacian data. The entropy H is also given for every considered case.

No. lead.	EC on R			EC on L		
	L_{HR}	L_{idxR}	L_R	L_{HL}	L_{idxL}	L_L
3	1.01	7.94	8.95	1.03	7.88	**8.92**
4	1.10	11.58	12.68	1.12	11.53	**12.65**
7	1.31	13.37	14.68	1.69	12.96	**14.65**
8	1.76	14.43	**16.19**	1.90	14.31	16.21

Table 3. Code length distribution between the class index and the position of codevectors within a class for the entropy coding with partitioning on norms (R) and on leaders (L) for Gaussian data.

case of Laplacian data. From the results of Table 4 the canonical Huffman coding of lattice codevectors is the best method in terms of average code-length.

As it was expected a definite winner method within the partitioning entropy coding methods cannot be indicated.

With respect to the memory requirements the three methods are comparable since for the partitioning methods two integer tables must be stored: one for the Huffman coding of the class index sequence and a second one for the number of bits needed for the fixed rate encoding of the codevectors positions within a class.

4.3. LSF quantization using MSLVQ with spherical truncation of the lattice D_{10}

The LPC parameters are extracted from the input speech signal and transmitted in order to reconstruct at the decoder the short-

No. lead.	H	CHC L_H	ECR L_R	ECL L_L	Enum. coding
3	7.57	**7.64**	8.95	8.92	8
4	11.77	**11.82**	12.68	12.65	12
7	14.20	**14.27**	14.68	14.65	15
8	15.66	**15.72**	16.19	16.21	16

Table 4. Comparison of canonical Huffman coding (CHC) with entropy coding with partitioning of lattice codevectors on norms (ECR) and on leaders (ECL) for different sizes of a spherical LVQ on Gaussian data. For the two latter methods the best achievable average code-length (equation (3)) is considered. The entropy H is also given for every considered case.

term spectral envelope of the decoded signal. The LPC parameters are the coefficients of the p-th order prediction polynomial $A(z)$, obtained by applying the Levinson algorithm to a frame of speech signal. At the decoder, the decoded excitation is filtered by the all-pole filter $H(z) = 1/A(z)$. The order of $A(z)$ is commonly taken $p = 10$. The LPC parameters are transformed into an equivalent set, the line spectrum pairs (LSF) and their quantized values are transmitted to the receiver. The process of quantizing the filter parameters to a finite number of bits/frame is referred to as LPC quantization.

The spectral distortion is often used as an objective measure of the encoding performance:

$$SD = \left\{ \frac{10}{\pi} \int_0^{\pi} \left[\log_{10} |A_n(e^{j\omega})|^2 - \log_{10} |\hat{A}_n(e^{j\omega})|^2 \right]^2 d\omega \right\}^{1/2} \quad (4)$$

where SD is given in dB, $A_n(z)$ and $\hat{A}_n(z)$ are the predictors of the $n-th$ speech frame without and with quantization respectively.

We experimented the predictive multiple-scale lattice VQ [4] for the quantization of LPC parameters, by utilizing the same LPC computation procedure used in G.729 codec [8]. There, a 10-th order LPC analysis with 10 ms analysis frame is employed, based on the auto-correlation method and the use of a 30 ms Hamming window. 4 copies of the same truncated lattice D_{10} are used in the MSLVQ structure. The 4 scales are optimized on 115006 frames using an empiric optimization method [5]. The probabilities of the codevectors are estimated using the speech data from TIMIT training set (1417087 frames). As the number of codevectors (considered individually for every scale) is 857617, not every codevector is necessarily chosen in the quantization process. Thus to estimate the probability of each codevector the average over each leader class is considered. As the 4 scales are not equiprobable, the ranking of the codevectors with respect to their probability for each of the scales differ. Therefore for the indexing method needed for the canonical Huffman coding the table $base\_index\_table$ must be 4 times bigger (which means in our case 52 integers instead of 13 integers). The comparison between the entropy coding methods is presented in Table 5. As we no longer have a uniform quantizer due to the MSLVQ structure, the gain with respect to the fixed rate coding code-length is smaller. The canonical Huffman permits the gain of about a bit. There are 13 distinct code-lengths which means that for entropy coding of the lattice codevectors a table of 13 integers is needed which is nearly insignificant. As for the performance of the method in terms of rate distortion for the LPC quantization we have obtained a mean spectral distortion of 0.98 dB for a rate of 19.11 bits which is better that the variable rate LPC quantization proposed in [9] where a spectral distortion below 1 dB was obtained for 20 bits. As an additional observation, the variable rate coding would permit a dynamic allocation of the bits between the LSF coefficients quantization and the coding of the excitation.

5. CONCLUSION

We have presented in this paper two novel methods for entropy coding of lattice codevectors. The first one is a particular case of the entropy coding methods using codevectors partitioning and its performance is similar to other equivalent methods. The second method is based on the canonical Huffman coding, whose reduced memory requirements and low computation complexity are assured by a suitably chosen indexing method of the codevectors that we propose. We have compared our entropy coding methods with one previously presented in the literature. The results are presented for artificial Laplacian and Gaussian data as well as for LSF parameters of speech signals. The memory and computation requirements are similar but the best average code-length is obtained with the canonical Huffman coding. In the case of LSF quantization the multiple scale lattice VQ (MSLVQ) is used for quantization, which reduces somewhat the rate gain compared with the fixed rate case, but also allows a dynamic allocation of the bits in the whole speech coding scheme and has a better performance in terms of rate-distortion than other variable rate methods proposed in the literature for LSF quantization.

Zero order entropy	19.08 bits
CHC	**19.11** bits
EC on norms	19.75 bits
EC on leaders	19.54 bits
Enum. coding	20.00 bits

Table 5. Average code length for different methods of entropy coding for lattice codevectors of a MSLVQ structure containing 4 copies of a lattice truncated up to the 13^{th} spherical leader (857617 codevectors). The mean spectral distortion is 0.98dB.

6. REFERENCES

[1] Z.M. Yusof and T.R. Fischer, "An entropy-coded lattice vector quantizer for transform and subband image coding," *IEEE Trans. Image Process.*, vol. 5, pp. 289–298, 1996.

[2] A. Woolf and G. Rogers, "Lattice vector quantization of image wavelet coefficient vectors using a simplified form of entropy coding," in *Proc. of ICASSP'94*, 1994, vol. 5, pp. V/269–V/272.

[3] W.-H. Kim, "Adaptive lossless coding scheme of lattice vector quantisation," *IEE Proc. Vis. Image Signal Process.*, vol. 146, no. 6, pp. 317–325, December 1999.

[4] A. Vasilache, M. Vasilache, and I. Tăbuş, "Predictive multiple-scale lattice VQ for LSF quantization," in *Proc. of ICASSP'99*, March, 15-19 1999, pp. 657–660.

[5] A. Vasilache, B. Dumitrescu, and I. Tăbuş, "Multiple scale leader-lattice VQ with application to LSF quantization," submitted to Signal Processing, 2000.

[6] M. Khataie and M.R. Soleymani, "Indexing the output points in a lattice based vector quantizer," in *Proc. of the Canadian Conference on Electrical and Computer Engineering*, 1995, vol. 1, pp. 152–155.

[7] I.H. Witten, A. Moffat, and T.C. Bell, *Managing gigabytes - compressing and indexing documents and images*, Morgan Kaufman Publishers, San Francisco, California, 1999.

[8] "Coding of speech at 8kbit/s using conjugate-structure algebraic-code-excited linear-prediction (CS-ACELP)," ITU Telecommunications Standardization Sector, December 1995, Draft Recommandation.

[9] S. Sridharan and J. Leis, "Two novel lossless algorithms to exploit index redundancy in VQ speech compression," in *Proc. of ICASSP'98*, Seattle, USA, May 1998.