Optimization of Product Code

Shin-ichi Maeda and Shin Ishii
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630--0192, JAPAN

Abstract: - Vector quantization is known as the best lossy source coding among the fixed-to-fixed coding methods because of its satisfactory ability of expression. Although it can represent any fixed-to-fixed code and has optimization methods that guarantee local optimality, its encoding and optimization require the computation that grows exponential to the data length. We propose an optimization method for a product coding, which avoids this computational explosion problem by applying a reasonable restriction to the model architecture. The performance of the product coding is evaluated by a simple problem.

Key-Words: - lossy source code, product code, companding function, optimization

1 Introduction

In this article, we propose an optimization method for lossy source coding. The lossy source coding aims at encoding original data within the minimum code length while the distortion between the original and reproduction data is kept low. Any fixed-to-fixed coding can be represented by vector quantization, suggesting that the vector quantization has the best coding ability among the fixed-to-fixed coding schemes. Moreover, various optimization methods of vector quantization have been developed, such as generalized Lloyd algorithm (GLA) [1]. However, the encoding computation required by the vector quantization is proportional to the number of reproduction vectors M. When the original data length is n and the data is encoded with a rate R bit per one data length, the number of reproduction vectors is $M = 2^{Rn}$. Then, the encoding computation grows exponentially to the original data length; so-called "curse of dimensionality" occurs. Since the encoding is necessary also in optimization, the optimization suffers from computation explosion when the original data length increases.

The increase of the computation originates from the expansion of the original data space $\Re^n$. Product coding is a method that replaces the vector quantization by component-wise scalar quantization [2]. This replacement works greatly to reduce the space of the optimization. We introduce a compressor, which transforms the original data space to a feature space, and an expander, which transforms the feature space again to the original data space, such that the simple scalar quantizers exhibit the best. In this article, we propose an optimization method for the compressor and the expander, while the scalar quantizers are fixed to uniform scalar quantizers. This optimization process enables the product coding to show the best coding performance for the given data.

2 Architecture of Product Code

Fig. 1 depicts the architecture of the product code we propose.

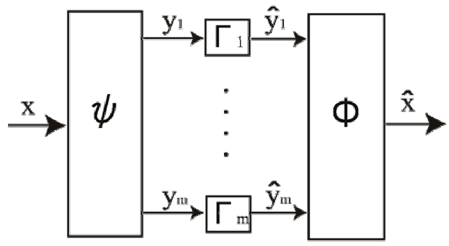

Fig. 1. Architecture of product code

Let $\mathbf{x} = [x_1, \cdots, x_n]^T \in \Re^n$ be an original datum and $\hat{\mathbf{x}} \in \{\hat{\mathbf{x}}\} \equiv \left\{ \hat{\mathbf{x}}^1, \cdots, \hat{\mathbf{x}}^M \mid \hat{\mathbf{x}}^k = [\hat{x}_1, \cdots, \hat{x}_n]^T \in \Re^n, k = 1, \cdots, M \right\}$ be a reproduction datum. First, a compressor ψ transforms the original data space into an m-dimensional feature space $\Re^n \to [0,1]^m$ such that $\mathbf{y} = \psi(\mathbf{x}) = [y_1, \cdots, y_m]^T$ denotes the transformed datum of the original datum $\mathbf{x}$. Next, a quantizer Γ quantizes the feature vector $\mathbf{y}$ into $\hat{\mathbf{y}} = \Gamma(\mathbf{y})$. The quantizer Γ consists of m uniform scalar quantizers, $\Gamma_1, \cdots, \Gamma_m$, each of which quantizes the corresponding element of $\mathbf{y}$ as $\hat{y}_i = \Gamma_i(y_i)$ within the quantizer level a_i. The quantizer levels $a_1, \cdots, a_m$ are related to the number of reproduction data M as $M = \prod_{i=1}^{m} a_i$. The uniform quantizer Γ_i we use is a piece-wise one as shown in Fig. 2.

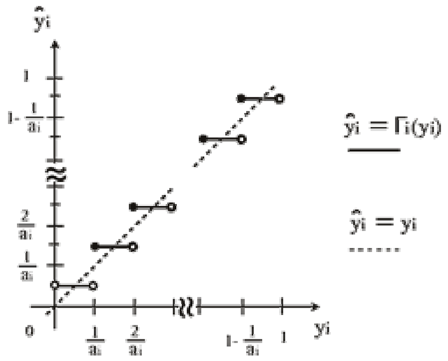

Fig. 2. Quantizer $\hat{y}_i = \Gamma_i(y_i)$

Then, an expander ϕ transforms the *m*-dimensional feature space into the original data space $[0,1]^m \to \Re^n$. The reproduction vector $\hat{\mathbf{x}}$ is hence obtained by an expander ϕ as $\hat{\mathbf{x}} = \phi(\hat{\mathbf{y}})$.

According to our product coding, the compressor and the expander are parameterized as $\psi(\mathbf{x};\theta_\alpha)$ and $\phi(\hat{\mathbf{y}};\theta_\beta)$, respectively, and parameters θ_α and θ_β are optimized such that the original data are encoded with the minimum distortion under the limitation of a given coding length $\log_2 M$.

3 Optimization Algorithm

Given coding length $\log_2 M$, the cost function is described as

$$\min_{\theta_\alpha, \theta_\beta} E[d(\mathbf{x}, \phi(\Gamma(\psi(\mathbf{x};\theta_\alpha));\theta_\beta)], \quad (1)$$

where $d(\mathbf{x}, \hat{\mathbf{x}})$ denotes a distortion measure, which is always positive $d(\mathbf{x}, \hat{\mathbf{x}}) > 0$ excepting the case $d(\mathbf{x}, \hat{\mathbf{x}}) = 0$ if and only if $\mathbf{x} = \hat{\mathbf{x}}$. $E[\cdot]$ denotes an expectation over the original data distribution $p(\mathbf{x})$. Since this source distribution $p(\mathbf{x})$ is often unknown in practice, the cost function (1) is approximated by using samples from the distribution $p(\mathbf{x})$ as

$$E[d(\mathbf{x}, \phi(\Gamma(\psi(\mathbf{x};\theta_\alpha));\theta_\beta)] \approx \frac{1}{N}\sum_{i=1}^{N} d(\mathbf{x}^i, \phi(\Gamma(\psi(\mathbf{x}^i;\theta_\alpha));\theta_\beta), \quad (2)$$

where $\mathbf{x}^i$ denotes the *i*-th sample and N the total sample number.

We here present an iterative procedure for minimizing the cost function (2). At the first step, we fix the parameter θ_α and optimize the parameter θ_β. Then, this step becomes

$$\min_{\theta_\beta} \sum_{i=1}^{N} d(\mathbf{x}^i, \phi(\hat{\mathbf{y}}^i;\theta_\beta)). \quad (3)$$

If $\phi(\hat{\mathbf{y}};\theta_\beta)$ is a differentiable function with respect to the parameter θ_β, the local optimal for θ_β is obtained by the steepest descent or Newton's method.

At the second step, we fix the parameter θ_β and optimize the parameter θ_α. Then, this step becomes

$$\min_{\theta_\alpha} \sum_{i=1}^{N} d(\mathbf{x}^i, \phi(\Gamma(\psi(\mathbf{x}^i;\theta_\alpha)))). \quad (4)$$

Since this cost function is not differentiable due to the discrete function Γ, we cannot apply a gradient-based optimization method like the steepest descent or Newton's method. So, we approximate each discrete function Γ_i as a continuous function:

$$\Gamma_i(y_i) \approx y_i. \quad (5)$$

The dotted line in Fig. 2 depicts the approximated function. This approximation provides an exact estimate when the quantizer level a_i becomes large. Using this approximation (5), the cost function (4) becomes

$$\min_{\theta_\alpha} \sum_{i=1}^{N} d(\mathbf{x}^i, \phi(\psi(\mathbf{x}^i;\theta_\alpha))), \quad (6)$$

which can be optimized by a gradient-based method if the compressor $\psi(\mathbf{x};\theta_\alpha)$ is a differentiable function with respect to the parameter θ_α. In particular, if the expander ϕ is an invertible function, the best compressor ψ that minimizes the cost function (6) is the inverse of the expander ϕ, $\psi = \phi^{-1}$, because the distortion takes its minimum value $d(\mathbf{x}, \phi(\psi(\mathbf{x};\theta_\alpha))) = d(\mathbf{x}, \mathbf{x}) = 0$.

4 Simulation

We tested the performance of our product coding by conducting a simple simulation experiment. The test original data is shown in Fig. 3.

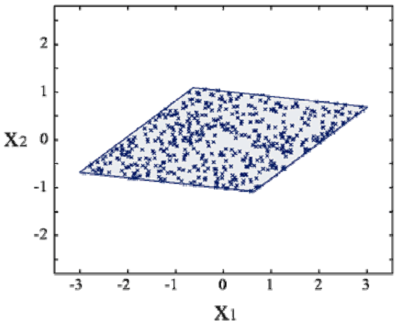

Fig. 3. The test original data (1000 samples)

The test data set consisted of 10000 samples, each obtained by a 2×2 linear transformation from a random sample taken from a two-dimensional uniform distribution. The distortion measure we use is a square error: $d(\mathbf{x},\hat{\mathbf{x}})=\frac{1}{2}\sum_{i=1}^{2}(x_i-\hat{x}_i)^2$.

The product code is parameterized as

$$\psi(\mathbf{x};\theta_\alpha)=f(A^{-1}\mathbf{x}), \quad (7)$$

$$\phi(\hat{\mathbf{y}};\theta_\beta)=Ag(\hat{\mathbf{y}}), \quad (8)$$

where A is a 2×2 linear matrix and A^{-1} is its inverse matrix. The functions $f(\cdot)$ and $g(\cdot)$ consist of component-wise scalar functions $f_i(\cdot)$ and $g_i(\cdot)$ ($i=1,2$), respectively. The scalar function $f_i(\cdot)$ is defined as

$$f_i(x_i)=\begin{cases} 1 & (1<x_i) \\ \frac{1}{2}(x_i+1) & (-1\le x_i\le 1) \\ 0 & (1<x_i) \end{cases}, \quad (9)$$

and $g_i(\cdot)$ is defined as

$$g_i(\hat{y}_i)=2\hat{y}_i-1, (0\le \hat{y}_i\le 1). \quad (10)$$

The compressor $\psi(\mathbf{x};\theta_\alpha)$ is set to the inverse function of $\phi(\hat{\mathbf{y}};\theta_\beta)$. The parameters θ_α and θ_β to be optimized here correspond to the linear matrix A. The matrix that minimizes the cost function (3) is analytically obtained as $A^*=XZ^\dagger$ where $X=[\mathbf{x}^1,\cdots,\mathbf{x}^N]$, $Z=[\mathbf{z}^1,\cdots,\mathbf{z}^N]$, $\mathbf{z}^i=g\left(\Gamma\left(f(A^{-1}\mathbf{x}^i)\right)\right)$ and $Z^\dagger$ denotes the Moore-Penrose's pseudo-inverse of Z.

We compared the performance of our product code with the transform coding based on Karhunen-Loeve transformation (KLT). In this transform coding, an original datum is transformed by KLT such as $\mathbf{y}=K\mathbf{x}$. After the transformation, the transformed vector $\mathbf{y}$ is quantized by a scalar quantizer. In the limit of high rate coding, the uniform quantization of $F_i(y_i)$ is optimal among the scalar quantizations [3]. $F_i(y_i)$ is defined as $F_i(y_i)=\frac{1}{Z}\int_{-\infty}^{y_i}p_i(z)^{1/3}dz$ where $Z=\int_{-\infty}^{\infty}p_i(z)^{1/3}dz$ and the random variable z is a normalized variable of y_i divided by its variance σ_i, $z=\frac{y_i}{\sigma_i}$. When the *i*-th scalar quantizer level is a_i ($i=1,\cdots,m$), the bit required by the quantizer is $b_i=\lceil \log_2 a_i \rceil$. Bit b_i is allocated subject to the restriction $\sum_{i=1}^{m}b_i\le\log_2 M$ and $b_i\ge 0$. An optimal bit allocation in the limit of high rate coding [4] is given by $b_i=\bar{b}+\frac{1}{2}\log_2\frac{\sigma_i^2}{\rho^2}+\frac{1}{2}\log_2\frac{h_i}{H}$, where $\bar{b}=\frac{\log_2 M}{m}$, $\rho^2=\left(\prod_{i=1}^{m}\sigma_i^2\right)^{1/n}$, $h_i=\frac{1}{12}\left\{\int_{-\infty}^{\infty}p_i(z)^{1/3}dz\right\}^3$ and $H=\left(\prod_{i=1}^{m}h_i\right)^{1/m}$. Since this bit allocation does not necessarily give an integer solution, an appropriate integer bit allocation is explored among the adjacent integers of the obtained allocation solution.

The result is shown in Fig. 4. In each figure, reproduction data are denoted by circles. A blank circle indicates that the reproduction vector encodes some original data, while a filled circle indicates that the reproduction vector encodes no original data. The dotted lines denote the directions of the feature vectors: y_1 and y_2. Figs. 4(a) and 4(c) show the results by our product coding and Figs. 4(b) and 4(d) show the results by the KLT-based transform coding. Figs. 4(a) and 4(b) show the results of 4 bit encoding, and Figs. 4(c) and 4(d) show those of 6 bit encoding.

The title of each figure denotes the mean squared error (mse), i.e., the achieved cost function. As can be seen in Fig. 4, the product coding allocates reproduction vectors, which the original data are transformed independently into and hence the reproduction vectors are not wasted, while the KLT-based transform coding wastes them.

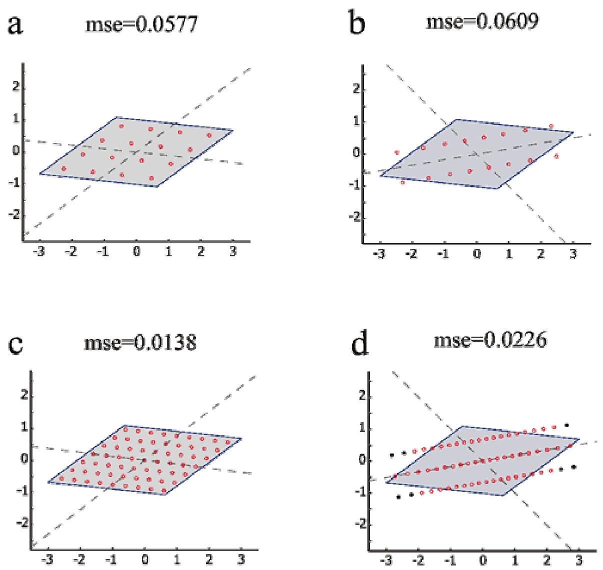

Fig. 4. Encoding results of our product coding after the optimization (a) in 4 bit encode, (c) in 6 bit encode, and those of the KLT-based transform coding (b) in 4 bit encode, (d) in 6 bit encode

We also compared the performance of the proposed product coding with that of the vector quantization optimized by GLA. The mse of the optimized vector quantization became 0.040 in 4 bit encoding or 0.010 in 6 bit encoding. Although these were better than those by the proposed product coding, 0.0577 in 4 bit encoding and 0.0138 in 6 bit encoding, if the product coding was modified into exploring the nearest neighbors like in the vector quantization, its performance greatly improved; the mse became 0.041 in 4 bit encoding or 0.010 in 6 bit encoding. These results indicate that the difference of the mse between the product coding and the vector quantization stems from the encoding methods. Nevertheless, our product coding has an advantage in the computation cost. The exploration of nearest neighbors used in the vector quantization requires the computation order of $o(n\prod_{i=1}^{m} a_i)$, while the encoding by the compressor ψ requires that of $o(nm)$.

5 Concluding Remarks

In our simple simulation, the optimized product coding performed better than the traditional transform coding. The KLT-based transform coding is optimal among the orthogonal transform coding when the source has a Gaussian distribution [5]. Although the KLT decomposes the original data such to have no correlation, the elements of the decomposed data are dependent on each other except for the Gaussian case. Our simple simulation indicated that there are better transformations that are not orthogonal, and our product coding is able to find such a transformation as to be an independent transformation. Product coding includes wide range of coding schemes such as shape-gain vector quantization, traditional transform coding, or even tree-structured vector quantization. The optimization method for more complex product coding scheme will be discussed in our future work.

References:

[1] Y. Linde, A. Buzo and R. M. Gersho, “An algorithm for vector quantizer design”, *IEEE Trans. Comm.*, COM-28, 1980, pp. 84-95.

[2] W. Y. Chan and A. Gersho, “Generalized product code vector quantization: A family of efficient techniques for signal compression”, *Digital Signal Processing*, 4(2), 1994, pp. 95-126.

[3] W. R. Bennet, “Spectra of quantized signals”, *Bell Syst. Tech. J.*, 27 1948, pp. 446-472.

[4] J.-H. Huang and P. M. Schultheiss, “Block quantization of correlated gaussian random variables”, IEEE Trans. Comm., CS-11, 1963, pp. 289-296.

[5] V. K. Goyal, “Theoretical foundations of transform coding”, IEEE Signal Processing Mag., 18(5), 2001, pp. 9-21.