Reduced-Memory Decoding of Low-Density Lattice Codes

Brian Kurkoski, *Member, IEEE,* and Justin Dauwels, *Member, IEEE*

***Abstract*—This letter describes a belief-propagation decoder for low-density lattice codes of finite dimension, in which the messages are represented as single Gaussian functions. Compared to previously-proposed decoders, memory is reduced because each message consists of only two values, the mean and variance. Complexity is also reduced because the check node operations are on single Gaussians, avoiding approximations needed previously, and because the variable node performs approximations on a smaller number of Gaussians. For lattice dimension n =1000 and 10,000, this decoder looses no more than 0.1 dB in SNR, compared to the decoders which use much more memory.**

***Index Terms*—Low-density lattice codes, lattice decoding, belief-propagation decoding.**

I. Introduction

LOW-DENSITY lattice codes (LDLC) are lattices which are constructed and decoded on principles similar to low-density parity-check codes. They can be decoded in large dimension, and error-free decoding is possible within 0.6 dB of the unconstrained-power channel capacity. The belief-propagation LDLC decoder passes functions, rather than real numbers as are used by low-density parity-check code decoders. These messages may be approximated by a discretely-quantized function, however this requires a large amount of storage, typically 1024 values per message [1]. In subsequent work, similar performance was obtained by instead approximating this function with a mixture of Gaussian functions. A variable number of Gaussians were used, but typically no more than ten, represented by 30 values [2] [3].

This letter describes a decoder for finite-dimension LDLC lattices which uses single-Gaussian messages between the variable nodes and check nodes. Since a single Gaussian is described by a mean and a variance, the message consists of only two values. This is fewer values than the previously-proposed methods. In addition, check node complexity is reduced because the inputs are single Gaussians, and approximations are not required. However, as in prior work, a Gaussian mixture approximation is used internally at the variable node. But since the input is a single Gaussian (effectively a single shift-and-repeated Gaussian), complexity at the variable node is still lower than other methods which must find approximations of a larger number of Gaussians. Variable node complexity is reduced by using a forward-backward algorithm which avoids the creation of periodic Gaussian mixtures, by using the aperiodic channel message.

Single Gaussian messages have previously been used to decode infinite-dimensional LDLC lattices, that is, to find noise thresholds [4]. That decoder also used single Gaussians internally at the variable node. However, when applied to finite-dimension lattices it has poor performance. Thus, an observation is that when decoding finite-dimension LDLC lattices, while a single Gaussian is sufficient to approximate the messages between the variable nodes and check nodes, that good performance is more sensitive to the approximations that occur internally at the variable node.

Manuscript received December 3, 2009. The associate editor coordinating the review of this letter and approving it for publication was Z. Yan.

B. Kurkoski is with the Department of Information and Communications Engineering, University of Electro-Communications, Tokyo, 182-8585, Japan (e-mail: kurkoski@ice.uec.ac.jp).

J. Dauwels is with the Department of ECE, Massachusetts Institute of Technology, Cambridge, MA.

B. Kurkoski was supported in part by the Ministry of Education, Science, Sports and Culture; Grant-in-Aid for Scientific Research (C) number 21560388. J. Dauwels was supported in part by post-doctoral fellowships from the King Baudouin Foundation and the Belgian American Educational Foundation (BAEF). Part of this work was carried out at the RIKEN Brain Science Institute, Saitama, Japan.

Digital Object Identifier 10.1109/LCOMM.2010.07.092350

II. Gaussian Mixtures

A Gaussian function, or just "Gaussian" in this letter, with mean m and variance v is denoted as $\mathcal{N}(z; m, v) = (2\pi v)^{-1/2} \exp\left(-\frac{(z-m)^2}{2v}\right)$. A message $f(z)$ is a mixture of N Gaussians:

$$f(z) = \sum_{i=1}^{N} c_i \mathcal{N}(z; m_i, v_i)\,, \tag{1}$$

where $c_i \geq 0$ are mixing coefficients with $\sum_{i=1}^{N} c_i = 1$. An alternative representation is a list of N triples of means, variances and mixing coefficients, $\{t_1, \ldots, t_N\}$ with $t_i = (m_i, v_i, c_i)$.

To approximate a true distribution $p(z)$, by another distribution $q(z)$, a reasonable approach is to choose $q(z)$ so that the Kullback-Leibler divergence [5, p. 18] is minimized. When $q(z)$ is a single Gaussian, selecting its mean and variance to be the same as those of $p(z)$ will minimize the divergence; this is sometimes known as "moment matching" [6, Appendix A]. In particular, if $p(z)$ is a mixture of N Gaussians $t_i = (m_i, v_i, c_i)$, $i = 1, \ldots, N$, then the single Gaussian with mean m and variance v which minimizes the divergence is given by:

$$m = \sum_{i=1}^{N} c_i m_i \quad \text{and} \tag{2}$$

$$v = \sum_{i=1}^{N} c_i (v_i + m_i^2) - m^2. \tag{3}$$

This single Gaussian t found by moment matching, denoted by MM, is:

$$t = (m, v, 1) = \text{MM}(t_1, \ldots, t_N). \tag{4}$$

A previously-presented Gaussian mixture reduction (GMR) algorithm approximates $f(z)$, a mixture of N Gaussians, by

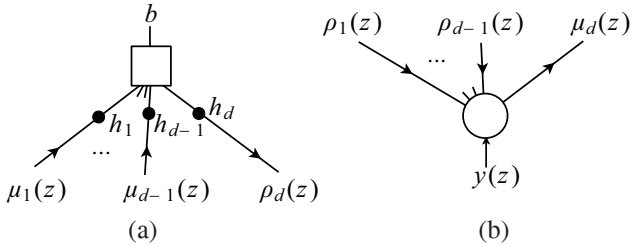

Fig. 1. Messages for (a) check node and (b) variable node decoding functions.

another mixture, $g(z)$, which consists of M Gaussians, where $M < N$ [2]. This algorithm is denoted as:

$$g(z) \quad = \quad \mathrm{GMR}\big(f(z)\big). \tag{5}$$

This algorithm uses the squared distance as a metric between two Gaussians, and greedily combines pairs of Gaussians until M Gaussians remain.

III. Single-Gaussian Decoding of LDLC Lattices

A. Setup

An n-dimensional LDLC lattice Λ is a lattice for which H, the inverse of the n-by-n generator matrix, is sparse; the generator matrix is $G = H^{-1}$. If $\mathbf{b} = (b_1, \ldots, b_n)^{\mathrm{t}}$ (t denotes transpose) with $b_i \in \mathbb{Z}$ is any integer vector, then $G\mathbf{b} \in \Lambda$.

An arbitrary element $\mathbf{x} \in \Lambda$, with $\mathbf{x} = (x_1, \ldots, x_n)$, is transmitted over an AWGN channel with noise variance σ^2. As in previous LDLC lattice studies, this model ignores the input power constraint to concentrate on the lattice coding gain. The received sequence is $\mathbf{y} = (y_1, \ldots, y_n)$.

A decoder finds an estimated lattice point $\widehat{\mathbf{x}}$ and a corresponding integer sequence $\widehat{\mathbf{b}}$, given $\mathbf{y}$. The belief-propagation decoding algorithm may be presented on a bipartite graph, with n variable nodes and n check nodes. A regular matrix H has constant row and column weight d, so all check nodes and all variable nodes have degree d. The variable-to-check messages are functions $\mu(z)$, and the check-to-variable messages are functions $\rho(z)$. Associated with variable node i is the channel output y_i, and the channel message is $y_i(z) = \mathcal{N}\big(z; y_i, \sigma^2\big)$ (y_i is distinct from $y_i(z)$). The initial variable-to-check message for edge k is $\mu_k(z) = y_i(z)$, if edge k is connected to variable node i. Iterations for the single-Gaussian decoder proceed as follows.

B. Check node

A check node of degree d has input messages $\mu_k(z)$ which are single Gaussians with mean m_k and variance v_k, for $k = 1, \ldots, d$. Associated with each edge are the non-zero labels from H, denoted as h_k, as shown in Fig. 1-(a). The check node output $\widetilde{\rho}_k(z)$ is the convolution of these messages, which is a Gaussian with mean $\widetilde{m}_k$ and variance $\widetilde{v}_k$ given by:

$$\widetilde{m}_k \quad = \quad -\frac{1}{h_k} \sum_{i=1}^{d \backslash k} h_i \cdot m_i, \text{ and} \tag{6}$$

$$\widetilde{v}_k \quad = \quad \frac{1}{h_k^2} \sum_{i=1}^{d \backslash k} h_i^2 \cdot v_i. \tag{7}$$

The computation of (6) and (7) can be simplified by using a forward-backward recursion. The description is omitted, but is similar to the variable node recursion in the following.

C. Variable Node

A variable node of degree d has input messages $\widetilde{\rho}_k(z)$ which are single Gaussians with mean $\widetilde{m}_k$ and variance $\widetilde{v}_k$ for $k = 1, \ldots, d$; the channel message is $y(z)$, as shown in Fig. 1-(b).

In prior work, the shift-and-repeat (periodic extension) step, which creates a periodic Gaussian mixture, occurs at the check node. In this work, the decoder uses single Gaussian messages, so this function occurs at the variable node. The repetition of $\widetilde{\rho}_k(z)$ with period $1/|h_k|$, is $\rho_k(z)$, given by:

$$\rho_k(z) \quad = \quad \sum_{b \in \mathbb{Z}} \widetilde{\rho}_k(z - b/h_k) = \sum_{b \in \mathbb{Z}} \mathcal{N}(z; m_k(b), \widetilde{v}_k)\,, \tag{8}$$

where the mean of each Gaussian $m_k(b)$ is:

$$m_k(b) \quad = \quad \widetilde{m}_k + \frac{b}{h_k}. \tag{9}$$

Nominally, the sum in (8) is over all integers $\mathbb{Z}$. However, since this periodic extension occurs at the variable node, this sum can be restricted to those Gaussians which are near the channel message; periodic Gaussians far from the channel message have near-zero mixing coefficients and can be safely ignored. In the simulations in the next section, this sum is over $\{-1, 0, +1\}$ because the all-zeros lattice point is used.

Using the periodic $\rho_k(z)$, the variable node output $\mu_k(z)$ is the product:

$$\mu_k(z) \quad = \quad y(z) \prod_{i=1}^{d \backslash k} \rho_i(z). \tag{10}$$

Naive computation of (10) requires $d \cdot (d-1)$ message multiplications.

However, a forward-backward recursion can be used to reduce the complexity to $2 \cdot (d-1) + d$ message multiplications. Also, multiplying two distinct periodic messages $\rho_i(z)$ and $\rho_j(z)$ is computationally complex, because a large number of Gaussians with significant mixing coefficients are required. However, since the channel message is non-periodic, the product $y(z)$ and $\rho(z)$ is also non-periodic, and tend to be well-approximated by one or two Gaussians. To exploit this, a forward and backward recursion is used, initialized with the channel message. Since there is only one channel message, each recursion is initialized with $\sqrt{y(z)}$, so that the final product correctly includes $y(z)$.

To describe the forward-backward recursion in detail, denote a single message multiplication as:

$$\mathrm{V}\big(\alpha(z), \rho(z)\big) \quad = \quad \alpha(z) \cdot \rho(z). \tag{11}$$

The product $\mathrm{V}\big(\alpha(z), \rho(z)\big)$, in triples representation $\big\{(m_\ell, v_\ell, c_\ell)\big\}$, can be computed as follows. Let $\alpha(z)$ be a mixture of I Gaussians, and let $\mu(z)$ be a mixture of J Gaussians, represented as:

$$\Big\{\big(m_i^\alpha, v_i^\alpha, c_i^\alpha\big)\Big\}_{i=1}^{I} \quad \text{and} \quad \Big\{\big(m_j^\mu, v_j^\mu, c_j^\mu\big)\Big\}_{j=1}^{J}, \tag{12}$$

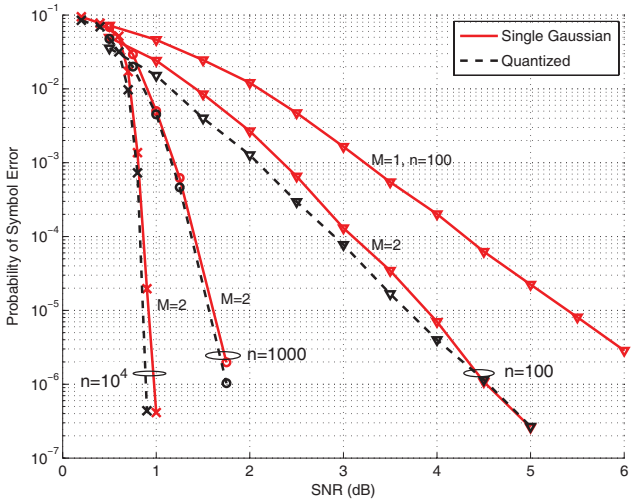

Fig. 2. Probability of symbol error for the single-Gaussian decoder and the quantized-message decoder. The Gaussians mixture size used internally at the variable node is M, and the lattice dimension is n.

respectively. The product is a mixture of $I \cdot J$ Gaussians, given by:

$$\frac{1}{v_\ell} = \frac{1}{v_i^\alpha} + \frac{1}{v_j^\mu}, \tag{13}$$

$$\frac{m_\ell}{v_\ell} = \frac{m_i^\alpha}{v_i^\alpha} + \frac{m_j^\mu}{v_j^\mu}, \text{ and} \tag{14}$$

$$c_\ell = \frac{c_i^\alpha c_j^\mu}{\sqrt{2\pi(v_i^\alpha + v_j^\mu)}} \exp\left(\frac{1}{2}\frac{(m_i^\alpha - m_j^\mu)^2}{v_i^\alpha + v_j^\mu}\right), \tag{15}$$

for $\ell = 1, \ldots, I \cdot J$, suitably indexed such as $\ell = J(i-1)+j$.

The forward recursion is initialized with $\alpha_0(z) = \sqrt{y(z)}$. The square root of a Gaussian distribution is another Gaussian distribution with the same mean and twice the variance, so in triples representation the initialization is $\{(y, 2\sigma^2, 1)\}$. Then each step of the forward recursion computes a product, and then applies the Gaussian-mixture approximation:

$$\widetilde{\alpha}_k(z) = \mathrm{V}\big(\alpha_{k-1}(z), \rho_k(z)\big), \text{ and} \tag{16}$$

$$\alpha_k(z) = \mathrm{GMR}(\widetilde{\alpha}_k(z)), \tag{17}$$

for $k = 1, 2 \ldots, d$.

The backward recursion is similarly initialized with $\beta_d(z) = \sqrt{y(z)}$, and computes messages $\beta_k(z)$, for $k = d-1, d-2, \ldots, 1$.

The variable node output is the single-Gaussian $\mu_k(z)$ found using moment-matching:

$$\mu_k(z) = \mathrm{MM}\big(\alpha_{k-1}(z) \cdot \beta_k(z)\big), \tag{18}$$

for $k = 1, \ldots, d$.

Instabilities are avoided by setting a minimum variance. A message variance v is replaced by $\max(v, \gamma)$, where γ is a constant. Values in the range $\gamma = 10^{-3}$ to 10^{-4} were used.

D. Hard Decisions

Hard decisions on the integers, $\widehat{\mathbf{b}}$, may be found by first making intermediate decisions $\widetilde{\mathbf{x}} = (\widetilde{x}_1, \ldots, \widetilde{x}_n)^{\mathrm{t}}$ at each variable node, and then finding $H\widetilde{\mathbf{x}}$. The intermediate decisions are found as,

$$\widetilde{x}_i = \arg\max_z y(z) \prod_{k=1}^{d} \rho_k(z). \tag{19}$$

For $\alpha_d(z)$ at node i, this $\widetilde{x}_i$ is the mean of the following single Gaussian found by moment matching:

$$\mathrm{MM}\big(\alpha_d(z)\sqrt{y_i(z)}\big), \tag{20}$$

since the maximum of a Gaussian occurs at its mean. Finally, $\widehat{\mathbf{b}}$ may be found by element-wise rounding the vector $H\widetilde{\mathbf{x}}$ to the nearest integer.

IV. Numerical Results

The single-Gaussian decoder was evaluated numerically on the unconstrained input power AWGN channel, and compared with the discretely-quantized message decoder with 1024 samples per message, with sample spacing 1/128 centered on zero. Lattices with dimension $n = 100$, 1000, and 10000 were used, and the matrix H had row and column weights d of 3, 7 and 7, respectively. The d non-zero entries in each row and column of H were $\pm\big(1, 1/\sqrt{d}, \cdots, 1/\sqrt{d}\big)$. The SNR is $\big(|\det G|^{2/n}\big)/2\pi e\sigma^2$, and the matrices were normalized so that $|\det G| = 1$. In this case, the capacity of the unconstrained input power AWGN channel corresponds to channel noise variance $\sigma^2 = 1/2\pi e$, that is, 0 dB [7]. A symbol error is the event that $\widehat{b}_i \neq b_i$.

As shown in Fig. 2, the losses for the single-Gaussian decoder with $M = 2$ are no more than 0.1 dB for dimension $n = 1000$ and 10000 lattices. For $n = 100$, this may increase to 0.3 dB at low SNR. When only one Gaussian is allowed internally at the variable node, that is $M = 1$, there was significant performance loss, as shown. But a mixture of two Gaussians was sufficient to obtain good performance.

Other LDLC lattice decoders use either 1024 values (quantized) or tens of values (Gaussian mixtures) for each message, but the decoder proposed in this letter uses only two values. Thus, memory requirements are reduced with minor loss of performance.

References

[1] N. Sommer, M. Feder, and O. Shalvi, "Low-density lattice codes," *IEEE Trans. Inf. Theory*, vol. 54, pp. 1561–1585, Apr. 2008.

[2] B. Kurkoski and J. Dauwels, "Message-passing decoding of lattices using Gaussian mixtures," in *Proc. IEEE International Symposium on Information Theory*, Toronto, Canada, July 2008.

[3] Y. Yona and M. Feder, "Efficient parametric decoder of low density lattice codes," in *Proc. IEEE International Symposium on Information Theory*, Seoul, Korea, pp. 744–748, June 2009.

[4] B. Kurkoski, K. Yamaguchi, and K. Kobayashi, "Single-Gaussian messages and noise thresholds for decoding low-density lattice codes," in *Proc. IEEE International Symposium on Information Theory*, Seoul, Korea, pp. 734–738, June 2009.

[5] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley, 1991.

[6] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

[7] V. Tarokh, A. Vardy, and K. Zeger, "Universal bound on the performance of lattice codes," *IEEE Trans. Inf. Theory*, vol. 45, pp. 670–681, Mar. 1999.