LDPC Decoding with Natural Redundancy

Pulakesh Upadhyaya and Anxiao (Andrew) Jiang
Computer Science and Engineering Department, Texas A&M University
pulakesh@tamu.edu, ajiang@cse.tamu.edu

I. Introduction

The increasingly wider application of non-volatile memories (NVMs) in big-data storage has led to a challenge: how to recover data from errors as effectively as possible for reliable long-term storage. Flash memories and other NVMs have noise mechanisms such as charge leakage, read/write disturbs, and cell-quality degradation due to P/E cycling. They make data more and more noisy over time. So there is a strong motivation in exploring new techniques for error correction.

In this paper, we study how to correct errors using *natural redundancy* (NR) in compressed data, and how to combine it with error-correcting codes (ECCs). By NR, we refer to the redundancy in data that is not artificially added for error correction, such as features in languages and images and structures in databases. It is a rich resource for error correction even after data are compressed (due to practical reasons such as high complexity in compression and data modeling), and is especially suitable for NVMs because an NR-decoder often needs to check the validity of candidate solutions (such as words/phrases for languages) in dictionaries, and the fast random access speed of NVMs makes it very efficient.

In an accompanying paper [1] (where a survey of related works is given), we have shown that after deep compression of languages, plenty of redundancy still remains. In this work, we also study it for images. We then propose a general scheme to combine NR-decoding with low-density parity-check (LDPC) codes, which are used very widely in NVMs. We explore the density evolution of LDPC decoding given information from the NR-decoder. We also propose a theoretical model for compressed languages, and study the performance of iterative decoding between the LDPC decoder and the NR-decoder. The results show that natural redundancy can substantially improve the error correction capability of LDPC codes.

II. NR-Decoding for Images and Languages

We have designed a natural-redundancy (NR) decoder for images. In particular, we focus on images of handwritten digits. They are from the National Institute of Standards and Technology (NIST) database, which have $70,000$ images as training or test data. We compress the bi-level images (of size $28{\times}28$ pixels) using run-length coding, where the run-lengths of 0s and 1s are compressed by two optimized Huffman codes, respectively. The rate is 0.27 bit/pixel.

Assume the compressed images have erasures. To decode noisy images, we have designed a convolutional neural network for recognizing noisy images, and also a specialized filter based on features of connected components in decompressed images. The NR-decoder is illustrated in Fig. 1 (a). The final step of decoding is: if all candidate solutions agree on a bit, set the bit to that value; otherwise, keep it as an erasure.

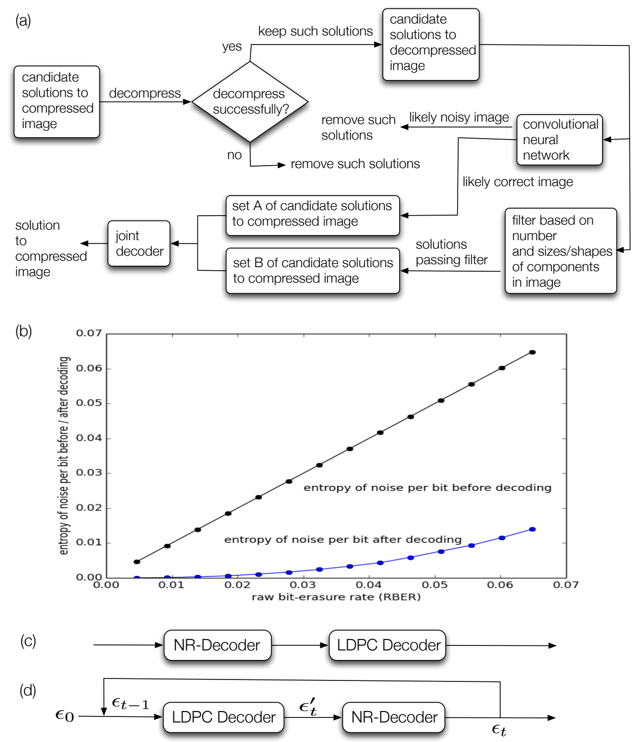

Fig. 1. (a) NR-decoder for images. (b) Performance of NR-decoder. (c) A concatenated decoding scheme. (d) An iterative decoding scheme.

The decoding performance can be measured as follows. Let $\epsilon \in [0,1]$ be the erasure probability before decoding. After the decoding by natural redundancy, let $\delta \in [0,1]$ denote the probability that an originally erased bit remains as an erasure, and let $\rho \in [0, 1-\delta]$ denote the probability that an originally erased bit is decoded to 0 or 1 incorrectly. Then the amount of noise after decoding can be measured by the entropy of the noise (erasures and errors) per bit: $E_{NR} \triangleq \epsilon(\delta + (1-\delta)H(\frac{\rho}{1-\delta}))$, where $H(p) = -p\log p - (1-p)\log(1-p)$ is the entropy function. We show E_{NR} in Fig. 1 (b). The NR-decoder reduces noise substantially: it removes noise effectively by over 75% for the compressed images (without any help from ECC), for raw bit-erasure rate (RBER) from 0.5% to 6.5%.

Similar performance can be obtained for languages. In [1], we have presented an NR-decoder for deeply compressed English texts. The NR-decoder reduces noise effectively (between 88.0% and 91.6%) for LZW-compressed texts, for raw bit-erasure rate from 5% to 30%.

III. Combine NR-decoding with LDPC Codes

We protect compressed data (languages or images) as information bits by a systematic LDPC code. The decoding process is a concatenation of two decoders: first, the NR-decoder decodes information bits, and outputs a partially corrected codeword with its updated soft information; then, the LDPC decoder takes that as input, and uses belief propagation (BP) for decoding. See Fig. 1 (c). We present a theoretical analysis for the decoding performance, and show that the NR-decoder can substantially improve the performance of LDPC codes.

Consider a binary-erasure channel (BEC) with erasure probability ϵ_0. (BSC can be analyzed similarly.) Let us call the non-erased bits *fixed bits*. Assume that after NR-decoding, an erased bit remains as an erasure with probability p_0, and becomes an error with probability $(1-p_0)\gamma_0$. We design the following iterative LDPC decoding algorithm, which generalizes both the peeling decoder for BEC and the Gallager B decoder for BSC: (1) let π and τ be two integer parameters; (2) in each iteration, for a variable node v that is an erasure, if π or more non-erased message bits come from check nodes and they all have the same value, set v to that bit value; (3) if v is not a fixed bit and not an erasure (but possibly an error) in this iteration, change v to the opposite bit value if τ or more non-erased message bits come from check nodes and they all have that opposite value. We analyze the density evolution for the decoding algorithm. (For $t = 0, 1, 2\cdots$, let α_t and β_t be the fraction of codeword bits that are errors or erasures, respectively, after t iterations of LDPC decoding. We have $\alpha_0 = \epsilon_0(1-p_0)\gamma_0$ and $\beta_0 = \epsilon_0 p_0$. Let $\kappa_0 = \epsilon_0(1-p_0)(1-\gamma_0)$.)

Theorem 1. *For a regular LDPC code with variable-node degree d_v and check-node degree d_c, we have $\alpha_{t+1} = \alpha_0 C_t + \kappa_0 D_t + \beta_0 \mu_t$, where $C_t = 1-(1-A_t)^{d_v-1} + \sum_{i=0}^{\tau-1}\binom{d_v-1}{i}B_t^i(1-A_t-B_t)^{d_v-i-1}$, $D_t = \sum_{j=\tau}^{d_v-1}\binom{d_v-1}{j}A_t^j(1-A_t-B_t)^{d_v-1-j}$, $\mu_t = \sum_{m=\pi}^{d_v-1}\binom{d_v-1}{m}A_t^m(1-A_t-B_t)^{d_v-1-m}$ with $A_t = \frac{(1-\beta_t)^{d_c-1}-(1-\beta_t-2\alpha_t)^{d_c-1}}{2}$ and $B_t = \frac{(1-\beta_t)^{d_c-1}+(1-\beta_t-2\alpha_t)^{d_c-1}}{2}$. And $\beta_{t+1} = \beta_0(1-\mu_t-\nu_t)$, where $\nu_t = \sum_{m=\pi}^{d_v-1}\binom{d_v-1}{m}B_t^m(1-A_t-B_t)^{d_v-1-m}$.*

Define *erasure threshold* ϵ^* as the maximum erasure probability (for ϵ_0) for which the LDPC code can decode successfully (which means the error/erasure probabilities α_t and β_t both approach 0 as $t\to\infty$). Let us show how the NR decoder can substantially improve ϵ^*. Consider a regular LDPC code with $d_v = 5$ and $d_c = 100$ of rate 0.95. Without NR-decoding, the erasure threshold is $\tilde{\epsilon}^* = 0.036$. Now let $\pi = 1$ and $\tau = 4$. For compressed images, when $\epsilon_0 = 0.065$, the NR-decoder gives $p_0 = 0.247$ and $\gamma_0 = 0.0008$, for which the LDPC decoder has $\lim_{t\to\infty}\alpha_t = 0$ and $\lim_{t\to\infty}\beta_t = 0$. (The same happens for $\epsilon_0 < 0.065$.) So with NR-decoding, $\epsilon^* \geq 0.065$, which means the improvement in erasure threshold is more than 80.5%. The same happens for LZW-compressed texts [1]: with NR-decoding, $\epsilon^* \geq 0.3$, which means the improvement in erasure threshold is more than 733.3%.

IV. Iterative Decoding for Languages

We present a theoretical model for compressed languages, and analyze the decoding performance when we use *iterative decoding between the LDPC decoder and NR-decoder*. (In last section's study, the NR-decoder is followed by the LDPC decoder, without iterations between them.) Let $T = (b_0, b_1, b_2, \cdots)$ be a compressed text. Partition T into segments $S_0, S_1, S_2\cdots$, where each segment $S_i = (b_{il}, b_{il+1}, \cdots, b_{il+l-1})$ has l bits. Consider erasures. Let $\theta \in [0,1]$, $l_\theta \triangleq \lfloor l\theta \rfloor$ and $p \in [0,1]$ be parameters. We assume that when a segment S_i has at most l_θ erasures, the NR-decoder can decode it by checking the validity of the up to 2^{l_θ} candidate solutions (based on the validity of words/phrases, grammar, etc.), and either determines the correct solution with probability p or makes no decision with probability $1-p$. And this NR-decoding operation can be performed only *once* for each segment. Here l_θ models the limit on time complexity, and p models the probability of making a high-confidence decision. The model is suitable for compression algorithms such as Huffman coding, LZW coding, etc., where each segment can be decompressed to a sequence of characters. The greater l is, the better the model is.

The decoding model is shown in Fig. 1 (d). Let $\epsilon_0 < 1$ be the BEC's erasure rate. Let ϵ'_t and ϵ_t be the LDPC codeword's erasure rate after the t-th iteration of the LDPC decoder and the NR-decoder, respectively. We analyze the density evolution for regular (d_v, d_c) LDPC codes of rate $R = 1 - \frac{d_v}{d_c}$.

Call an l-bit segment *lucky* if the NR-decoder can decode it successfully when it has no more than l_θ erasures. For $t = 1, 2, 3\cdots$ and $k = 0, 1, \cdots, l$, let $f_k(t)$ denote the probability that a lucky segment contains k erasures after t iterations of decoding by the NR-decoder. Define $q_0 = 1$, $q_t \triangleq \frac{\epsilon_t}{\epsilon'_t}$ and $d_t \triangleq \frac{\epsilon'_t}{\epsilon_{t-1}}$ for $t \geq 1$. Note that decoding will end after t iterations if one of these conditions occurs: (1) $\epsilon'_t = 0$, because all erasures are corrected by the t-th iteration; (2) $d_t = 1$, because the LDPC decoder corrects no erasure in the t-th iteration, and nor will the NR-decoder since the input codeword is identical to its previous output. We now study density evolution before those boundary cases occur.

Theorem 2. *For $t \geq 1$, $\epsilon_t = ((1-R) + R(1-p))\epsilon_0(\prod_{i=1}^{t} d_t) + Rp\sum_{k=l_\theta+1}^{l}\frac{k}{l}f_k(t)$, and $\epsilon'_t = (\prod_{m=0}^{t-1} q_m)\epsilon_0(1-(1-\epsilon_{t-1})^{d_c-1})^{d_v-1}$. $f_k(1)$ is $\sum_{i=0}^{l_\theta}\binom{l}{i}(\epsilon'_1)^i(1-\epsilon'_1)^{l-i}$ if $k=0$, is 0 if $1\leq k\leq l_\theta$, and is $\binom{l}{k}(\epsilon'_1)^k(1-\epsilon'_1)^{l-k}$ if $l_\theta+1\leq k\leq l$. For $t\geq 2$, $f_k(t)$ is $f_k(t-1)+\sum_{i=l_\theta+1}^{l}\sum_{j=0}^{l_\theta} f_i(t-1)\binom{i}{j}(d_t)^j(1-d_t)^{i-j}$ if $k=0$, is 0 if $1\leq k\leq l_\theta$, and is $\sum_{i=k}^{l} f_i(t-1)\binom{i}{k}(d_t)^k(1-d_t)^{i-k}$ if $l_\theta+1\leq k\leq l$.*

References

[1] A. Jiang, P. Upadhyaya, E. Haratsch and J. Bruck, "Error Correction by Natural Redundancy for Long Term Storage," in *Proc. NVMW 2017*.