Approximate Locally Decodable Codes with Constant Query Complexity and Nearly Optimal Rate

Geoffrey Mon, Dana Moshkovitz, and Justin Oh
Department of Computer Science
University of Texas at Austin
Austin, TX, USA
Email: {gmon, danama, sjo}@cs.utexas.edu

***Abstract*—We present simple constructions of good approximate locally decodable codes (ALDCs) in the presence of a δ-fraction of errors for $\delta < 1/2$. In a standard locally decodable code $C\colon \Sigma_1^k \to \Sigma_2^n$, there is a decoder M that on input $i \in [k]$ correctly outputs the i-th symbol of a message x (with high probability) using only q queries to a given string w that is δ-close to $C(x)$. In an ALDC, the decoder M only needs to be correct on a $1 - \varepsilon$ fraction of $i \in [k]$ for ε much smaller than δ. We present a construction of explicit ALDCs for all constants $1/2 > \delta > \varepsilon$ with a constant number of queries q and with constant, near-optimal rate. Standard LDCs with constant number of queries and *any* constant rate are known to be impossible.**

We additionally explore what is the lowest error probability ε one can achieve for fixed δ and q. We show that for any ALDC, $\varepsilon = \Omega(\delta^{\lceil q/2 \rceil})$. We then show that there exist explicit constant rate ALDCs for any constant q that achieve $\varepsilon = O(\delta^{\lceil q/2 \rceil})$. In particular, for $q = 3$, we have a constant rate ALDC with error probability $\varepsilon = O(\delta^2)$.

A full version of this paper is available at https://eccc.weizmann.ac.il/report/2023/056/.

I. Introduction

Locally decodable codes (LDCs) are a useful and pervasive tool in both application and theory, and there has been intense study towards constructing such codes with optimal parameters. By the seminal work of Katz and Trevisan [1], asymptotically good LDCs with the "dream" parameters of constant rate, distance, and query complexity cannot exist. In this work, we show that simple constructions *can* achieve such ideal parameters for the relaxed notion of *approximate* locally decodable codes (ALDCs), which are natural codes that, like LDCs, often emerge in complexity theory.

In a standard (q, δ, ε)-LDC $C\colon \Sigma_1^k \to \Sigma_2^n$, there exists a randomized decoding algorithm M that takes as input $i \in [k]$ and has query access to a string w that is δ-close to a codeword $C(x)$. This sublinear-time decoder must correctly output $M^w(i, r) = x_i$ with probability at least $1 - \varepsilon$ over its internal randomness r for any $i \in [k]$, using at most q (non-adaptive) queries to w. In contrast, an approximate locally decodable code only requires that M successfully decodes *most* of the message coordinates with few queries. That is, for a (q, δ, ε)-approximate locally decodable code $C\colon \Sigma_1^k \to \Sigma_2^n$ there again exists a randomized algorithm M that takes as input $i \in [k]$ and makes at most q queries to a string w that is δ-close to a codeword $C(x)$. This time, the decoder must correctly output $M^w(i, r) = x_i$ with probability $1 - \varepsilon$ over its internal randomness r *on average* over all coordinates $i \in [k]$ of the message.[1] The identity code is a trivial $(1, \delta, \varepsilon)$-ALDCs for $\delta = \varepsilon$, so $\varepsilon \ll \delta$ is the only parameter regime where ALDCs make sense.

To make sense of this definition, suppose we would like to amplify the decoding radius of a code $C\colon \Sigma_0^k \to \Sigma_1^n$ from ε to δ. It is natural to seek a map $C'\colon \Sigma_1^n \to \Sigma_2^m$ equipped with a decoder which can, given any input which is δ-close to some $C'(x)$, return a string x' which is ε-close to x. Then, $C' \circ C$ is a new code with a decoding radius of δ: run the decoder of C' on any input which is δ-close to $C' \circ C$ to get x' which is ε-close to some codeword of C, and then run the decoder of C on x' to uniquely recover the message. If the decoder of C' features locality, then C' is precisely an ALDC which can be used to amplify the decoding radius of an LDC [3].

Historically, the notion of ALDCs is motivated by topics in computational complexity theory such as hardness amplification. Informally, suppose there is a function f with a truth table that differs from the truth table of every possible efficiently computable function on at least an ε fraction of inputs—that is, no efficient algorithm can compute f with accuracy better than $1 - \varepsilon$. Then, we can encode the truth table of f with an ALDC to get the truth table of a new, *harder*, function which cannot be computed by any efficient algorithm with accuracy better than $1 - \delta < 1 - \varepsilon$: if there existed such an algorithm, then we could combine it with an algorithm that computes the ALDC decoder to build an algorithm computing f with accuracy better than $1 - \varepsilon$, which is a contradiction. Explicit ALDCs considered in this context are the *XOR code* and the *direct product code* as well as their derandomized counterparts [4]–[6].

Versions of ALDCs also appear in the context of probalistically checkable proofs, which are proofs that can be verified

[1]The decoder does not necessarily know which indices i are decoded correctly. A different relaxed definition, known as *relaxed locally decodable codes* [2], requires the decoder to report which indices are corrupted.

with very few queries and randomness. In this research area, certain constructions allow for symbols of the witness to be decoded from the proof using few queries, a concept called a *PCP of proximity* [2] or *decoding PCP* [7], [8]. Such constructions typically only guarantee that most (instead of all) symbols are decoded correctly.

II. Our Results

We give both constructions and lower bounds, demonstrating the possibility and limits of ALDCs in terms of rate and locality. The first constructive result gives constant query ALDCs with near optimal rate and arbitrary error reduction.

A. Constant Query ALDCs with Nearly Optimal Rate

In the case of unique decoding, the Singleton bound tells us that a code that can be decoded from δ fraction errors must have rate at most $1 - 2\delta + o(1)$, because a decoding radius of δ implies a distance of at least 2δ. Notably, the Reed–Solomon code is a uniquely (non-local) decodable code that achieves this bound. Kopparty, Meir, Ron-Zewi, and Saraf [9] use Reed–Solomon as the base code for Alon–Edmonds–Luby distance amplification [10], [11] to construct ALDCs with rate approaching the Singleton bound; this construction is crucial to their construction of asymptotically-good LDCs with subpolynomial query complexity:

Theorem 2.1 (due to [9, Lemma 3.2]): For any $\delta, \varepsilon > 0$ and any parameter $\alpha > 0$, there is an explicit (q, δ, ε)-ALDC with rate $1 - 2\delta - 2\alpha$ and query complexity $q = \mathrm{poly}(1/\varepsilon\alpha)$.

This theorem shows a quite striking contrast. One cannot hope to have an LDC with both constant rate and constant number of queries. Nevertheless, it is possible to have an ALDC that reduces any sufficiently small constant error rate δ, to any arbitrarily smaller constant error ε, with constant query complexity and with arbitrarily high rate.

Then, a natural question to ask is: what is the optimal rate for a (q, δ, ε)-ALDC? In particular, the Singleton rate upper bound of $1 - 2\delta + o(1)$ does *not* apply to (q, δ, ε)-ALDCs, because they can have distance less than 2δ: an ALDC's decoder is only required to return most of the message correctly, so two messages can map to the same ALDC codeword as long as these messages are ε-close to each other. We take advantage of this fact to get an ALDC construction with higher rate slightly exceeding the Singleton bound:

Theorem 2.2 (See Corollary 3.6): For any constants $\delta, \varepsilon > 0$, any sufficiently small constant parameter α, any constant sized finite field Σ_1, and any sufficiently large k, there is an explicit $C \colon \Sigma_1^k \to \Sigma_2^n$ which is a (q, δ, ε)-ALDC with $q = \mathrm{poly}(1/\varepsilon\alpha)$ with $|\Sigma_2| \leq |\Sigma_1|^{q^{2^{O(q)}}}$. The rate of the code is

$$\frac{1 - 2\delta - 2\alpha}{1 - .99H_{|\Sigma_1|}(\varepsilon/2)} - o(1).$$

To achieve this, we refine the Alon–Edmonds–Luby technique [10], [11] as used by Kopparty, Meir, Ron-Zewi, and Saraf [9]. This technique builds a code that divides the message into small constant-size blocks, encodes each block with a base code, and then permutes all of the symbols according to a *sampler* graph to form the codeword. The sampler graph guarantees that no matter which δ fraction of codeword symbols are corrupted, at least a $1-\varepsilon$ fraction of the blocks will "see" an approximately δ fraction of corruption. Because the sampler graph permutes symbols but does not duplicate them, the rate of this code is the same as the rate of the base code. In addition, each message symbol can be decoded by querying all of the (constant many) symbols in its corresponding block. If Reed–Solomon is used as the base code, then this yields an ALDC with constant query complexity and rate approaching the Singleton bound [9]. The $1-\varepsilon$ fraction of blocks have bounded distance from the Reed–Solomon base code and can be uniquely decoded, while the remaining ε fraction of blocks which have too much corruption are written off.

We improve the rate by using a (δ, ε)-approximate code[2] as the base code for the AEL construction instead of Reed–Solomon. This base code is weaker than a uniquely decodable code such as Reed–Solomon, and hence can have higher rate. Then, in each of the blocks with bounded corruption, a $1 - \varepsilon$ fraction of message symbols can be recovered. Since $1 - \varepsilon$ of the blocks have at most $O(\delta)$ corruption, a $1 - O(\varepsilon)$ fraction of the entire message can be decoded. Hence, we can spread the message corruption across all of the blocks, instead of concentrating it in the ε fraction of blocks written off by the sampler, in order to relax the base code and get a higher rate.

The final step is to show that an approximate code with higher rate exists. This code will be used on constant-sized blocks, so we can afford to construct it by brute force. Because this code only needs to preserve $1-\varepsilon$ of the message coordinates, we can pick a subset D (known as a covering code) of the message space Σ^k such that every message is ε-close to some string in D. Then, we can view D as a new, smaller set of messages $\Sigma^{k'}$ and encode it using Reed–Solomon with distance 2δ. To approximately decode, we can use the Reed–Solomon decoder to fully recover an element of D which is guaranteed to be ε-close to our true original codeword. The rate of this code is $k/k' \cdot (1 - 2\delta)$, which "exceeds" the Singleton bound.

B. Rate Upper Bound

Theorem 2.2 demonstrates that an ALDC can have rate slightly exceeding a uniquely decodable code of the same decoding radius. We show that this rate is in fact nearly optimal for approximate codes in general, even without locality. Simply observe that composing an approximate code with a uniquely decodable code yields a uniquely decodable code, which is then governed by traditional coding theory rate bounds. Hence, adding locality incurs almost no cost on rate.

Theorem 2.3 (See Theorem 3.3): A (δ, ε)-approximate code $C \colon \Sigma_1^k \to \Sigma_2^n$ must have rate

$$R \leq \frac{1 - 2\delta + o(1)}{1 - H_{|\Sigma_1|}(2\varepsilon) - o(1)}.$$

[2]A (δ, ε)-*approximate code* is an ALDC with no requirement of locality: there is some (potentially global) decoder that reduces error from δ to ε.

C. 3-Query ALDCs and Optimal Error Reduction

In the most extreme setting of local decoding, one may ask what is possible with only 3 queries. Indeed, this has been the subject of extended study in the case of LDCs with both upper bounds [12]–[14] and lower bounds [15]–[19], and as discussed, it is impossible to achieve constant rate in that case.

Evidently, relaxing the goal of the decoder to error reduction rather than "error elimination" drastically improves the state of what is feasible in the constant rate regime. Thus it is natural to ask, what is the best error reduction possible given q queries for a constant rate ALDC? We show that for 3 queries, one cannot hope for better than a $(3, \delta, \Theta(\delta^2))$-ALDC for *any* rate. Intuitively, a randomized decoder makes three uniformly random queries to the coordinates of a codeword with a δ-fraction of corruptions. When at least two of these queries read a corrupted symbol, there is no hope the decoder will output the correct message symbol. In fact, we show that any q-query decoder cannot succeed (with probability over both its randomness and a uniform choice of message coordinate) with probability better than the probability that the majority of its queries land on uncorrupted symbols.

Theorem 2.4 (See Theorem 4.1): Let $C\colon \Sigma_1^k \to \Sigma_2^n$ be a (q, δ, ε)-ALDC with query complexity $q = O(1)$ and decoding radius $\delta < 1/2$. Then $\varepsilon = \Omega(\delta^{\lceil q/2 \rceil})$.

In addition, we show that it is indeed possible to construct a constant rate 3-query ALDC with this optimal error reduction. In fact we show that it is possible to obtain optimal error reduction for any given constant number of queries, by adapting a different but related distance amplification technique due to Alon–Bruck–Naor–Naor–Roth [20].

Theorem 2.5 (See Corollary 4.3): Let $q > 1$ and $0 < \delta < 1/2$ be constants. Let Σ_1 be any alphabet. There are explicit (q, δ, ε)-ALDCs $C\colon \Sigma_1^k \to \Sigma_2^n$ with constant rate, and $\Sigma_2 = \Sigma_1^{D_R}$, where $\varepsilon = O(\delta^{\lceil q/2 \rceil})$, $D_L = O(\frac{q^2}{\delta^2} \log \frac{q}{\delta^{\lceil q/2 \rceil}})$, and $D_R = 2^{D_L 2^{O(D_L)}}$.
This construction is similar to the near-optimal rate ALDC construction. We can use a sampler graph with the repetition code for each message symbol, and decode by taking the majority over q uniformly random copies of the desired message symbol.

III. ALDCs with Nearly Optimal Rate

In this section, we will first present an approximate code with high rate. We will then prove that such a rate is (nearly) optimal. Finally we'll show how to use this approximate code to construct an explicit ALDC with efficient encoding and decoding procedure with roughly the same rate.

A. An Approximate Code "Surpassing" the Singleton Bound

Since the decoding radius δ is the relevant parameter for an approximate code, one might believe that, analogously to the Singleton bound, the rate of an approximate code is at most roughly $1 - 2\delta$. We first present an approximate code showing that we can in fact do slightly better. To achieve slightly better rate, we will make use of nearly-optimal covering codes.

Proposition 3.1 (due to [21, Corollary 1.4]): For all $\varepsilon \geq 3/k$ and all constant size alphabets Σ, there (non-explicitly) exists a covering code $D \subseteq \Sigma^k$ of radius ε such that

$$|D| \leq O(\varepsilon k \log(\varepsilon k)) \cdot \frac{|\Sigma|^k}{\mathrm{Vol}_{|\Sigma|}(\varepsilon, k)} = |\Sigma|^{(1 - H_{|\Sigma|}(\varepsilon) + o(1))k}.$$

We show that approximate codes exist for any constants $\delta > \varepsilon > 0$ with nearly-optimal rate by using a covering code to compress the message space, and then applying Reed-Solomon.

Lemma 3.2: For any constants $1/2 > \delta > \varepsilon > 0$, any constant size field $\mathbb{F}$, and any $k \geq 3/\varepsilon$, there exists an approximate code $C\colon \mathbb{F}^k \to (\mathbb{F}')^n$ with decoding radius δ, error ε, and rate $\frac{1-2\delta}{1 - H_{|\mathbb{F}|}(\varepsilon) + o(1)}$, where $\mathbb{F}'$ is an extension field of $\mathbb{F}$ and $|\mathbb{F}'| \leq |\mathbb{F}| n$.

Proof sketch: Let $D \subseteq \mathbb{F}^k$ be a covering code of radius ε from Proposition 3.1, so that

$$|D| = |\mathbb{F}|^{(1 - H_{|\mathbb{F}|}(\varepsilon) + o(1))k} =: |\mathbb{F}|^{k'}.$$

We can (inefficiently) obtain D by brute force. Let $f\colon \mathbb{F}^k \to D$ be a function such that for all $w \in \mathbb{F}^k$, $f(w)$ is some element of D which is ε-close to w (if there are multiple such elements in the covering code, pick one arbitrarily and deterministically). Let $g\colon D \to \mathbb{F}^{k'}$ be an arbitrary bijection. Both f and g can be obtained by brute force. For any given message $w \in \mathbb{F}^k$, we can then let $x = g(f(w)) \in \mathbb{F}^{k'}$, and then encode this string using Reed–Solomon with distance 2δ. Since $k'/k = 1 - H_{|\mathbb{F}|}(\varepsilon) + o(1)$ and the rate of Reed–Solomon is at least $1 - 2\delta$, the overall rate is

$$\frac{k}{k'} \cdot (1 - 2\delta) = \frac{1 - 2\delta}{1 - H_{|\mathbb{F}|}(\varepsilon) + o(1)},$$

as desired. ■

B. An "Approximate" Singleton Bound

We now show that the rate of the code above is in fact essentially optimal. To do so, we prove an approximate analogue of the Singleton bound.

Theorem 3.3: Suppose $C\colon \Sigma_1^k \to \Sigma_2^n$ is a (δ, ε)-approximate code with rate R. Then

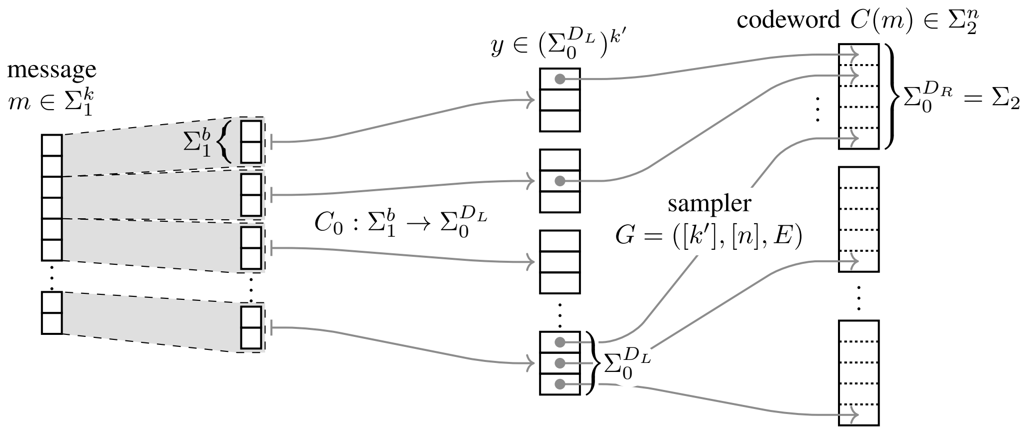
$$R \leq \frac{1 - 2\delta + o(1)}{1 - H_{|\Sigma_1|}(2\varepsilon) - o(1)}.$$

Proof: Let $C'\colon \{0,1\}^{k'} \to \Sigma_1^k$ be a standard code of radius ε and rate $R' = 1 - H_{|\Sigma_1|}(2\varepsilon) - o(1)$ guaranteed by the Gilbert–Varshamov bound.

The composition $C \circ C'$ is a code with radius δ. The rate of this code is RR'. Moreover, the rate of this code must obey the Singleton bound. Thus we have

$$(1 - H_{|\Sigma_1|}(2\varepsilon) - o(1))R \leq RR' \leq 1 - 2\delta + o(1). \quad ■$$

Note that this same approach can be applied to any classical coding theory rate upper bound, and we can pick whichever is strongest for the desired alphabet size.



codeword $C(m) \in \Sigma_2^n$
$y \in (\Sigma_0^{D_L})^{k'}$
message
$m \in \Sigma_1^k$
Σ_1^b
$\Sigma_0^{D_R} = \Sigma_2$
sampler
$G = ([k'], [n], E)$
$C_0 : \Sigma_1^b \to \Sigma_0^{D_L}$
$\Sigma_0^{D_L}$

Fig. 1. The construction of our ALDC with nearly optimal rate. We first divide the message into blocks of constant length b and encode each block with a near-optimal rate approximate code C_0 found via brute force. On the right hand side, we have a biregular (δ, ε)-sampler between $k' = k/b$ nodes and n nodes with left degree D_L and right degree D_R. Each coordinate of the final codeword is the tuple of neighboring symbols. The code inherits the rate of C_0.

C. An ALDC Approaching the "Approximate" Singleton Bound

Now that we see that our approximate code has nearly optimal rate, we turn to the task of making such codes local. To do so, we closely follow the techniques of [10] and [9]. We will sketch the ideas here, which are also depicted in Figure 1, and leave the technical details for the full version.

To build our ALDC, we will instantiate the Alon–Edmonds–Luby distance amplification technique with our nearly-optimal approximate code. As we described in the introduction, this technique is built on using samplers.

Definition 3.4: A bipartite graph G with bipartition $([k], [n])$ and with left degree D is an *oblivious (α, β)-sampler* if for every function $f\colon [n] \to [0, 1]$,

$$\Pr_i \left[\left| \mathop{\mathbb{E}}_{j \in N(i)}[f(j)] - \mathop{\mathbb{E}}_{j' \in [n]}[f(j')] \right| \le \alpha \right] \ge 1 - \beta,$$

where $N(i)$ denotes the right vertices neighboring the ith left vertex in G.

The value of an oblivious sampler is it provides a blueprint for redistributing message symbols such that if any δ fraction of codeword symbols are corrupted (represented by an indicator function f), then most of the message symbols will still have roughly δ fraction of their relevant codeword symbols affected. Indeed, we use a well-known oblivious sampler based on random walks on expanders in both of our ALDC constructions.

Theorem 3.5 (due to [22]): For any $\alpha, \beta > 0$ and for sufficiently large k, there exists an efficiently constructable biregular (α, β)-sampler with k left vertices, left degree $D = O(\frac{1}{\alpha^2} \log \frac{1}{\beta})$, and $n = k/2^{O(D)}$ right vertices.

Hence, if we wish to construct a (q, δ, ε)-ALDC $C\colon \Sigma_1^k \to \Sigma_2^n$ with good rate and small q for some Σ_2 and n, we can begin by choosing any $0 < \alpha < \min\{1/4 - \delta/2, \varepsilon\}$ be a parameter. There is an explicit biregular $(\alpha, \varepsilon/2)$-sampler with left degree $D_L = \Theta(\frac{1}{\alpha^2} \log \frac{1}{\varepsilon})$, so we can choose some integer b such that there exists a $(\delta + \alpha, \varepsilon/2)$-approximate code $C_0\colon \Sigma_1^b \to \Sigma_0^{D_L}$ from Lemma 3.2 with rate at least $\frac{1-2\delta-2\alpha}{1-H_{|\Sigma_1|}(\varepsilon/2)}$. Since D_L is a constant, we can afford to brute force the covering code necessary to construct C_0.

a) Encoding: To encode a length k input message x, we divide the message into $k' = k/b$ blocks of length b, and encode each block using C_0. This gives us a string $y \in ((\Sigma_0)^{D_L})^{k'}$. We treat each index of y as a left vertex of G. The final codeword $C(x) \in \Sigma_2^n$ for $\Sigma_2 = (\Sigma_0)^{D_R}$ is defined as follows. Fix any $i \in [n]$. We now define $C(x)_i$. For every neighbor $j \in [k']$ of i, let $e(j) \in [D_L]$ be the number such that the edge entering j from i is the $e(j)$-th edge leaving i in some arbitrary ordering of the edges leaving j. Finally let $\sigma_j \in \Sigma_0$ be the $e(j)$-th symbol in the block corresponding to neighbor j. The i-th symbol of $C(x)$ is the concatenation of all such σ_j-s.

b) Decoding and analysis: The rate of this code is clearly the same as the rate of C_0, because the sampler step permutes symbols but does not duplicate any symbols. In addition, the query complexity for decoding is D_L because we can approximately decode an entire b-sized block of message symbols by querying each of the neighboring D_L codeword symbols and running the decoder for C_0. This procedure is deterministic, and it satisfies the ALDC definition because by the sampler property, all but at most $\varepsilon/2$-fraction of the k' message blocks have a neighborhood with at most $(\delta + \alpha)$-fraction of neighbors in the corrupted set. This means that on at least $(1 - \varepsilon/2)$-fraction of blocks, the decoding algorithm for C_0 will return a string $s \in \{0, 1\}^b$ that is $\varepsilon/2$-close to the true message on that block. Thus the decoder will only err on at most ε-fraction of the message coordinates overall.

Finally, it remains to express D_L in terms of our parameters. The parameters we compute here incorporate ancillary factors due to technicalities with rounding, padding, etc. which can be found in the full version.

Corollary 3.6: For any constants $\delta, \varepsilon > 0$, any constant parameter $0 < \alpha < \min\{1/4 - \delta/2, \varepsilon\}$, any constant sized alphabet Σ_1 which is a finite field, and any sufficiently large

k, there exists an explicit $C\colon \Sigma_1^k \to \Sigma_2^n$ is a (q,δ,ε)-ALDC with $q = O\left(\frac{1}{\alpha^3}\log\frac{1}{\varepsilon}\right)$ with $|\Sigma_2| \le |\Sigma_1|^{q2^{O(q)}}$. The rate of the code is

$$\frac{1-2\delta-2\alpha}{1-.99H_{|\Sigma_1|}(\varepsilon/2)} - o(1).$$

IV. Achieving Optimal Error Reduction

What is the best possible error reduction that a q-query ALDC can achieve, and what structure would such a decoder have? In fact, we are able to show that the best possible error reduction, even for inefficient ALDCs, is $\varepsilon = \Theta(\delta^{\lceil q/2\rceil})$ which is achieved by taking the majority of q queries. We construct an efficient q-query ALDC with constant rate and alphabet that achieves this optimal error reduction up to constant factors.

A. Majority Lower Bound on Error

Let C be some nonadaptive (q,δ,ε)-ALDC, over any size alphabet and with any rate. Then, we can show that $\varepsilon \ge \Omega(\delta^{\lceil q/2\rceil})$; in particular, for two queries, this shows that C must have error $\ge \delta$ which is already achieved by the 1-query identity code.

Theorem 4.1: Let $C\colon \Sigma_1^k \to \Sigma_2^n$ be a nonadaptive (q,δ,ε)-ALDC with $q = O(1)$, $0 < \delta < 1/2$, and sufficiently large n. Then, $\varepsilon \ge \Omega(\delta^{\lceil q/2\rceil})$.

Proof sketch: At a high level, we can use the probabilistic method to craft a corrupted codeword on which an ALDC decoder must fail with probability at least $\Omega(\delta^{\lceil q/2\rceil})$. Begin with two messages m and m' which differ from each other in every coordinate. Then, choose a uniformly random subset S and craft two corrupted codewords w and w', such that $w|_{\bar{S}} = C(m)|_{\bar{S}}$ and $w|_S = C(m')|_S$, and vice versa for w'. Thus, if the decoder queries w inside S, it sees symbols that look like $C(m')$, and if it queries w outside S, it sees symbols that look like $C(m)$. Through some computation, we can prove that the decoder must fail on either w or w' with at least the probability that half of its queries land in S, which is $\Omega(\delta^{\lceil q/2\rceil})$. Intuitively, if the decoder succeeds on message m even if most of its queries are corrupted, then it is biased towards returning the content of m, causing it to perform worse on m'. ■

The exact form of this bound (in the full version) proves that there is no 2-query ALDC that outperforms the identity code (where $C(x) = x$), which is a $(1,\delta,\varepsilon)$-ALDC for $\varepsilon = \delta$.

Corollary 4.2: A nonadaptive ALDC with $q \le 2$ queries has error $\ge \delta$.

B. Achieving the Majority Lower Bound

Although the majority lower bound applies to all (q,δ,ε)-ALDCs, including ones with inefficient decoders, huge codeword lengths, or huge alphabet size, we can show that the bound can be achieved up to a constant factor by explicit and efficient $(q,\delta,O(\delta^{\lceil q/2\rceil}))$-ALDCs with constant rate and alphabet size. To do so, we apply the Alon–Bruck–Naor–Naor–Roth technique [20].

a) Encoding: Let $0 < \gamma < 1$ be an arbitrarily small constant, q be a positive constant integer, and $0 < \delta < 1/2$. To construct an ALDC $C\colon \Sigma_1^k \to \Sigma_2^n$, begin with a biregular graph G corresponding to an oblivious $(\alpha = \gamma\delta, \beta = \gamma\delta^{\lceil q/2\rceil})$-sampler with bipartition $([k],[n])$, left degree D_L, and right degree D_R. Let $m \in \Sigma_1^k$ be an arbitrary message. The jth codeword symbol $C(m)_j$ is defined as the string $(m_i)_{i\in N(j)} \in \Sigma_1^{D_R}$ where $N(j)$ is the set of left vertices neighboring the jth right vertex.

Immediately, we can see that the rate of this code will be $1/D_L$ because each message symbol is duplicated D_L times in the codeword, and the alphabet is $\Sigma_2 = \Sigma_1^{D_R}$.

b) Decoding and analysis: For a message index i and input string w, consider the decoder which independently repeats the following step q times, and takes the majority of the results:

1) Pick a uniformly random right vertex j neighboring left vertex i in the sampler graph G.
2) Query the jth index of w, and then return the symbol of w_j which corresponds to message index i.

If w is δ-close to some codeword $C(m)$, then by the sampler property, a $1-\beta = 1-\gamma\delta^{\lceil q/2\rceil}$ fraction of message coordinates will be "good": these coordinates i have at most a $\le \delta+\alpha = (1+\gamma)\delta$ fraction of their neighbors which are corrupted. If a message index i is good, then the probability of a uniformly random neighbor being corrupt is $\le (1+\gamma)\delta$, and so the probability that the majority of q repetitions is corrupt is

$$\le O(((1+\gamma)\delta)^{\lceil q/2\rceil}).$$

Hence the decoder achieves the following error:

$$\le \beta + (1-\beta)\cdot O(((1+\gamma)\delta)^{\lceil q/2\rceil}) \le O(((1+\gamma)\delta)^{\lceil q/2\rceil}).$$

If $\gamma = 1/100q = O(1)$, we can apply the sampler from Theorem 3.5 to complete the construction.

Corollary 4.3: For any $0 < \delta < 1/2$ and constant positive integer q, there exists an explicit code $C\colon \Sigma_1^k \to \Sigma_2^n$ which is a $(q,\delta,O(\delta^{\lceil q/2\rceil}))$-ALDC with rate $1/D_L$ and alphabet $\Sigma_2 = \Sigma_1^{D_R}$, where $D_L = O(\frac{q^2}{\delta^2}\log\frac{q}{\delta^{\lceil q/2\rceil}})$ and $D_R = 2^{D_L 2^{O(D_L)}}$.

Acknowledgements

GM is supported by NSF Grant CCF-2200956, an NSF Graduate Research Fellowship (DGE-2137420), and a UT Austin Dean's Prestigious Fellowship Supplement. DM is supported by NSF Grants CCF-1705028, CCF-2200956, and CCF-2312573. JO is supported by NSF Grants CF-2008076 and CCF-2312573. We are grateful to anonymous reviewers for their input.

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-2137420. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] J. Katz and L. Trevisan, “On the efficiency of local decoding procedures for error-correcting codes,” in *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, ser. STOC ’00. New York, NY, USA: Association for Computing Machinery, May 2000, pp. 80–86. [Online]. Available: https://doi.org/10.1145/335305.335315

[2] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan, “Robust PCPs of proximity, shorter PCPs, and applications to coding,” *SIAM Journal on Computing*, vol. 36, no. 4, pp. 889–974, Jan. 2006, publisher: Society for Industrial and Applied Mathematics. [Online]. Available: https://doi.org/10.1137/S0097539705446810

[3] A. Ben-Aroya, K. Efremenko, and A. Ta-Shma, “A note on amplifying the error-tolerance of locally decodable codes,” Electronic Colloquium on Computational Complexity (ECCC), Tech. Rep. TR10-134, Dec. 2010. [Online]. Available: https://eccc.weizmann.ac.il/report/2010/134/

[4] R. Impagliazzo and A. Wigderson, “$P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma,” in *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, F. T. Leighton and P. W. Shor, Eds. ACM, 1997, pp. 220–229. [Online]. Available: https://doi.org/10.1145/258533.258590

[5] L. Trevisan, “List-decoding using the XOR lemma,” in *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*. IEEE Computer Society, 2003, pp. 126–135. [Online]. Available: https://doi.org/10.1109/SFCS.2003.1238187

[6] R. Impagliazzo, R. Jaiswal, V. Kabanets, and A. Wigderson, “Uniform direct product theorems: Simplified, optimized, and derandomized,” *SIAM J. Comput.*, vol. 39, no. 4, pp. 1637–1665, 2010. [Online]. Available: https://doi.org/10.1137/080734030

[7] D. Moshkovitz and R. Raz, “Two-query PCP with subconstant error,” *J. ACM*, vol. 57, no. 5, pp. 29:1–29:29, 2010. [Online]. Available: https://doi.org/10.1145/1754399.1754402

[8] I. Dinur and P. Harsha, “Composition of low-error 2-query PCPs using decodable PCPs,” *SIAM J. Comput.*, vol. 42, no. 6, pp. 2452–2486, 2013. [Online]. Available: https://doi.org/10.1137/100788161

[9] S. Kopparty, O. Meir, N. Ron-Zewi, and S. Saraf, “High-rate locally correctable and locally testable codes with sub-polynomial query complexity,” *J. ACM*, vol. 64, no. 2, pp. 11:1–11:42, 2017. [Online]. Available: https://doi.org/10.1145/3051093

[10] N. Alon, J. Edmonds, and M. Luby, “Linear time erasure codes with nearly optimal recovery,” in *Proceedings of IEEE 36th Annual Foundations of Computer Science*, Oct. 1995, pp. 512–519, iSSN: 0272-5428. [Online]. Available: https://doi.org/10.1109/SFCS.1995.492581

[11] N. Alon and M. Luby, “A linear time erasure-resilient code with nearly optimal recovery,” *IEEE Trans. Inf. Theory*, vol. 42, no. 6, pp. 1732–1736, 1996. [Online]. Available: https://doi.org/10.1109/18.556669

[12] S. Yekhanin, “Towards 3-query locally decodable codes of subexponential length,” *J. ACM*, vol. 55, no. 1, pp. 1:1–1:16, 2008. [Online]. Available: https://doi.org/10.1145/1326554.1326555

[13] K. Efremenko, “3-query locally decodable codes of subexponential length,” *SIAM J. Comput.*, vol. 41, no. 6, pp. 1694–1703, 2012. [Online]. Available: https://doi.org/10.1137/090772721

[14] Z. Dvir, P. Gopalan, and S. Yekhanin, “Matching vector codes,” *SIAM J. Comput.*, vol. 40, no. 4, pp. 1154–1178, 2011. [Online]. Available: https://doi.org/10.1137/100804322

[15] D. P. Woodruff, “New lower bounds for general locally decodable codes,” Electronic Colloquium on Computational Complexity (ECCC), Tech. Rep. TR07-006, Jan. 2007. [Online]. Available: https://eccc.weizmann.ac.il/report/2007/006/

[16] O. Alrabiah, V. Guruswami, P. K. Kothari, and P. Manohar, “A near-cubic lower bound for 3-query locally decodable codes from semirandom CSP refutation,” in *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, ser. STOC 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 1438–1448. [Online]. Available: https://doi.org/10.1145/3564246.3585143

[17] P. K. Kothari and P. Manohar, “An exponential lower bound for linear 3-query locally correctable codes,” Electronic Colloquium on Computational Complexity (ECCC), Tech. Rep. TR23-162, Nov. 2023. [Online]. Available: https://eccc.weizmann.ac.il/report/2023/162/

[18] T. Yankovitz, “A stronger bound for linear 3-LCC,” Electronic Colloquium on Computational Complexity (ECCC), Tech. Rep. TR24-036, Apr. 2024. [Online]. Available: https://eccc.weizmann.ac.il/report/2024/036/

[19] O. Alrabiah and V. Guruswami, “Near-tight bounds for 3-query locally correctable binary linear codes via rainbow cycles,” Electronic Colloquium on Computational Complexity (ECCC), Tech. Rep. TR24-062, Apr. 2024. [Online]. Available: https://eccc.weizmann.ac.il/report/2024/062/

[20] N. Alon, J. Bruck, J. Naor, M. Naor, and R. M. Roth, “Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs,” *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 509–516, Mar. 1992. [Online]. Available: https://doi.org/10.1109/18.119713

[21] M. Krivelevich, B. Sudakov, and V. H. Vu, “Covering codes with improved density,” *IEEE Transactions on Information Theory*, vol. 49, no. 7, pp. 1812–1815, Jul. 2003. [Online]. Available: https://doi.org/10.1109/TIT.2003.813490

[22] D. Gillman, “A Chernoff bound for random walks on expander graphs,” *SIAM Journal on Computing*, vol. 27, no. 4, pp. 1203–1220, Jan. 1998, publisher: Society for Industrial and Applied Mathematics. [Online]. Available: https://doi.org/10.1137/S0097539794268765