Texts in Computational Complexity: On Error Correcting Codes

Oded Goldreich
Department of Computer Science and Applied Mathematics
Weizmann Institute of Science, Rehovot, Israel.

January 5, 2006

In this section we highlight some issues and aspects of coding theory that are most relevant to the current book. The interested reader is referred to [20] for a more comprehensive treatment of the computational aspects of coding theory. Structural aspects of coding theory, which are at the traditional focus of that field, are covered in standard textbook such as [18].

1 Getting started

Loosely speaking, an error correcting code is a mapping of strings to longer strings such that any two different strings are mapped to a corresponding pair of strings that are far apart (and not merely different). Specifically, $C : \{0,1\}^k \to \{0,1\}^n$ is a (binary) code of distance d if for every $x \neq y \in \{0,1\}^k$ it holds that $C(x)$ and $C(y)$ differ on at least d bit positions.

It will be useful to extend this definition to sequences over an arbitrary alphabet Σ, and to use some notations. Specifically, for $x \in \Sigma^m$, we denote the i^{th} symbol of x by x_i (i.e., $x = x_1 \cdots x_m$), and consider codes over Σ (i.e., mappings of Σ-sequences to Σ-sequences). The mapping (code) $C : \Sigma^k \to \Sigma^n$ has distance d if for every $x \neq y \in \Sigma^k$ it holds that $|\{i : C(x)_i \neq C(y)_i\}| \geq d$. The members of $\{C(x) : x \in \Sigma^k\}$ are called codewords (and in some texts this set itself is called a code).

In general, we define a metric, called Hamming distance, over the set of n-long sequences over Σ. The Hamming distance between y and z, where $y, z \in \Sigma^n$, is defined as the number of locations on which they disagree (i.e., $|\{i : y_i \neq z_i\}|$). The Hamming weight of such sequences is defined as the number of non-zero elements (assuming that one element of Σ is viewed as zero). Typically, Σ is associated with an additive group, and in this case the distance between y and z equals the Hamming weight of $w = y - z$, where $w_i = y_i - z_i$ (for every i).

Asymptotics. We will actually consider infinite families of codes; that is, $\{C_k : \Sigma_k^k \to \Sigma_k^{n(k)}\}_{k \in S}$, where $S \subseteq \mathbb{N}$ (and typically $S = \mathbb{N}$). (N.B., we allow Σ_k to depend on k.) We say that such a family has distance $d : \mathbb{N} \to \mathbb{N}$ if for every $k \in S$ it holds that C_k has distance $d(k)$. Needless to say, both $n = n(k)$ (called the block-length) and $d(k)$ depend on k, and the aim is to have a linear dependence (i.e., $n(k) = O(k)$ and $d(k) = \Omega(n(k))$). In such a case, one talks of the relative rate of the code (i.e., the constant $k/n(k)$) and its relative distance (i.e., the constant $d(k)/n(k)$).

In general, we will often refer to relative distances between sequences. For example, for $y, z \in \Sigma^n$, we say that y and z are ε-close (resp., ε-far) if $|\{i : y_i \neq z_i\}| \leq \varepsilon \cdot n$ (resp., $|\{i : y_i \neq z_i\}| \geq \varepsilon \cdot n$).

Computational problems. The most basic computational tasks associated with codes are encoding and decoding (under noise). The definition of the encoding task is straightforward (i.e., map $x \in \Sigma_k^k$ to $C_k(x)$), and an efficient algorithm is required to compute each symbol in $C_k(x)$ in poly$(k, |\Sigma_k|)$-time.[1] When defining the decoding task we note that "minimum distance decoding" (i.e., given $w \in \Sigma_k^{n(k)}$, find x such that $C_k(x)$ is closest to y (in Hamming distance)) is just one natural possibility. Two related variants, regarding a code of distance d, are:

Unique decoding: Given $w \in \Sigma_k^{n(k)}$ that is at Hamming distance less than $d(k)/2$ from some codeword $C_k(x)$, retrieve the corresponding decoding of $C_k(x)$ (i.e., retrieve x).

Needless to say, this task is well-defined because there cannot be two different codewords that are each at Hamming distance less than $d(k)/2$ from w.

List decoding: Given $w \in \Sigma_k^{n(k)}$ and a parameter $d' \geq d(k)/2$, output a list of all $x \in \Sigma_k^k$ that are at Hamming distance at most d' from w.

Typically, one considers the case that $d' < d(k)$. See Section 4 for discussion of upper-bounds on the number of codewords that are within a certain distance from a generic sequence.

Two additional computational tasks are considered in Section 3.

Linear codes. Associating Σ_k with some finite field, we call a code $C_k : \Sigma_k^k \to \Sigma_k^{n(k)}$ linear if it satisfies $C_k(x + y) = C_k(x) + C_k(y)$, where x and y (resp., $C_k(x)$ and $C_k(y)$) are viewed as k-dimensional (resp., $n(k)$-dimensional) vectors over Σ_k, and the arithmetic is of the corresponding vector space. A useful property of linear codes is that their distance equals the Hamming weight of the lightest codeword other than $C_k(0^k)$; that is, $\min_{x \neq y}\{|\{i : C_k(x)_i \neq C_k(y)_i\}|\}$ equals $\min_{x \neq 0^k}\{|\{i : C_k(x)_i \neq 0\}|\}$. Another useful property is that the code is fully specified by a k-by-$n(k)$ matrix, called the generating matrix, that consists of the codewords of some fixed basis of Σ_k^k. That is, the set of all codewords is obtained as the $|\Sigma_k|^k$ different linear combination of the rows of the generating matrix.

2 A few popular codes

Our focus will on explicitly constructible codes; that is, (families of) codes of the form $\{C_k : \Sigma_k^k \to \Sigma_k^{n(k)}\}_{k \in S}$ that are coupled with efficient encoding and decoding algorithms. But before presenting a few such codes, let us consider a non-explicit construction.

Proposition 1 (random linear codes): *Let $c > 1$ and $n, d : \mathbb{N} \to \mathbb{N}$ be such that for every k it holds that $n(k) > c \cdot k/(1 - H_2(d(k)/n(k)))$ and $d(k) < n(k)/2c$, where $H_2(\alpha) \stackrel{\text{def}}{=} \alpha \log_2(1/\alpha) + (1 - \alpha)\log_2(1/1 - \alpha)$. Then, with high probability, a random linear transformation of $\{0,1\}^k$ to $\{0,1\}^{n(k)}$ constitutes a code of distance $d(k)$.*

Thus, *for every constant $\delta \in (0, 0.5)$ there exists a constant $\rho > 0$ and an infinite family of codes $\{C_k : \{0,1\}^k \to \{0,1\}^{k/\rho}\}_{k \in \mathbb{N}}$ of relative distance δ.* Specifically, $\rho = c/(1 - H_2(\delta))$ will do.

[1] This formulation is not the one common in coding theory, but it is the most natural one for our applications. On one hand, this formulation is applicable also to codes with super-polynomial block-length. On the other hand, this formulation does not support a discussion of practical algorithms that compute the codeword faster than by computing each of its bits separately.

Proof: We consider a uniformly selected k-by-$n(k)$ generating matrix over GF(2), and upper-bound the probability that it yields a linear code of distance less than $d(k)$. We use a union bound on all possible $2^k - 1$ linear combinations of the rows of the generating matrix, where for each such combination we compute the probability that it yields a vector of Hamming weight less than $d(k)$. Observe that the result of each such linear combination is uniformly distributed over $\{0,1\}^{n(k)}$, and thus has Hamming weight less than $d(k)$ with probability $\sum_{i=0}^{d(k)-1} \binom{n(k)}{i} \cdot 2^{-n(k)} \approx 2^{-(1-H_2(d(k)/n(k)))\cdot n(k)}$. Using $(1 - H_2(d(k)/n(k))) \cdot n(k) > c \cdot k$, the proposition follows. ■

2.1 A mildly explicit version of Proposition 1

Note that Proposition 1 yields a (deterministic) $\exp(k \cdot n(k))$-time algorithm that finds a linear code of distance $d(k)$. The time bound can be improved to $\exp(k + n(k))$, by observing that we may choose the rows of the generating matrix one by one, making sure that all non-empty linear combinations of the current rows have weight at least $d(k)$. Note that the proof of Proposition 1 can be adapted to assert that as long as we have less than k rows a random choice of the next row will do with high probability. Note that in the case that $n(k) = O(k)$, this yields an algorithm that runs in time that is polynomial in the size of the code (i.e., the number of codewords). Needless to say, this mild level of explicitness is inadequate for most coding applications; however, it will be useful to us in Section 2.5.

2.2 The Hadamard Code

The Hadamard code is the longest (non-repetitive) linear code over $\{0,1\} \equiv \mathrm{GF}(2)$. That is, $x \in \{0,1\}^k$ is mapped to the sequence of all $n(k) = 2^k$ possible linear combinations of its bits (i.e., bit locations in the codewords are associated with k-bit strings, and location $\alpha \in \{0,1\}^k$ in the codeword of x holds the value $\sum_{i=1}^k \alpha_i x_i$). It can be verified that each non-zero codeword has weight 2^{k-1}, and thus this code has relative distance $d(k)/n(k) = 1/2$ (albeit its block-length $n(k)$ is exponential in k).

Turning to the computational aspects, we note that encoding is very easy. The proof of the fact that inner-product (mod 2) is a hardcore predicate [14] (see also [12, Thm. 2.5.2]), provides a very fast probabilistic algorithm for list decoding. An even faster algorithm for unique decoding is provided in the warm-up discussions towards such a proof (see, e.g., [12, P. 67]).

We note that the Hadamard code has played a key role in the proof of the PCP Theorem [2, 1]; see [1].

A propos long codes. We note that the longest (non-repetitive) binary code (called the Long-Code and introduced in [5]) is extensively used in the design of "advanced" PCP systems (see, e.g., [16, 17]). In this code, a k-bit long string x is mapped to the sequence of $n(k) = 2^{2^k}$ values, each corresponding to the evaluation of a different Boolean function at x; that is, bit locations in the codewords are associated with Boolean functions such that the location associated with $f:\{0,1\}^k \to \{0,1\}$ in the codeword of x holds the value $f(x)$.

2.3 The Reed–Solomon Code

A Reed-Solomon code is defined for a non-binary alphabet, which is associated with a finite field of n elements, denoted $\mathrm{GF}(n)$. For any $k < n$, we consider the mapping of univariate degree $k-1$ polynomials over $\mathrm{GF}(n)$ to their evaluation at all field elements. That is, $p \in \mathrm{GF}(n)^k$ (viewed as such a polynomial), is mapped to the sequence $(p(\alpha_1), ..., p(\alpha_n))$, where $\alpha_1, ..., \alpha_n$ is a canonical enumeration of the elements of $\mathrm{GF}(n)$.[2]

The Reed-Solomon code offers infinite families of codes with constant rate and constant relative distance (e.g., by taking $n(k) = 3k$ and $d(k) = 2k$), but the alphabet size grows with k (or rather with $n(k) > k$). Efficient algorithms for unique decoding and list decoding are known (see [19] and references therein). These computational tasks correspond to the extrapolation of polynomials based on a noisy version of their values at all possible evaluation points.

2.4 The Reed–Muller Code

Reed-Muller codes generalize Reed-Solomon codes by considering multi-variate polynomials rather than univariate polynomials. Consecutively, the alphabet may be any finite field, and in particular the two-element field GF(2). Reed-Muller codes (and variants of them) are extensively used in complexity theory; for example, they underly the hardness amplification of [21], and the basis PCP constructions of [3, 10, 2, 1]. The relevant property of these codes is that, under a suitable setting of parameters that satisfies $n(k) = \mathrm{poly}(k)$, they allow super fast "codeword testing" and "self-correction" (see discussion in Section 3).

For any prime power q and parameters m and r, we consider the set, denoted $P_{m,r}$, of all m-variate polynomials of total degree at most r. Each polynomial in $P_{m,r}$ is represented by the $k = \log_q |P_{m,r}|$ coefficients of all relevant monomials, where in the case that $r < q$ it holds that $k = \binom{m+r}{m}$. We consider the code $C : \mathrm{GF}(q)^k \to \mathrm{GF}(q)^n$, where $n = q^m$, mapping m-variate polynomials of total degree at most r to their values at all q^m evaluation points. That is, the m-variate polynomial p of total degree at most r is mapped to the sequence of values $(p(\overline{\alpha}_1), ..., p(\overline{\alpha}_n))$, where $\overline{\alpha}_1, ..., \overline{\alpha}_n$ is a canonical enumeration of all the m-tuples of $\mathrm{GF}(q)$. The relative distance of this code is lower-bounded by $(q - r)/q$.

In typical applications one sets $r = \Theta(m^2 \log m)$ and $q = \mathrm{poly}(r)$, which yields $k > m^m$ and $n = \mathrm{poly}(r)^m = \mathrm{poly}(m^m)$. Thus we have $n(k) = \mathrm{poly}(k)$ but not $n(k) = O(k)$. As we shall see in Section 3, the advantage (in comparison to the Reed-Solomon code) is that codeword testing and self-correction can be performed at complexity related to $q = \mathrm{poly}(\log n)$. Actually, in most complexity applications, a variant in which only m-variate polynomials of individual degree $r' = r/m$ are used. In this case, an alternative presentation analogous to the one presented in Footnote 2 is preferred: The information is viewed as a function $H^m \to \mathrm{GF}(q)$, where $H \subset \mathrm{GF}(q)$ is of size $r' + 1$, and is encoded by a m-variate polynomial of individual degree r' evaluated at all points in $\mathrm{GF}(q)^m$.

2.5 Binary codes of constant relative distance and constant rate

Recall that we seek binary codes of constant relative distance and constant rate. Proposition 1 asserts that such codes exists, but does not provide an explicit construction. The Hadamard code is explicit but does not have a constant rate (to say the least (since $n(k) = 2^k$)).[3] The Reed-Solomon code has constant relative distance and constant rate but uses a non-binary alphabet (which grows at least linearly with k). We achieve the desired construction by using the paradigm of concatenated codes [11], which is of independent interest. (Indeed, concatenated codes may be viewed as a simple version of the proof composition paradigm of [2].)

[2]Alternatively, we may map $(v_1, ..., v_k) \in \mathrm{GF}(n)^k$ to $(p(\alpha_1), ..., p(\alpha_n))$, where p is the unique univariate polynomial of degree $k-1$ that satisfies $p(\alpha_i) = v_i$ for $i = 1, ..., k$. Note that this modification amounts to a linear transformation of the generating matrix.

[3]Binary Reed-Muller codes also fail to simultaneously provide constant relative distance and constant rate.

Intuitively, concatenated codes are obtained by first encoding information, viewed as a sequence over a large alphabet, by some code and next encoding each resulting symbol, which is viewed as a sequence of over a smaller alphabet, by a second code. Formally, consider $\Sigma_1 \equiv \Sigma_2^{k_2}$ and two codes, $C_1 : \Sigma_1^{k_1} \to \Sigma_1^{n_1}$ and $C_2 : \Sigma_2^{k_2} \to \Sigma_2^{n_2}$. Then, the concatenated code of C_1 and C_2, maps $(x_1, ..., x_{k_1}) \in \Sigma_1^{k_1} \equiv \Sigma_2^{k_1 k_2}$ to $(C_2(y_1), ..., C_2(y_{n_1}))$, where $(y_1, ..., y_{n_1})) = C_1(x_1, ..., x_{k_1})$.

Note that the resulting code $C : \Sigma_2^{k_1 k_2} \to \Sigma_2^{n_1 n_2}$ has constant rate and constant relative distance if both C_1 and C_2 have these properties. Encoding in the concatenated code is straightforward. To decode a corrupted codeword of C, we first apply the decoder of C_2 to the n_1 blocks (which are each a n_2-long sequence over Σ_2), obtaining n_1 sequences over Σ_2, which are each interpreted as a symbol of Σ_1. Next we apply the decoder of C_1 to the resulting n_1-long sequence (over Σ_1), and finally we interpret the resulting k_1-long sequence (over Σ_1) as a $k_1 k_2$-long sequence over Σ_2. The key observation is that if we are given a sequence that is $\varepsilon_1\varepsilon_2$-close to $C(x_1, ..., x_{k_1}) = (C_2(y_1), ..., C_2(y_{n_1}))$ then at least $1 - \varepsilon_1$ of the blocks are ε_2-close to the corresponding $C_2(y_i)$.

We are going to consider the concatenated code obtained by using the Reed-Solomon Code $C_1 : \mathrm{GF}(n_1)^{k_1} \to \mathrm{GF}(n_1)^{n_1}$ as the large code, setting $k_2 = \log_2 n_1$, and using the mildly explicit version of Proposition 1, $C_2 : \{0,1\}^{k_2} \to \{0,1\}^{n_2}$ as the small code. We use $n_1 = 3k_1$ and $n_2 = O(k_2)$, and so the concatenated code is $C : \{0,1\}^k \to \{0,1\}^n$, where $k = k_1 k_2$ and $n = n_1 n_2 = O(k)$. The key observation is that C_2 can be constructed in $\exp(k_2)$-time, whereas here $\exp(k_2) = \mathrm{poly}(k)$. Furthermore, both encoding and decoding with respect to C_2 can be performed in time $\exp(k_2) = \mathrm{poly}(k)$. Thus, we get:

Theorem 2 (an explicit good code): *There exists constants $\delta, \rho > 0$ and an explicit family of binary codes of rate ρ and relative distance at least δ.*[4] *That is, there exists a polynomial-time* (encoding) *algorithm C such that $|C(x)| = |x|/\rho$* (for every x) *and a polynomial-time* (decoding) *algorithm D such that for every y that is $\delta/2$-close to some $C(x)$ it holds that $D(y) = x$. Furthermore, C is a linear code.*

The linearity of C is justified by using a Reed-Solomon code over the extension field $F = \mathrm{GF}(2^{k_2})$, and noting that this code induces a linear transformation over GF(2). Specifically, the value of a polynomial p over F at a point $\alpha \in F$ can be obtained as a linear transformation of the coefficient of p, when viewed as k_2-dimensional vectors over GF(2).

Relative distance approaching one half. Starting with a Reed-Solomon code of relative distance δ_1, we obtain a concatenated code of relative distance $\delta_1/2$. Note that, for any constant $\delta_1 < 1$, there exists a Reed-Solomon code $C_1 : \mathrm{GF}(n_1)^{k_1} \to \mathrm{GF}(n_1)^{n_1}$ of relative distance δ_1 and constant rate (i.e., $1 - \delta_1$). Giving up on constant rate, we may start with a Reed-Solomon code of block-length $n_1(k_1) = \mathrm{poly}(k_1)$ and distance $n_1(k_1) - k_1$ over $[n_1(k_1)]$, which yields a (concatenated) binary code of block length $n(k) \approx n_1(k)^2$ and distance approximately $(n(k) - k \cdot n_1(k))/2$. Thus, the relative distance is approximately $(1/2) - (k/\sqrt{n(k)})$.

3 Two additional computational problems

In this section we briefly review relaxations of two traditional coding theoretic tasks. The purpose of these relaxations is enabling super-fast (randomized) algorithms that provide meaningful information. Specifically, these algorithms may run in sub-linear (e.g., poly-logarithmic) time, and thus cannot possibly solve the unrelaxed version of the problem.

[4] The relative distance of the code may be actually larger than δ. We set δ such that we can guarantee efficient decoding of any input that is $\delta/2$-close to some codeword.

Local testability. This task refers to testing whether a given word is a codeword (in a predetermine code), based on (randomly) inspecting few locations in the word. Needless to say, we can only hope to make an approximately correct decision; that is, accept each codeword and reject with high probability each word that is *far* from the code. (Indeed, this task is within the framework of property testing.)

Local decodability. Here the task is to recover a specified bit in the plaintext by (randomly) inspecting few locations in a mildly corrupted codeword. This task is somewhat related to the task of self-correction (i.e., recovering a specified bit in the codeword itself, by inspecting few locations in the mildly corrupted codeword).

Note that the Hadamard code is both locally testable and locally decodable as well as self-correctable (based on a constant number of queries into the word); these facts were demonstrated and extensively used in [8, 1]. However, the Hadamard code has an exponential block-length (i.e., $n(k) = 2^k$), and the question is whether one can achieve analogous results with respect to a shorter code (e.g., $n(k) = \mathrm{poly}(k)$). As hinted in Section 2.4, the answer is positive (when referring to performing these operations in time that is poly-logarithmic in k):

Theorem 3 *For some constant $\delta > 0$ and polynomials $n, q : \mathbb{N} \to \mathbb{N}$, there exists an explicit family of codes $\{C_k : [q(k)]^k \to [q(k)]^{n(k)}\}_{k \in \mathbb{N}}$ of relative distance δ that can be locally testable and locally decodable in* $\mathrm{poly}(\log k)$*-time. That is, the following three conditions hold.*

1. Encoding: *There exists a polynomial time algorithm that on input $x \in [q(k)]^k$ returns $C_k(x)$.*
2. Local Testing: *There exists a probabilistic polynomial-time oracle machine T that given k* (in binary)[5] *and oracle access to $w \in [q(k)]^{n(k)}$ distinguishes the case that w is a codeword from the case that w is $\delta/2$-far from any codeword. Specifically:*
   (a) *For every $x \in [q(k)]^k$ it holds that $\mathsf{Pr}[T^{C_k(x)}(k){=}1] = 1$.*
   (b) *For every $w \in [q(k)]^{n(k)}$ that is $\delta/2$-far from any codeword of C_k it holds that $\mathsf{Pr}[T^w(k){=}1] \leq 1/2$.*

   As usual, the error probability can be reduced by repetitions.
3. Local Decoding: *There exists a probabilistic polynomial-time oracle machine D that given k and $i \in [k]$* (in binary) *and oracle access to any $w \in [q(k)]^{n(k)}$ that is $\delta/2$-close to $C_k(x)$ returns x_i; that is, $\mathsf{Pr}[D^w(k,i) = x_i] \geq 2/3$.*

   Self correction *holds too: there exists a probabilistic polynomial-time oracle machine M that given k and $i \in [n(k)]$* (in binary) *and oracle access to any $w \in [q(k)]^{n(k)}$ that is $\delta/2$-far from $C_k(x)$ returns $C_k(x)_i$; that is, $\mathsf{Pr}[D^w(k,i) = C_k(x)_i] \geq 2/3$.*

We stress that both oracle machines work in time that is polynomial in the binary representation of k, which means that they run in time that is poly-logarithmic in k. The code asserted in Theorem 3 is a (small modification of a) Reed-Muller code, for $r = m^2 \log m < q(k) = \mathrm{poly}(r)$ and $[n(k)] \equiv \mathrm{GF}(q(k))^m$ (see Section 2.4).[6] The aforementioned oracle machines query the oracle

[5] Thus, the running time of T is $\mathrm{poly}(|k|) = \mathrm{poly}(\log k)$.

[6] The modification is analogous to the one presented in Footnote 2: For a suitable choice of k points $\overline{\alpha}_1, ..., \overline{\alpha}_k \mathrm{GF}(q(k))^m$, we map $v_1, ..., v_k$ to $(p(\overline{\alpha}_1), ..., p(\overline{\alpha}_n))$, where p is the unique m-variate polynomial of degree at most r that satisfies $p(\overline{\alpha}_i) = v_i$ for $i = 1, ..., k$.

$w : [n(k)] \to \mathrm{GF}(q(k))$ at a non-constant number of locations. Specifically, self-correction for location $i \in \mathrm{GF}(q(k))^m$ is performed by selecting a random line (over $\mathrm{GF}(q(k))^m$) that passes through i, recovering the values assigned by w to all $q(k)$ points on this line, and performing univariate polynomial extrapolation (under mild noise). Local testability is easily reduced to self-correction, and local decodability is enabled by the aforementioned modification.

Constant number of queries. The local testing and decoding algorithms asserted in Theorem 3 make a polylogarithmic number of queries into the oracle. In contrast, the Hadamard code supports these operation using a *constant number of queries. Can this be obtained with much shorter codewords?* For local testability the answer is definitely positive. One can obtain such locally testable codes with length that is nearly linear (see [7, 9]). For local decodability based on q queries the shortest known code has codewords of length $n(k) = \exp(k^{O(\log\log q)/q\log q})$; see [4]. We conjecture that for every constant q there exists a constant $c_q > 0$ such that $n(k) \geq \exp(k^{c_q})$ must hold (for local decodability based on q queries). In light of this conjecture, we advocate a relaxation of the local decodability task (e.g., the one studied in [6]).

The interested reader is referred to [13], which includes more details on locally terstable and decodable codes as well as a wider perspective. (Note, however, that this survey was written prior to [9], which resolves the main opend problem posed in [13].)

4 A list decoding bound

A necessary condition for the feasibility of the list decoding task is that the list of codewords that are close to the given word is short. In this section we present an upper-bound on the length of such lists, noting that this bound has found several applications in complexity theory (and specifically to studies related to the contents of this course). In contrast, we do not present far more famous bounds (which typically refer to the relation among the main parameters of codes (i.e., k, n and d)), because they seem irrelevant to the contents of this course.

We start with a general statement that refers to any alphabet $\Sigma \equiv [q]$, and later specialize it to the case that $q = 2$. Especially in the general case, it is natural and convenient to consider the agreement (rather than the distance) between sequences over $[q]$. Furthermore, it is natural to focus on agreement rate of at least $1/q$, and convenient to state the following result in terms of the "excessive agreement rate" (i.e., the excess beyond $1/q$).[7]

Lemma 4 (Part 2 of [15, Thm. 15]): *Let $C : [q]^k \to [q]^n$ be an arbitrary code of distance $d \leq n - (n/q)$, and let $\gamma_{\mathtt{C}} \stackrel{\mathrm{def}}{=} 1 - (d/n) - (1/q) \geq 0$ denote the corresponding upper-bound on the excessive agreement rate between codewords. Suppose that $\gamma_{\mathtt{W}} \in (0,1)$ satisfies*

$$\gamma_{\mathtt{W}} > \sqrt{\left(1 - \frac{1}{q}\right) \cdot \gamma_{\mathtt{C}}} \tag{1}$$

Then, for any $w \in [q]^n$, the number of codewords that are at distance at most $(1 - ((1/q) + \gamma_{\mathtt{W}})) \cdot n$ from w (i.e., agree with w on at least $((1/q) + \gamma_{\mathtt{W}}) \cdot n$ positions) *is upper-bounded by*

$$\frac{(1 - (1/q))^2 - (1 - (1/q)) \cdot \gamma_{\mathtt{C}}}{{\gamma_{\mathtt{W}}}^2 - (1 - (1/q)) \cdot \gamma_{\mathtt{C}}} \tag{2}$$

[7] Indeed, we only consider codes with distance $d \leq (1 - 1/q) \cdot n$ and words that are at distance at most d from the code. Note that $1/q$ is a natural threshold for an upper-bound on the relative agreement between sequences over $[q]$, because a random sequence is expected to agree with any fixed sequence on a $1/q$ fraction of the locations.

In the binary case (i.e., $q = 2$), Eq. (1) requires $\gamma_{\tt w} > \sqrt{\gamma_{\tt c}/2}$ and Eq. (2) yields the upper-bound $(1-2\gamma_{\tt c})/(4{\gamma_{\tt w}}^2 - 2\gamma_{\tt c})$. In the case of the Hadamard code, we have $\gamma_{\tt c} = 0$, and so (for every $w \in \{0,1\}^n$ and every $\gamma_{\tt w} > 0$) the number of codewords that are $(0.5 - \gamma_{\tt w})$-close to w is at most $1/(4{\gamma_{\tt w}}^2)$. In the general case (and specifically for $q \gg 2$) it is useful to simplify Eq. (1) by $\gamma_{\tt w} > \min\{\sqrt{\gamma_{\tt c}}, (1/q) + \sqrt{\gamma_{\tt c} - (1/q)}\}$ and Eq. (2) by $\frac{1}{{\gamma_{\tt w}}^2 - \gamma_{\tt c}}$.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof Verification and Intractability of Approximation Problems. *Journal of the ACM*, Vol. 45, pages 501–555, 1998. Preliminary version in *33rd FOCS*, 1992.

[2] S. Arora and S. Safra. Probabilistic Checkable Proofs: A New Characterization of NP. *Journal of the ACM*, Vol. 45, pages 70–122, 1998. Preliminary version in *33rd FOCS*, 1992.

[3] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking Computations in Polylogarithmic Time. In *23rd ACM Symposium on the Theory of Computing*, pages 21–31, 1991.

[4] A. Beimel, Y. Ishai, E. Kushilevitz, and J.F. Raymond. Breaking the $O(n^{1/(2k-1)})$ barrier for information-theoretic private information retrieval. In *43rd IEEE Symposium on Foundations of Computer Science*, pages 261–270, 2002.

[5] M. Bellare, O. Goldreich and M. Sudan. Free Bits, PCPs and Non-Approximability – Towards Tight Results. *SIAM Journal on Computing*, Vol. 27, No. 3, pages 804–915, 1998. Extended abstract in *36th FOCS*, 1995.

[6] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan. Robust PCPs of proximity, Shorter PCPs and Applications to Coding. In *36th ACM Symposium on the Theory of Computing*, pages 1–10, 2004.

[7] E. Ben-Sasson and M. Sudan. Simple PCPs with Poly-log Rate and Query Complexity. *ECCC*, TR04-060, 2004.

[8] M. Blum, M. Luby and R. Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. *Journal of Computer and System Science*, Vol. 47, No. 3, pages 549–595, 1993.

[9] I. Dinur. The PCP Theorem by Gap Amplification. *ECCC*, TR05-046, 2005.

[10] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating Clique is almost NP-complete. *Journal of the ACM*, Vol. 43, pages 268–292, 1996. Preliminary version in *32nd FOCS*, 1991.

[11] G.D. Forney. *Concatenated Codes.* MIT Press, Cambridge, MA 1966.

[12] O. Goldreich. *Foundation of Cryptography: Basic Tools.* Cambridge University Press, 2001.

[13] O. Goldreich. Short Locally Testable Codes and Proofs (Survey). *ECCC*, TR05-014, 2005.

[14] O. Goldreich and L.A. Levin. Hard-core Predicates for any One-Way Function. In *21st ACM Symposium on the Theory of Computing*, pages 25–32, 1989.

[15] O. Goldreich, R. Rubinfeld and M. Sudan. Learning polynomials with queries: the highly noisy case. *SIAM J. Discrete Math.*, Vol. 13 (4), pages 535–570, 2000.

[16] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, Vol. 182, pages 105–142, 1999. Preliminary versions in *28th STOC* (1996) and *37th FOCS* (1996).

[17] J. Håstad. Getting optimal in-approximability results. In *29th ACM Symposium on the Theory of Computing*, pages 1–10, 1997.

[18] F. MacWilliams and N. Sloane. *The theory of error-correcting codes.* North-Holland, 1981.

[19] M. Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, Vol. 13 (1), pages 180–193, 1997.

[20] M. Sudan. Algorithmic introduction to coding theory. Lecture notes, Available from `http://theory.csail.mit.edu/~madhu/FT01/`, 2001.

[21] , M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR Lemma. *Journal of Computer and System Science*, Vol. 62, No. 2, pages 236–266, 2001.