Lecture 1 in "Robust computation: from local pieces to global structure"

Irit Dinur

July 25, 2025

1 Introduction - robust local to global

When studying large complex objects (a large set of data or a large computation) we often look at small pieces of it that are easy to understand, and then study how these relate to the global object. This question occurs in many scenarios in computer science: in error correcting codes, in probabilistically checkable PCPs, in hardness of approximation of constraint satisfaction problems (CSPs), and more.

We will look into some nice local to global techniques and theorems, and show how they apply in a variety of settings.

When can a global object like a proof be broken down into local pieces? A proof is classically thought of as a complex global object. Of course verification can be broken into local steps, but if even one piece is wrong, the entire correctness breaks down. Can this be robustified? A priori it seems impossible, and that's the power of the PCP theorem.

The PCP theorem says that proofs can be verified by reading only a few bits, with good confidence. This is a very robust way to verify correctness, and it is the essence of the PCP theorem.

2 Example: verifying a system of linear equations

Let us begin with a simple but important example, which we can call the bank statement example. The input is a sequence of numbers $a_1, \ldots, a_n$, and we want to verify that the bottom line of a bank statement is correct, namely $S = \sum_{i=1}^{n} a_i$. We can think of the bank statement as a system of equations that describes the bottom line, like so:

$$s_1 = a_1$$

$$
\begin{aligned}
s_2 &= a_2 + s_1 \\
s_3 &= a_3 + s_2 \\
&\vdots \\
s_n &= a_n + s_{n-1}
\end{aligned}
$$

Here the variables are $s_1, \ldots, s_n$, and the input $a_1, \ldots, a_n$ is viewed as constants in the equations. How robust is this system of equations? If the bank is cheating me and giving me a false bottom line, how many equations does it need to violate?

While we think about this, let us look at a simpler system of equations. Let the variables be $x_1, \ldots, x_n$, and let the equations be

$$x_i = x_j, \qquad \forall \{i, j\} \in E.$$

This system is very naturally described by a graph whose vertices are $[n]$ and whose edges are E.

It is easy to see that the graph is connected iff the only solutions are constant (namely there is some $b \in \{0, 1\}$ such that $x_i = b$ for all i). How robust is this system? If we have a solution that satisfies most of the equations, is it "close" to a constant solution?

This depends on the graph. If the graph is a long path, for example, then the answer is no (Why?). If the graph is a grid, again the answer is no. But, if the graph is an expander, the answer is yes. In fact, this is really an *equivalent* definition of expansion: whenever there is a non constant assignment of values, compare the fraction of violated equations (edges crossing between color classes) and the distance of the assignment to the constant functions.

This was a very simple example, with equality equations. Going back to the bank statement example, that system looks like a path, so can easily be "broken". Can this be improved?

For enhancing robustness, a main idea is to use encodings. Instead of providing the solution itself, we can ask for an encoding of the solution, potentially in a way that enables easy verification.

We will allow equations to include *more than two* variables each. Now, the system of equations is no longer a graph, but rather a hypergraph. Notions of robustness correspond to new notions of expansion, called *high dimensional expansion*, variants of which we will encounter later on.

Verifying a system of linear equations - formal problem definition

Let us define our problem more formally. The input is a matrix $M \in \mathbb{F}_2^{m \times n}$ and a vector $b \in \mathbb{F}_2^m$. The verifier wants to know if there exists some $a \in \mathbb{F}_2^n$ such that $Ma = b$, but without reading too many bits in a. The prover provides some proof π, which the verifier queries in a few locations and then accepts or rejects (by making some linear computation on the queries). The system has these properties:

- [Completeness:] If there exists a such that $Ma = b$, then there exists a proof π such that the verifier accepts with probability 1. We can further think of π as an encoding of a.
- [Soundness:] If there is no such a, then for every proof π the verifier accepts with probability at most $1 - \varepsilon$. Moreover,
- [Robustness:] If the verifier accepts some π with probability $1 - \delta$, then π must be $O(\delta)$ close to some $Enc(a)$ such that $Ma = b$.

This problem can be viewed as a stripped down version of the PCP problem, where the initial claim is a linear claim rather than a general NP claim. It might seem useless because the verifier can invert M and compute $M^{-1}b$ on its own (with no proof), but it will be an important component towards the PCP theorem.

Gap Amplification

One can view the above as a reduction from a linear system $Mx = b$ to a new (sparse) system $M'y = b'$, in which every row is sparse, together with an encoding map $x \mapsto y$. The reduction satisfies the following.

- [Completeness:] If $Ma = b$ then $M'a' = b'$ for $a' = Enc(a)$.
- [Robustness:] If y satisfies $1-\delta$ of the equations in the second system, namely $M'y - b'$ has weight at most δm, then y must be close to some $Enc(x)$ for x that satisfies $Mx = b$.

3 The Hadamard Code and linearity testing

Here is a proposed way to "robustify" the bank statement system. First, let us abstract the problem. We are given input $S, a_1, \ldots, a_n \in \mathbb{F}_2$ and a linear claim saying that $S = \sum_{i=1}^n a_i$. How can this linear statement be checked?

The idea is to encode the input, in a way that allows us to check the claim locally.

We will use the celebrated Hadamard code. This is a well known error correcting code, that maps n bits to 2^n bits, given by the linear map

$$\forall a \in \mathbb{F}_2^n, \quad a \mapsto f_a$$

where f_a is defined by

$$\forall x \in \mathbb{F}_2^n, \quad f_a(x) = \sum_{i=1}^{n} a_i x_i.$$

The image of this map is an n-dimensional subspace $L \subset \mathbb{F}_2^{2^n} = \{f : \mathbb{F}_2^n \to \mathbb{F}_2\}$, consisting of all possible *linear* functions. We will split this task into two parts:

- Check that the given function f is indeed linear, namely $f = f_a$ for some $a \in \mathbb{F}_2^n$.
- Check that said a satisfies the linear equation $\sum_i \alpha_i a_i = b$.

Blum Luby and Rubinfeld studied the linearity testing problem, and it became the first result in "property testing". The proposed the following test: choose $x, y \in \mathbb{F}_2^n$ uniformly at random, and check whether $f(x+y) = f(x)+f(y)$ holds. It is easy to see that f is linear iff the test always passes,

Claim 3.1. Consider the following system of linear equations,

$$\forall x, y \in \mathbb{F}_2^n, \quad f(x+y) = f(x) + f(y) \tag{3.1}$$

Then

$$L = \{f : \mathbb{F}_2^n \to \mathbb{F}_2 \mid \ f \text{ satisfies all equations in (3.1)}\}$$

Proof. There are two parts to the claim, first, clearly, every linear function f_a, for every a, satisfies the equations.

$$f_a(x+y) = \sum_{i=1}^{n} a_i(x_i + y_i) = \sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n} a_i y_i = f_a(x) + f_a(y).$$

Next, if a function satisfies all equations, then it must be linear. Let us see the latter. Fix $a_i := f(e_i)$ for all $i \in [n]$. We claim that $f = f_a$.

$$f(x) = f\left(\sum_{i=1}^{n} x_i e_i\right) = \sum_{i=1}^{n} x_i f(e_i) = \sum_{i=1}^{n} x_i a_i = f_a(x).$$

where the second equality is by repeated use of the equations in (3.1). □

What BLR showed is much stronger. They showed that the system (3.1) is *robust*, namely that if the test fails with small probability then the function is close to linear.

Theorem 3.2 (Blum-Luby-Rubinfeld (BLR) linearity testing)**.** *If a function* $f : \mathbb{F}_2^n \to \mathbb{F}_2$ *satisfies* $1 - \delta$ *fraction of equations in* (3.1)*, namely*

$$\varepsilon(f) = \mathop{\mathbb{P}}_{x,y \in \mathbb{F}_2^n}[f(x) + f(y) \neq f(x+y)] \leqslant \delta$$

then there exists a nearby function $\ell \in L$*,* $\mathrm{dist}(f, \ell) \leqslant O(\delta)$

4 Proof of Theorem 3.2

There are two different and insightful proofs for this theorem. A very short one, based on Fourier analysis, and a more constructive one, based on the local correction. We will see both proofs in this course, starting from the constructive one. Let us define a function ℓ by a majoriy vote over all equations involving $f(x)$, namely:

$$\ell(x) = \mathrm{Maj}_{y \in \mathbb{F}_2^n}[f(y) + f(x+y)] \tag{4.1}$$

where Maj is the majority function, which outputs 1 if the input is 1 on more than half of the inputs, and 0 otherwise. We can think of ℓ as a correction to f. If f satisfies all equations, then $\ell = f$. We will prove that if f satisfies ore than $7/9$ of the equations, then ℓ is both close to f, and in L.

Define the probability of the popular vote for x as

$$P_x = \mathop{\mathbb{P}}_{y}[f(y) + f(x+y) = \ell(x)]$$

We will show two claims, the first showing that P_x is large for all x, and the second showing that ℓ is linear. Finally, it will be easy to show that $f \approx \ell$.

Claim 4.1. If $\varepsilon(f) < 2/9$ then for all $x \in \mathbb{F}_2^n$, $P_x > 2/3$.

Proof. Fix x, and choose y and z independently uniformly at random. On one hand,

$$P_x = \mathop{\mathbb{P}}_{y}[f(y) + f(x+y) = \ell(x)] \tag{4.2}$$

$$P_x = \mathop{\mathbb{P}}_{z}[f(z) + f(x+z) = \ell(x)] \tag{4.3}$$

thus the probability that $f(y)+f(x+y) = f(z)+f(x+z)$ is $(P_x)^2+(1-P_x)^2$ because y and z are independent. Switching terms around this holds iff

$$f(y) + f(z) = f(x + y) + f(x + z)$$

Now by the assumption of the theorem, with probability at least $1 - \delta$, $f(y)+f(z) = f(y+z)$. Similarly, with probability $1-\delta$, $f(x+y)+f(x+z) = f(y + z)$ since $x + y, x + z$ are uniform and independent. So by a union bound the probability that both hold (now there is no independence!) is at least $1 - 2\delta \geqslant 5/9$. So

$$P_x^2 + (1 - P_x)^2 > 5/9$$

and this can only hold if $P_x < 1/2$ or $P_x > 2/3$. The first is ruled out since $P_x \geqslant 1/2$. □

Claim 4.2. For every x, y, $\ell(x) + \ell(y) = \ell(x + y)$, hence $\ell \in L$.

Proof. Fix x, y. By the previous claim, $P_x > 2/3, P_y > 2/3$ and $P_{x+y} > 2/3$, so each of the following three equations hold with probability above $2/3$ over the choice of z:

$$\begin{aligned} \ell(x) &= f(z) + f(x + z) \\ \ell(y) &= f(z) + f(y + z) \\ \ell(x + y) &= f(x + z) + f(y + z) \end{aligned}$$

The only slightly tricky statement is the third equation. Note that choosing a random z and setting $w = x + z$ gives a uniformly distributed w and then $\ell(x + y) = f(w) + f(w + (x + y)) = f(x + z) + f(y + z)$ for more than $2/3$ of the choices of w which means $2/3$ of the choices of z.

In combination, each equation fails for less than $1/3$ of the z's, so there must be some z_0 for which all three equations hold simultaneously. If we sum them up we get identically zero on the right hand side, implying that $\ell(x) + \ell(y) + \ell(x + y) = 0$. □

That's great, we have shown that our majority vote function ℓ is in fact linear. It remains to observe that it is close to f.

Whenever $f(x) \neq \ell(x)$ we have that (since $P_x > 2/3$) for at least $2/3$ of the choices of y, $f(x) \neq f(y) + f(x + y)$ holds. So we have

$$\begin{aligned} \varepsilon(f) &= \mathop{\mathbb{E}}_{x} \mathop{\mathbb{P}}_{y}[f(x) \neq f(y) + f(x + y)] \\ &\geqslant \mathop{\mathbb{P}}_{x}[f(x) \neq \ell(x)] \cdot \mathop{\mathbb{E}}_{x:f(x)\neq\ell(x)} \mathop{\mathbb{P}}_{y}[f(x) \neq f(y) + f(x + y)] \\ &= \operatorname{dist}(f, \ell) \cdot \mathop{\mathbb{E}}_{x:f(x)\neq\ell(x)} P_x \end{aligned}$$

$$\geqslant \quad \mathrm{dist}(f,\ell)\cdot\frac{2}{3}$$

where in the first inequality we used the formula $\mathbb{E}[A] \geqslant \mathbb{P}[B]\cdot\mathbb{E}[A|B]$. As long as $\varepsilon(f) \leqslant 2/9$ we have shown that

$$\varepsilon(f) \geqslant \frac{2}{3}\cdot\mathrm{dist}(f,\ell) \geqslant \frac{2}{3}\,\mathrm{dist}(f,L)$$

and altogether $\varepsilon(f) \geqslant \min(\frac{2}{9}, \mathrm{dist}(f,L)\cdot\frac{2}{3}) \geqslant \frac{2}{9}\,\mathrm{dist}(f,L)$.

4.1 Testing a system of linear equations

Given that we can test linearity, we can now test any system of equations! Suppose the system is $Ma = b$ for some fixed M, b. Take a random linear combination of the rows of M, namely let yM for some random vector y, and read off $f(x) + f(x + yM)$. We are relying on the fact that the Hadamard code is a locally correctible code: if we want to know the value at $x_0 = yM$ we make two queries, to x and to $x + x_0$ for a uniformly random x.

4.2 Locally testable code

We have proven that the Hadamard code is an LTC. Other codes (with much better rate) are also LTCs, such as the Reed-Muller code, and some newer codes defined on high dimensional expanders.

These other codes don't always allow for such elegant local correction, so it is open whether they provide a way to check arbitrary linear constraints.

5 Geometry of the linearity testing constraints

We defined the Hadamard code via its "parity checks", namely via the linear constraints defining it (3.1), which are the triples

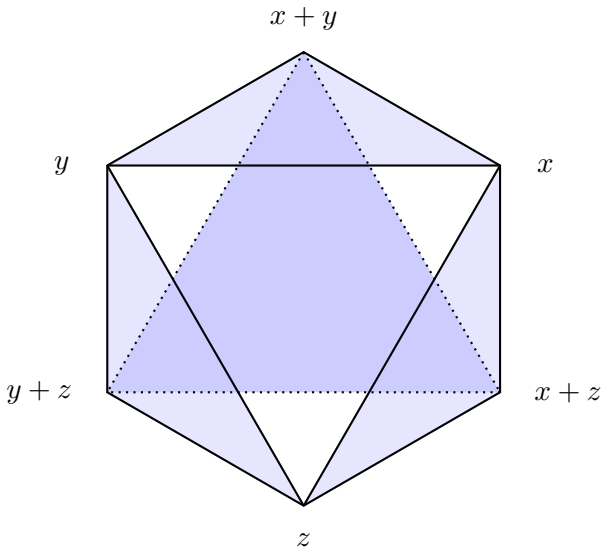
$$\{x, y, x+y\} \quad \forall x \neq y \in \mathbb{F}_2^n \setminus \{0\}.$$

Each constraint can be viewed as a row in the parity check matrix of the Hadamard code, which has three non-zero entries. There are $(2^n-1)(2^{n-1}-1) \approx 2^{2n-1}$ constraints and clearly they are not all linearly independent. In fact, the proof heavily relies on having very short dependencies among these constraints. The following table shows four constraints, whose non zeros are indexed by appropriate choice of three out of $x, y, z, x+y, x+z, y+z$.

x	y	z	x+y	x+z	y+z
	1	1			1
1		1		1	
			1	1	1
1	1		1		

These four constraints are dependent, as can be seen from the fact that the four rows sum to zero. This fact is at the heart of the proof of Claim 4.2.

It is interesting that in this table, every two rows intersect on exactly one out of the three non-zeros. In fact, geometrically, we can draw the constraints as hyperedges (triangles) in a hyper-graph whose vertices are $\mathbb{F}_2^n \setminus \{0\}$. The four constraints correspond to the four triangles in the figure below (one triangle is shown in a different color to make the figure more legible).



One way to understand this figure is by starting with a single violated hyperedge. One violated constraint immediately implies that at least one of the other constraints is violated. This is because the four constraints must some to zero. Therefore, if we look at all possible hyperedges touching a given point, say x, every pair of violated and non-violated constraint span a new violated constraint that *doesn't touch* x.

We can think about a random walk that moves from triangle to triangle in a way that the violated triangles propagate.