1 Linearity

What does it mean for a boolean function to be *linear*? For the question to make sense, we must have a notion of adding two binary strings. So let’s take

\[
f : \{0, 1\}^n \to \{0, 1\}
\]

and treat \( \{0, 1\} \) as \( \mathbb{F}_2 \).

Now there are two well-known classical notions of being linear:

**Definition 1.1**

1. \( f \) is linear iff \( f(x + y) = f(x) + f(y) \) for all \( x, y \in \{0, 1\}^n \).
2. \( f \) is linear iff there are some \( a_1, \ldots, a_n \in \mathbb{F}_2 \) such that \( f(x_1, \ldots, x_n) = a_1 x_1 + \cdots + a_n x_n \)
   \[
   \iff \text{there is some } S \subseteq [n] \text{ such that } f(x) = \sum_{i \in S} x_i.
   \]

(Sometimes in (2) one allows an additive constant; we won’t, calling such functions *affine*.)

Since these definitions sound equally good we may hope that they’re equivalent; happily, they are. Now (2) ⇒ (1) is easy:

\[
(2) \Rightarrow (1) : \quad f(x + y) = \sum_{i \in S} (x + y)_i = \sum x_i + \sum y_i = f(x) + f(y).
\]

But (1) ⇒ (2) is a bit more interesting. The easiest proof:

\[
(1) \Rightarrow (2) : \quad \text{Define } \alpha_i = f(0, \ldots, 0, 1, 0, \ldots, 0). \text{ Now repeated use of condition } 1 \text{ implies } f(x^1 + x^2 + \cdots + x^n) = f(x^1) + \cdots + f(x^n), \text{ so indeed}
\]

\[
f((x_1, \ldots, x_n)) = f(\sum x_i e_i) = \sum x_i f(e_i) = \sum \alpha_i x_i.
\]
1.1 Approximate Linearity

Nothing in this world is perfect, so let’s ask: What does it mean for \( f \) to be *approximately linear*? Here are the natural first two ideas:

**Definition 1.2**

1. \( f \) is approximately linear if \( f(x + y) = f(x) + f(y) \) for most pairs \( x, y \in \{0, 1\}^n \).
2. \( f \) is approximately linear if there is some \( S \subseteq [n] \) such that \( f(x) = \sum_{i \in S} x_i \) for most \( x \in \{0, 1\}^n \).

Are these two equivalent? It’s easy to see that (2) \( \Rightarrow \) (1) still essentially holds: If \( f \) has the right value for both \( x \) and \( y \) (which happens for most pairs), the equation in the (2) \( \Rightarrow \) (1) proof holds up.

The reverse implication is not clear: Take any linear function and mess up its values on \( e_1, \ldots, e_n \). Now \( f(x + y) = f(x) + f(y) \) still holds whenever \( x \) and \( y \) are not \( e_i \)’s, which is true for almost all pairs. But now the equation in the (1) \( \Rightarrow \) (2) proof is going to be wrong for very many \( x \)’s. So this proof doesn’t work — but actually our \( f \) does satisfy (2), so maybe a different proof will work.

We will investigate this shortly, but let’s first decide on (2) as our official definition:

**Definition 1.3** \( f, g : \{0, 1\}^n \to \{0, 1\} \) are \( \epsilon \)-close if they agree on a \( (1 - \epsilon) \)-fraction of the inputs \( \{0, 1\}^n \). Otherwise they are \( \epsilon \)-far.

**Definition 1.4** \( f \) is \( \epsilon \)-close to having property \( P \) if there is some \( g \) with property \( P \) such that \( f \) and \( g \) are \( \epsilon \)-close.

A “property” here can really just be any collection of functions. For our current discussion, \( P \) is the set of \( 2^n \) linear functions.

1.2 Testing Linearity

Given that we’ve settled on definition (2), why worry about definition (1)? Imagine someone hands you some black-box software \( f \) that is supposed to compute some linear function, and your job is to test it — i.e., try to identify bugs. You can’t be sure \( f \) is perfect unless you “query” its value \( 2^n \) times, but perhaps you can become convinced \( f \) is \( \epsilon \)-close to being linear with many fewer queries.

If you knew which linear function \( f \) was supposed to be close to, you could just check it on \( O(1/\epsilon) \) many random values — if you found no mistakes, you’d be quite convinced \( f \) was \( \epsilon \)-close to linear.
Now if you just look at definition \((2')\), you might think that all you can do is make \(n\) linearly independent queries to first determine which linear function \(f\) is supposed to be, and then do the above. (We imagine that \(n \gg 1/\epsilon\).) But it’s kind of silly to use complexity \(n\) to “test” a program that can itself be implemented with complexity \(n\). But if \((1') \Rightarrow (2')\), it would give a way to give a much more efficient test. This was suggested and proved by M. Blum, Luby, and Rubinfeld in 1990:

**Definition 1.5** The “BLR Test”: Given an unknown \(f : \{0, 1\}^n \rightarrow \{0, 1\}\):

- Pick \(x\) and \(y\) independently and uniformly at random from \(\{0, 1\}^n\).
- Set \(z = x + y\).
- Query \(f\) on \(x, y,\) and \(z\).
- “Accept” iff \(f(z) = f(x) + f(y)\).

Today we will prove:

**Theorem 1.6** Suppose \(f\) passes the BLR Test with probability at least \(1 - \epsilon\). Then \(f\) is \(\epsilon\)-close to being linear.

Given this, suppose we do the BLR test \(O(1/\epsilon)\) times. If it never fails, we can be quite sure the true probability \(f\) passes the test is at least \(1 - \epsilon\) and thus that \(f\) is \(\epsilon\)-close to being linear.

NB: BLR originally proved a slightly weaker result than Theorem 1.6 (they lost a constant factor). We present the ‘95 proof due to Bellare, Coppersmith, Håstad, Kiwi, and Sudan.

2 The Fourier Expansion

Suppose \(f\) passes the BLR test with high probability. We want to try showing that \(f\) is \(\epsilon\)-close to some linear function. But which one should we pick?

There’s a trick answer to this question: We should pick the closest one! But given \(f : \{0, 1\}^n \rightarrow \{0, 1\}\), how can we decide which linear function \(f\) is closest to?

Stack the \(2^n\) values of \(f(x)\) in, say, lexicographical order, and treat it as a vector in \(2^n\)-dimensional space, \(\mathbb{R}^{2^n}\):

\[
\begin{bmatrix}
0 \\
1 \\
1 \\
0 \\
\vdots \\
1
\end{bmatrix}
\]
Do the same for all \(2^n\) linear (Parity) functions:

\[
\chi_\emptyset = \begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
\vdots \\
0
\end{bmatrix}, \quad \chi_{\{1\}} = \begin{bmatrix}
0 \\
1 \\
0 \\
0 \\
\vdots \\
1
\end{bmatrix}, \quad \ldots, \chi_{[n]} = \begin{bmatrix}
0 \\
1 \\
0 \\
0 \\
\vdots \\
1
\end{bmatrix}
\]

Notation: \(\chi_S\) is Parity on the coordinates in set \(S\); \([n] = \{1, 2, \ldots, n\}\).

Now it’s easy the closest Parity to \(f\) is the physically closest vector.

\[f\text{ is closest to } \chi_{S_1}\]

It’s extra-convenient if we replace 0 and 1 with 1 and \(-1\); then the dot product of two vectors measures their closeness (the bigger the dot product, the closer). This motivates the Great Notational Switch we’ll use 99% of the time.

**Great Notational Switch:** \(0/\text{False} \rightarrow +1, \quad 1/\text{True} \rightarrow -1\).

We think of \(+1\) and \(-1\) here as \textit{real numbers}. In particular, we now have:

\[\text{Addition (mod 2)} \rightarrow \text{Multiplication (in } \mathbb{R} \text{)}\]

We now write:

A generic boolean function: \(f : \{-1, 1\}^n \rightarrow \{-1, 1\}\).

The Parity on bits \(S\) function, \(\chi_S : \{-1, 1\}^n \rightarrow \{-1, 1\}\):
\[ \chi_S(x) = \prod_{i \in S} x_i. \]

We now have:

**Fact 2.1** The dot product of \( f \) and \( \chi_S \), as vectors in \( \{-1, 1\}^{2^n} \), equals

\[ (# \text{ x's such that } f(x) = \chi_S(x)) - (# \text{ x's such that } f(x) \neq \chi_S(x)). \]

**Definition 2.2** For any \( f, g : \{-1, 1\}^n \rightarrow \mathbb{R} \), we write

\[ \langle f, g \rangle = \frac{1}{2^n} (\text{dot product of } f \text{ and } g \text{ as vectors}) = \text{avg}_{x \in \{-1,1\}^n} [f(x)g(x)] = \mathbb{E}_{x \in \{-1,1\}^n} [f(x)g(x)]. \]

We also call this the correlation of \( f \) and \( g \).

**Fact 2.3** If \( f \) and \( g \) are boolean-valued, \( f, g : \{-1, 1\}^n \rightarrow \{-1, 1\} \), then \( \langle f, g \rangle \in [-1, 1] \). Further, \( f \) and \( g \) are \( \epsilon \)-close iff \( \langle f, g \rangle \geq 1 - 2\epsilon \).

Now in our linearity testing problem, given \( f : \{-1, 1\}^n \rightarrow \{-1, 1\} \) we are interested in the Parity function having maximum correlation with \( f \). Let’s give notation for these correlations:

**Definition 2.4** For \( S \subseteq \llbracket n \rrbracket \), we write

\[ \hat{f}(S) = \langle f, \chi_S \rangle \]

Now with the switch to \(-1,1\), something interesting happens with the \( 2^n \) Parity functions; they become orthogonal vectors:

**Proposition 2.5** If \( S \neq T \) then \( \chi_S \) and \( \chi_T \) are orthogonal; i.e., \( \langle \chi_S, \chi_T \rangle = 0 \).

**Proof:** Let \( i \in S \Delta T \) (the symmetric difference of these sets); without loss of generality, say \( i \in S \setminus T \). Pair up all \( n \)-bit strings: \((x, x^{(i)})\), where \( x^{(i)} \) denotes \( x \) with the \( i \)th bit flipped.

Now the vectors \( \chi_S \) and \( \chi_T \) look like this on “coordinates” \( x \) and \( x^{(i)} \)

\[
\begin{align*}
\chi_S &= \begin{bmatrix} a & -a \end{bmatrix} \\
\chi_T &= \begin{bmatrix} b & b \end{bmatrix}
\end{align*}
\]

\[ \downarrow \quad x \quad \downarrow x^{(i)} \]

for some bits \( a \) and \( b \). In the inner product, these coordinates contribute \( ab - ab = 0 \). Since we can pair up all coordinates like this, the overall inner product is 0. \( \square \)

---

\(^1\)This doesn’t agree with the technical definition of correlation in probability, but never mind.
Corollary 2.6  The set of $2^n$ vectors $(\chi_S)_{S \subseteq [n]}$ form an complete orthogonal basis for $\mathbb{R}^{2^n}$.

Proof: We have $2^n$ mutually orthogonal nonzero vectors in a space of dimension $2^n$. □

Fact 2.7  If $f : \{-1, 1\}^n \to \{-1, 1\}$, "\|f\|" = $\sqrt{\langle f, f \rangle} = 1$.

Corollary 2.8  The functions $(\chi_S)_{S \subseteq [n]}$ form an orthonormal basis for $\mathbb{R}^{2^n}$.

In other words, these Parity vectors are just a rotation of the standard basis.

As a consequence, the most basic linear algebra implies that every vector in $\mathbb{R}^{2^n}$ — in particular, any $f : \{-1, 1\}^n \to \{-1, 1\}$ — can be written uniquely as a linear combination of these vectors:

$$f = \sum_{S \subseteq [n]} c_S \chi_S \quad \text{as vectors, for some } c_S \in \mathbb{R}.$$ 

Further, the coefficient on $\chi_S$ is just the length of the projection; i.e., $\langle f, \chi_S \rangle$:

$$\hat{f}(T) = \langle f, \chi_T \rangle = \langle \sum_s c_s \chi_S, \chi_T \rangle = \sum_s c_s \langle \chi_S, \chi_T \rangle = c_T.$$ 

I.e., we’ve shown:

Theorem 2.9  Every function $f : \{-1, 1\}^n \to \mathbb{R}$ — in particular, every boolean-valued function $f : \{-1, 1\}^n \to \{-1, 1\}$ — is uniquely expressible as a linear combination (over $\mathbb{R}$) of the $2^n$ Parity functions:

$$f = \sum_{S \subseteq [n]} \hat{f}(S) \chi_S. \quad (1)$$

(This is a pointwise equality of functions on $\{-1, 1\}^n$.)

The real numbers $\hat{f}(S)$ are called the Fourier coefficients of $f$, and (1) the Fourier expansion of $f$.

Recall that for boolean-valued functions $f : \{-1, 1\}^n \to \{-1, 1\}$, $\hat{f}(S)$ is a number in $[-1, 1]$ measuring the correlation of $f$ with the function Parity-on-$S$. In (1) we have the property that for every string $x$, the $2^n$ real numbers $\hat{f}(S) \chi_S(x)$ “magically” always add up to a number that is either $-1$ or 1.
2.1 Examples

Here are some example functions and their Fourier transforms. In the Fourier expansions, we will write \( \prod_{i \in S} \) in place of \( \chi_S \).

\[
\begin{array}{c|c}
\text{ } & \text{Fourier transform} \\
\hline
f & 1 \\
f(x) = 1 & x_i \\
f(x) = x_i & \frac{1}{2} + \frac{1}{3} x_1 + \frac{1}{3} x_2 - \frac{1}{3} x_1 x_2 \\
\text{AND}(x_1, x_2) & \frac{1}{2} x_1 + \frac{1}{2} x_2 + \frac{1}{2} x_3 - \frac{1}{2} x_1 x_2 x_3 \\
\text{MAJ}(x_1, x_2, x_3) & \frac{1}{2} x_1 + \frac{1}{2} x_2 + \frac{1}{2} x_3 - \frac{1}{2} x_1 x_2 x_3 \\
\end{array}
\]

2.2 Parseval, Plancherel

We will now prove one of the most important, basic facts about Fourier transforms:

**Theorem 2.10 ("Plancherel’s Theorem")** Let \( f, g : \{-1, 1\}^n \to \mathbb{R} \). Then

\[
\langle f, g \rangle = \mathbb{E}_{x \in \{-1, 1\}^n} [f(x)g(x)] = \sum_{S \subseteq [n]} \hat{f}(S)\hat{g}(S).
\]

This just says that when you express two vectors in an orthonormal basis, their inner product is equal to the sum of the products of the coefficients. **Proof:**

\[
\langle f, g \rangle = \left\langle \sum_{S \subseteq [n]} \hat{f}(S)\chi_S, \sum_{T \subseteq [n]} \hat{g}(T)\chi_T \rightangle
\]

\[
= \sum_S \sum_T \hat{f}(S)\hat{g}(T) \langle \chi_S, \chi_T \rangle \quad \text{(by linearity of inner product)}
\]

\[
= \sum_S \hat{f}(S)\hat{g}(S) \quad \text{(by orthonormality of } \chi \text{'s}).
\]

\[\square\]
Corollary 2.11 ("Parseval’s Theorem") Let $f : \{-1, 1\}^n \to \mathbb{R}$. Then

$$\langle f, f \rangle = \mathbb{E}_{x \in \{-1,1\}^n} [f(x)^2] = \sum_{S \subseteq [n]} \hat{f}(S)^2.$$ 

This just says that the squared length of a vector, when expressed in an orthonormal basis, equals the sum of the squares of the coefficients. In other words, it’s the Pythagorean Theorem.

One very important special case:

Corollary 2.12 If $f : \{-1, 1\}^n \to \{-1, 1\}$ is a boolean-valued function,

$$\sum_{S \subseteq [n]} \hat{f}(S)^2 = 1.$$