REU 2006 · Discrete Math · Lecture 16

Instructor: László Babai
Scribe: Elizabeth Beazley
Editor: Eliana Zoque

August 7, 2006. Last updated August 7, 2006 at 11:15 p.m.
NOT PROOF-READ

16 Hadamard Matrices

16.1 Results about Hadamard Matrices

How do we construct a Hadamard matrix; *i.e.*, an $N \times N$ matrix H such that $HH^t = NI$ and the entries will all be ± 1. This condition implies that $N = 2$ or $4|N$, as we will prove now.

If the first row consists of all 1's, then all subsequent rows have half of their entries 1 and the other half -1. Let's assume,rearranging the rows if necessary, that second row has the first half of the row 1's and the second half -1's. If the third row has more 1's than -1's on the first half, then it is going to have more -1's than 1's on the second half, and the inner product between the second and third row would not be zero, as in the following matrix. The same argument applies if the third row has more -1's than 1's on the first half.

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix}$$

Then the third row has as many 1's a -1's on the first half. Rearranging the rows we have $\frac{N}{4}$ 1's and $\frac{N}{4} - 1$'s, followed by $\frac{N}{4}$ 1's and then $\frac{N}{4} - 1$'s, as in the following matrix. This implies that $4|N$.

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{pmatrix}$$

Note that being Hadamard does not change if we multiply a row through by -1; similarly for a column. Thus, we can indeed assume WLOG that the first row consists of all 1's.

Let's focus on constructing a $(p+1) \times (p+1)$ matrix, where $p \equiv -1 \bmod 4$ is a prime.

First we construct an auxiliary matrix C in the following way: In the ij^{th} spot, let us put the Legendre symbol $(\frac{i-j}{p})$. If we multiply CC^t, the diagonal consists of entries $p-1$.

Computing the inner product of rows h and i, we have

$$\sum_{j=0}^{p-1} \left(\frac{h-j}{p}\right)\left(\frac{i-j}{p}\right) = \sum_{j=0}^{p-1} \left(\frac{(h-j)(i-j)}{p}\right)$$

Note that if $j = h$ or i, then the product inside the sum is 0. Denote by $\chi(a) := \left(\frac{a}{p}\right)$. Then we see that $\chi(h-j)\chi(i-j) = \frac{\chi(h-j)}{\chi(i-j)} = \chi\left(\frac{h-j}{i-j}\right)$, where $i \neq j$. Denote this function of the character χ by $f(x) := \frac{h-x}{i-x}$. Then $f : \mathbb{F}_p - \{i\} \to \mathbb{F}_p$.

Question 16.1.1. Do we see a number that is not in the range of the function f?

The number 1 never occurs, since $h \neq i$; *i.e.*, that we are looking at distinct rows.

Claim 16.1.2. *$f : \mathbb{F}_p - \{i\} \to \mathbb{F}_p - \{1\}$ is a bijection.*

Proof. We shall argue that f is onto, which implies the result. We claim that $(\forall z \neq 1)(\exists x)(\frac{h-x}{i-x} = z)$. We need only show that we can solve this equation:

$$\begin{aligned}(h-x) = z(i-x) &\iff h - x = zi - zx \iff zx - x = zi - h \\ &\iff (z-1)x = zi - h \iff x = \frac{zi-h}{z-1} \quad (16.1.1)\end{aligned}$$

And this final equality has a solution, since $z \neq 1$. □

As a consequence, we have that the values of $\chi(f(x))$ are: 0 for $x = h$; *i.e.*, if $f(x) = 0$. Since $f(x) \neq 1$, then we get the value 1 exactly $\frac{p-3}{2}$ times, and we have -1 exactly $\frac{p-1}{2}$ times. This tells us that this matrix has 0's on the diagonal and ± 1 elsewhere; and that the inner product of two distinct rows equals -1.

Now we augment the matrix by a row and a column of all one's and replacing all zeros by -1. The dot product of a row with itself is $(p-1) + 1 + 1 = p + 1$. Consider two distinct rows h and i. The hi and ih entries are $\left(\frac{h-i}{p}\right)$ and $\left(\frac{i-h}{p}\right)$, respectively. Note that $\left(\frac{-j}{p}\right) = -\left(\frac{j}{p}\right)$, since $\left(\frac{-1}{p}\right) = -1$ because $p \equiv -1 \mod 4$. Thus the dot product of the rows h and i is $-1 + 1 + \left(\frac{h-i}{p}\right) + \left(\frac{i-h}{p}\right) = 0$.

16.2 Eigenvalues of Hadamard Matrices

Claim 16.2.1. *If λ is an eigenvalue of an $n \times n$ Hadamard matrix, then $|\lambda| = \sqrt{N}$*

Recall that orthogonal matrices A are such that $AA^t = I$.

Claim 16.2.2. *If μ is an eigenvalue of A, then $|\mu| = 1$.*

To prove this, one can argue that $\|Ax\| = \|x\|$ and so $Ax = \mu x$. Thus, $\|\mu x\| = |\mu| \cdot \|x\|$. But there is a problem with this argument: if $A = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$, then the characteristic polynomial is

$$f_A(x) = \det(xI - A) = \begin{vmatrix} x - \cos\alpha & -\sin\alpha \\ \sin\alpha & x - \cos\alpha \end{vmatrix} = (x - \cos\alpha)^2 + (\sin\alpha)^2$$
$$= x^2 - 2\cos\alpha x + 1.$$

But note that $f_A(x) = 0 \iff (x - \cos\alpha)^2 = -(\sin\alpha)^2 \iff x - \cos\alpha = \pm i \sin\alpha \iff x = \cos\alpha \pm i \sin\alpha$.

This proof only goes through if μ is real! But in general, $\mu \in \mathbb{C}$. How do we fix it?

Lemma 16.2.3. *If $AA^t = I$, then $\|Ax\|^2 = \|x\|^2$.*

Proof. $(Ax)^t(Ax) = x^t A^t A x = x^t x = \sum {x_i}^2$. □

Recall that the norm in $\mathbb{C}$ is given by $\|x\|^2 = \sum |x_i|^2$, so that $x \cdot y = \sum_{i=1}^{n} \overline{x_i} y_i$.

Definition 16.2.4. Denote by A^* the conjugate-transpose of the complex matrix A. We say that A is a *unitary* matrix if $AA^* = I$. Equivalently, we could require that $A^*A = I$.

Definition 16.2.5. If A is unitary and real, then A is *orthogonal*.

Lemma 16.2.6. *If A is unitary and $x \in \mathbb{C}^n$, then $\|Ax\| = \|x\|$.*

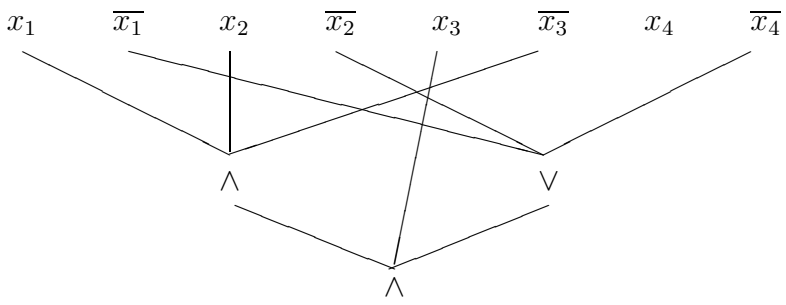
Proof. $\|Ax\|^2 = (Ax)^*(Ax) = x^*A^*Ax = x^*x = \|x\|^2$. □

Using this Lemma and the definition of unitary matrices, the same proof we provided in our first attempt will now go through.

16.3 Boolean circuits

Input $x_1, \ldots, x_n \in \{0, 1\}$. Denote by "and" =:$\wedge$, "or" =: $\vee$ and the negation of x by $\overline{x}$.

Example 16.3.1. If we let $x_1 = 1, x_2 = 1, x_3 = 0$, and $x_4 = 0$, then the following is an unsatisfiable Boolean circuit:



We will introduce several bits of complexity. First, the size of the circuit. At the minimum, we want to keep the circuit size polynomial in n, where n is the number of wires in the Boolean circuit. The second parameter is the *depth*, or the largest path from input to output.

Definition 16.3.2. A *Boolean function* is a function $f : \{0, 1\}^n \to \{0, 1\}$

Note that $\overline{(a \vee b)} = \overline{a} \wedge \overline{b}$.

Example 16.3.3. If negation permitted along wires, then such a circuit can be simulated by a circuit on which all negations are at the input level and newsize = O(oldsize), with newdepth = olddepth.

Claim 16.3.4. *Every Boolean function is computed by a Boolean circuit of depth 2.*

This can be done in two ways, depending on the type of logic symbols that we use in each level.

Conjunctive Normal Form: AND or OR's.

Disjunctive Normal Form: OR of AND's.

Here we have a string of literals, or variables and their negations:

$$x_1 \ \overline{x_1} \ x_2 \ \overline{x_2} \ x_3 \ \overline{x_3} \ x_4 \ \overline{x_4}$$

Take some and's among these literals, and then below, take the or of these and's. Specifically, every and has n terms, consisting of either the variable or its negation. Then we take the or of all of these and's.

Definition 16.3.5. A *complete clause* of literals is a string of literals such that $(\forall i)(x_1$or $\overline{x_1}$ is included)

In a Disjunctive Normal Form (DNF), we do not necessarily require that the clauses be complete.

Example 16.3.6. Here is an example of a DNF: $(x_1 \wedge \overline{x_2}) \vee (\overline{x_1} \wedge x_3 \wedge x_4) \vee (x_2 \wedge \overline{x_3} \wedge x_4)$

Note that $(x_1 \wedge \phi) \vee (\overline{x_1} \wedge \phi) \leftrightarrow \phi$.

Exercise 16.3.7 (Shannon). For almost all Boolean functions, the circuit size is ckt size $= \Omega\left(\frac{2^n}{n}\right)$.

Note here that there is no condition on the depth in the above exercise.

Exercise 16.3.8. Every CNF for parity has $\geq 2^{n-1}$ clauses.

Exercise 16.3.9. Every depth-2 circuit for parity has size $\Omega(n2^n)$.

Question 16.3.10. What is the number of Boolean functions in a given number of variables, say n?

The number of Boolean functions is 2^{2^n}.

Definition 16.3.11. $\text{PARITY}(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i \mod 2$.

Exercise 16.3.12. Compute parity in depth-3, size $\approx 2^{\sqrt{n}}$. Here, it is OK if $n^c 2^{\sqrt{n}}$.

HINT: Suppose we have a parity gate $\bigoplus$ having $\sqrt{n}$ inputs. We could group each $\sqrt{n}$ collection with a parity gate, and then connect all of these parity gates with a parity gate. The output would then be the parity of the parity, which computes the total parity. Altogether, we would have size $= \sqrt{n}2^{\sqrt{n}}$.

Question 16.3.13. How can we compress this circuit to depth-3?

Suppose we have a Boolean circuit having or's on the first level and only and's on the next two levels, followed by a fourth level having only or's. This is a depth-4 Boolean circuit, which we can make depth-3 by compressing the second and third levels, since and(and) = and. If we perform this compression on our circuit of parity gates, this will yield the depth-3 result.

We can compute parity in depth-d with size $\approx 2^{n^{(\frac{1}{d-1})}}$.

Exercise 16.3.14. Compute the sum of two n-bit integers in bounded depth and polynomial size. (You will actually use depth 3).

Theorem 16.3.15 (Ajtai, Furst-Saxe-Sipsor). *PARITY cannot be computed in bounded depth and polynomial size.*

Exercise 16.3.16 (Corollary). Multiplication of n-bit integers cannot be done in bounded depth and polynomial size.

Theorem 16.3.17 (Yao, Hastad). *PARITY in depth d requires size $2^{\frac{n}{10}^{(\frac{1}{d-1})}}$.*

Definition 16.3.18. $\text{MOD}_3(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } \sum x_i \equiv 0 \mod 3 \\ 0 & \text{otherwise} \end{cases}$

Theorem 16.3.19 (Razborov). *Even with MOD_2 gates, MOD_3 cannot be computed in bounded depth with polynomial size.*

We will work through the proof of Razborov's Theorem next time.