CS 880: Advanced Complexity Theory 1/23/2008

Lecture 1: Introduction

Instructor: Dieter van Melkebeek Scribe: Jeff Kinne

Welcome to computer science 880. This is a course which treats advanced topics in complexity theory, with the particular topic covered different for each offering of the course. For the current instantiation of the course, we will focus on harmonic analysis of Boolean functions and some of its many applications. If extra time permits, we may spend some time on other topics – such as quantum computing – at the end of the semester. Harmonic analysis enjoys a wide variety of applications in part because Boolean functions can be viewed as many different types of objects and harmonic analysis has proved fruitful in analyzing the properties of these objects. Today we give a brief description of harmonic analysis, discuss some of its applications, and conclude the lecture by introducing one of the properties of Boolean functions that will prove useful to us – influence of variables.

1 Harmonic Analysis

We will get into the details of what exactly harmonic analysis means in the next lecture. Today we only give a flavor of what is entailed. *Harmonic analysis* first developed to analyze functions of the form $f : \mathbb{R} \to \mathbb{C}$, and in this setting the techniques are typically called *Fourier analysis*. We begin by writing f as a linear combination of basis functions, with the basis functions referred to as *harmonics*. The coefficients of the harmonics in this representation are referred to as the *Fourier coefficients* of f.

It may seem arbitrary to decompose f in this way, but it turns out that many functions are simpler to analyze in this form. The techniques used in the setting of functions from the reals to the complex numbers can be generalized to functions of the form $f : G \to \mathbb{C}$ for certain groups G, in particular for finite abelian groups. We are mainly interested in Boolean functions, so we will typically deal with functions of the form $f : \{0,1\}^n \to \{0,1\}$, where we view the domain as the group $(\{0,1\}^n, +)$ or $\mathbb{Z}_2^n$ with addition.

2 Applications

Harmonic analysis has applications ranging across many areas of mathematics, computer science, and other areas. Here, we list a few, many of which we will see later in this course. Some of these deal specifically with harmonic analysis of real-valued functions; we will not study these in this course.

2.1 Mathematics

Harmonic analysis has enjoyed applications in many areas of mathematics, including the following.

- Differential Equations: Harmonic analysis began as a method to assist in solving diffusion equations. Both the boundary conditions and solution to the system of equations would be decomposed as harmonics, a transformation that can make solving the system simpler.

- Group Theory: As already stated, the techniques originally applied to real-valued functions but have been generalized to other groups.
- Number Theory
- Probability Theory: For example, the distribution of a sum of independent random variables is equal to the convolution of the distributions of the individual random variables. Harmonic analysis is useful here because the Fourier transform of the convolution of two distributions is equal to the point-wise product of the Fourier transform of the original distributions. We will say more about this in the next lecture.
- Combinatorics: The construction of certain explicit expander graphs, such as Ramanujan graphs, uses harmonic analysis.

2.2 Computer Science

There is of course an overlap between computer science and mathematics, with many of the applications listed under mathematics also having applications within computer science. Nonetheless, the following are additional applications of harmonic analysis within computer science.

- Fast Fourier Transform: Used in signal processing and in efficient integer multiplication algorithms.
- Derandomization: Several of the objects used in derandomization are analyzed with harmonic analysis, including the aforementioned expander graphs as well as small-bias sample spaces.
- PCP Theorem: the proof of.
- Hardness of Approximation: related to the PCP theorem.
- Learning Theory

2.3 Other Areas

We mostly focus on the applications of harmonic analysis within mathematics and computer science. One area outside of those we will look at is the following.

- Social Choice Theory: The study is of voting schemes, where we wish to determine a winner of an election given the preferences of each citizen; simple majority is one potential scheme. Arrow's Theorem is an example of a result in this area.

3 Boolean Functions

Harmonic analysis enjoys such a wide range of applications in part because Boolean functions can be viewed as different types of objects, and these different views can be applied in various settings. The following are among the standard ways of viewing a Boolean function $f : \{0,1\}^n \to \{0,1\}$.

- Truth Table: the standard view in circuit complexity.

- Subset $A \subseteq \{0,1\}^n$ under Hamming distance: Recall the Hamming distance between two strings is the number of bit positions on which they differ. This is a more geometrical view, where we can view the function as a metric space. This view has applications within, among other settings, error-correcting codes.

- Concept: This is the default view within learning theory. Each variable represents some feature, with the combined set of features indicating a sample that either belongs to a concept or not. The concept is the set of samples with a certain property. A typical example is that of detecting whether an email is spam or not. Each word of interest is given an indicator variable indicating whether the word is present in the message, and a message is within the concept if it is spam. The goal is to take a relatively small training set of samples – samples for which we are told whether they are spam or not – and produce a hypothesis which predicts whether future messages are spam or not.

- Set System: Each string in the domain is viewed as a subset of the integers $[n] = \{1, 2, 3, ..., n\}$, where a 1 in bit position i indicates that integer i is present in the set. The function is viewed as the collection of subsets of $[n]$ for which the function maps to 1. This view is used in the study of hyper-graphs in combinatorics.

- Graph: Each variable is associated with an edge of a graph indicating whether the edge is present or not. This view is often used in statistical physics.

- Social Choice: The setting is that of an election between two candidates, the "0 candidate" and the "1 candidate". Each variable represents which candidate a particular individual prefers, and the outcome of the function is the outcome of the election.

4 Influence of Variables

From the list of applications given above, we see that harmonic analysis can be used in very diverse settings. As such, it is natural that many of the properties of Boolean functions we will examine are very general properties – so they can be applied in many different areas. We begin today by introducing the first of these properties.

Definition 1. *Let $f : \{0,1\}^n \to \{0,1\}$.*

We define the total influence *of f, also called the* average sensitivity *of f as:*

$$I(f) = \mathrm{E}_x[\#\ \textit{neighbors } y \textit{ of } x \textit{ such that } f(x) \neq f(y)]$$

where y is a neighbor of x if the Hamming distance between the two is 1 – if x and y differ in one bit position.

The influence of the i^{th} variable *is defined as $I_i(f) = \Pr_x[f(x) \neq f(x^{(i)})]$, where $x^{(i)}$ is equal to x but with the i^{th} bit flipped.*

In other words, $I(f)$ is the expected value of the number of positions in x on which f is sensitive in that flipping that bit would flip the value of the function. We begin with a few basic properties of this quantity.

Proposition 1. *Let $f : \{0,1\}^n \to \{0,1\}$. Then*

1. *$0 \leq I(f) \leq n$*
2. *$0 = I(f)$ only for the two constant functions $f(x) \equiv 0$ and $f(x) \equiv 1$.*
3. *$I(f) = n$ only for the parity function and the complement of the parity function.*
4. *$I(f) = \sum_{i=1}^{n} I_i(f)$*

Proof. The first item follows by looking at the definition of $I(f)$: the inside term of the expectation is non-negative and each n-bit string has exactly n neighbors.

The second and third items are left as exercises for the reader.

The fourth item follows by the definition of expectation and rearranging, or by

$$\sum_{i=1}^{n} I_i(f) = \sum_{i=1}^{n} \Pr_x[f(x) \neq f(x^{(i)})] = \sum_{i=1}^{n} \mathrm{E}_x[1_{f(x)\neq f(x^{(i)})}] = \mathrm{E}_x[\sum_{i=1}^{n} 1_{f(x)\neq f(x^{(i)})}] = I(f),$$

where we used the fact that the expected value of an indicator variable is the probability the variable is equal to 1, and the second to last equality is by linearity of expectation. □

To get a further feel for the influence of functions, consider the total influence of the majority function.

Proposition 2. *$I(Maj) = \Theta(\sqrt{n})$.*

Proof. Notice that for most strings, there are no bits that would change the majority value just by flipping that bit. We only need to look at strings that have exactly half 1's (or one fewer or greater than half 1's, depending on how majority is defined and whether n is even or odd – we ignore those issues here). For these strings, half of the bits would change the outcome by flipping the bit. So we need to determine the relative fraction of strings that have half 1's. Assuming n is even, this is exactly $\frac{\binom{n}{n/2}}{2^n}$. Using Sterling's formula, or other techniques, this is $\Theta\left(\frac{1}{\sqrt{n}}\right)$. Then $I(Maj) = \frac{1}{2} \cdot \Theta\left(\frac{n}{\sqrt{n}}\right) = \Theta(\sqrt{n})$.

We point out that we also could have arrived at this conclusion by considering $I_i(Maj)$ and the fact that $I(Maj) = \sum_{i=1}^{n} I_i(Maj)$. □

We point out that a random function has expected total influence half of the maximum.

Proposition 3. *$E_f[I(f)] = \frac{n}{2}$.*

Proof. Exercise for the reader. □

Total influence can play the role of an indicator of the complexity of a function – with small total influence indicating a function is uncomplicated in some sense, and large influence indicating a more complicated function. Influence is only an indication of complexity; e.g. parity has maximal influence but is a relatively simple function.

If a function favors either the 0 or 1 output, this can sometimes artificially effect some of its properties, and for this reason we often only consider balanced functions – functions f for which $\Pr_x[f(x) = 1] = \frac{1}{2}$. For these, we get a slightly better lower bound on the total influence.

Proposition 4. *Let f be a balanced function. Then $I(f) \geq 1$. Further, $I(f) = 1$ only for functions of the form $f(x) = x_i$ or $f(x) = \overline{x_i}$.*

Proof. Exercise for the reader. □

Functions that assign the output based on a single input variable are called *dictator functions*, for the meaning such a function gives in the social choice setting. These can be generalized to functions where a small group of bits determines the outcome.

Definition 2. *A function f is an r-*junta *if f depends on no more than r of its variables.*

We get the following trivial observation about r-junta functions.

Proposition 5. *For f any r-junta function, $I(f) \leq r$.*

In words, a function that depends on few variables has small total influence. We can ask ourselves whether the converse is true: does every function with small influence depend on few variables?

This direction does not hold, at least not in a strong sense. The following example exhibits a function that depends on all its variables but has only $O(\log n)$ total influence.

Definition 3. *The tribes function is defined as:*

$$Tribes(n) = \bigvee_{j=1}^{2^s} \bigwedge_{k=1}^{s} x_{j,k}.$$

That is, there are 2^s many groups ("tribes"), and the output of the function is 1 exactly when there is at least one tribe that is unanimously equal to 1. Notice that $n = s \cdot 2^s$.

It is immediate that the tribes function depends on each of its variables. For example, consider variable $x_{j,k}$. If we set $x_{j,k'} = 1$ for all $k' \neq k$ and set all remaining bits to 0, then the output of the function is determined by the value of $x_{j,k}$. Further, the tribes function as we have defined it is close to being balanced, as follows. There is exactly one choice of bits for a given group that results in a unanimous choice of 1, so with probability $(1 - \frac{1}{2^s})$ the group does not do so. Further, the tribes function outputs 0 if each of 2^s independent such events occur, meaning $\Pr_x[Tribes(x) = 0] = (1 - \frac{1}{2^s})^{2^s} \approx \frac{1}{e}$. Note that we could make the Tribes function as close to balanced as we like by tweaking the relation between the size of the tribes and the number of tribes.

Consider the total influence of the tribes function.

Proposition 6. *$I(Tribes) \leq O(\log n)$.*

Proof. We first show that each individual bit has small influence. The fraction of inputs on which bit i determines the value of the function is upper bounded by the probability that the remaining bits from that bit's group are all equal to 1, so $I_i(Tribes) \leq \frac{1}{2^{s-1}}$ for all i. As there are $s \cdot 2^s$ bits, we conclude that $I(Tribes) \leq \frac{s \cdot 2^s}{2^{s-1}} = O(\log n)$, where the last equality follows because $n = s \cdot 2^s$. □

So we have a function that depends on all n of its bits but still $I(f) = O(\log n)$, so there can be an exponential gap between the number of bits a function depends on and its total influence. In fact, this is the worst-case scenario for the separation between these two values, as given by the following theorem.

Theorem 1. *Let f be a Boolean function. Then for every $\epsilon > 0$, f is ϵ-close to a $2^{O(I(f)/\epsilon)}$-junta.*

By ϵ-close we refer to the relative Hamming distance – that the fraction of positions on which f differs from such a function is at most ϵ. We will prove this theorem and see certain applications of it later.

4.1 Applications of Influence

Here we give a few applications of influence.

Circuit Complexity

The following theorem is an application of the idea that influence is a measure in a sense of complexity. The theorem states that a function that is simple in terms of circuit complexity has small influence. We will prove this theorem later in the course.

Theorem 2. *Let f be a function that has a depth d unbounded fan-in circuits of size s. Then $I(f) \leq O(\log^{d-1}(s))$.*

Corollary 1. *Let f be a function in AC^0. Then $I(f) \leq polylog(n)$.*

Notice that along with the total influence of parity and majority given earlier in this lecture, we get as an immediate corollary that neither of these functions is contained in AC^0.

Learning Theory

Influence can be used to develop learning algorithms under the uniform distribution for functions with small influence. The intuition is that functions with small influence have some sort of long range correlations – that even by flipping many bits, the function likely will not change values. This suggests a learning algorithm where we ask a number of random samples until we have samples within small Hamming distance of all possible strings; for a given query, we then take a weighted average of the values on our sample set, where samples that are closer to the new query are given higher weight. It can be shown that the number of random samples that suffices for this algorithm to do well is at most $n^{O(I(f))}$. Combined with the above results on circuit complexity, we conclude that we can learn functions in AC^0 with a quasi-polynomial number of random samples.

The analysis uses as a key component that a function with small influence has most of the weight of its Fourier transform concentrated on a small known set of coefficients. This fact suggests a generalization – that any function with the Fourier transform concentrated on a small number of coefficients can be learned. Such a result is obtained by attempting to discover the Fourier coefficients with large weight. The latter can be accomplished using a list-decoding procedure for the Hadamard code, due to Goldreich and Levin. We conclude that, for example, we can learn small DNF's with $n^{O(\log \log n)}$ random samples and membership queries. We will cover these results later in this course. Additional techniques that we will not cover have shown that DNF's can be learned with a polynomial number of such samples and queries.

5 Next Time

Next time we will cover three main topics: i) we will begin by looking at other applications of influence; ii) we will then conclude our introduction to the course by looking at two other structural properties of languages – noise sensitivity and closeness to linear functions; iii) after finishing this introductory material, we will begin to develop the setting of harmonic analysis.