Boolean Functions

Already met.....sort of....

- Already seen that a simplicial complex (e.g., independence complex of a graph) defines a ``lower part'' of the Boolean cube (vertices that are associated to faces of the simplicial complex) and an upper part that is not part of the simplicial complex…
- Think of assigning 0 to vertices of the Boolean cube in the simplicial complex and 1 to vertices outside…then we have a function from the Boolean cube to {0,1}. (It happens to be a monotonic boolean function – MBF – see later).
- Boolean functions are "everywhere" – in a sense everything a computer does is evaluating Boolean functions because all computer data and programs are bits and the output is also a string of bits…

References….

There is a free textbook…

http://www.cs.cmu.edu/~odonnell/papers/Analysis-of-Boolean-Functions-by-Ryan-ODonnell.pdf

But we will be looking at only a small part of this…

Some parts of these slides are extracted from

http://www.cs.cmu.edu/~odonnell/papers/barbados-aobf-lecture-notes.pdf

$\{0,1\}^n \rightarrow \{0,1\}$ or $\{1,-1\}^n \rightarrow \{1,-1\}$

- The two symbols/values used in Boolean logic are in some sense arbitrary. T and F can be (and are) used. 0 and 1 can be convenient especially when "what you do in the calculations" aligns with arithmetic modulo 2. 1 and -1 can be convenient when you use arithmetic of multiplication and taking signs…parity (-1*-1….-1 is 1 if an even number of -1's – even parity, else -1 is odd parity).
- $\{1,-1\}^n \rightarrow \{1,-1\}$ aligns well with viewing a Boolean function as a multilinear polynomial (a type of Fourier expansion) and is favoured in the O'Donnell book and other works.

Example – Majority function

- $MAJ_3(x)$ is the function that takes a 3 bit input and returns the result that is the majority of the bits (viewed as votes).
- Let $x=(x_1,x_2,x_3)$
- What is $MAJ_3(x)$ when
  - x=(-1,-1,-1)?
  - x=(-1,1,-1)?
  - x=(-1,1,-1)?
  - x=(1,1,-1)?

Multilinear expressions

$MAJ_3(x) = sgn(x_1 + x_2 + x_3)$

But this isn't a polynomial….

You can check that

$MAJ_3(x) = 0.5x_1+0.5x_2+0.5x_3-0.5x_1x_2x_3$

(naively you need to check 8 possible values of the input – but you can be a little more clever by seeing symmetry)

The RHS is a multilinear polynomial – no powers exist above 1…so linear in each variable…

Note: of course one can evaluate the polynomial at values other than -1,1 – and then you will get a function from $R^3 \rightarrow R$ (that agrees with MAJ_3 at Boolean vertices).

Fourier Expansion for (functions over) the Boolean Cube

Theorem 1 (Fourier expansion) *Every $f : \{-1,1\}^n \to \mathbb{R}$ can be uniquely expressed as a multilinear polynomial $\mathbb{R}$,*

$$f(x) = \sum_{S \subseteq [n]} c_S \prod_{i \in S} x_i, \quad \text{where each } c_S \in \mathbb{R}.$$

We will write $\hat{f}(S)$ to denote the coefficient c_S and $\chi_S(x)$ for the function $\prod_{i \in S} x_i$, and call $f(x) = \sum_{S \subseteq [n]} \hat{f}(S)\chi_S(x)$ the Fourier expansion of f. We adopt the convention that $\chi_\emptyset \equiv 1$, the identically 1 function. We will write $\deg(f)$ to denote $\max_{S \subseteq [n]}\{|S| : \hat{f}(S) \neq 0\}$, and call this quantity the Fourier degree of f.

[n] means all the subsets of 1..n – so the sum loops over all vertices of the Boolean cube interpreting that vertex as a subset S of the 1…n.

Fourier Expansion for (functions over) the Boolean Cube

We will sometimes refer to $\chi_S(x) : \{-1,1\}^n \to \{-1,1\}$ as the "parity-on-S" function, since it takes value 1 if there are an even number of -1 coordinates in x and -1 otherwise. Using the notation of Theorem 1, we have that $\widehat{\mathsf{MAJ}_3}(\{1\}) = \frac{1}{2}$, $\widehat{\mathsf{MAJ}_3}(\{1,2,3\}) = -\frac{1}{2}$, $\widehat{\mathsf{MAJ}_3}(\{1,2\}) = 0$, and $\deg(\mathsf{MAJ}_3) = 3$.

How to get these "Fourier Coefficients"? Well it follows the pattern of more familiar Fourier expansion. They are the inner product between the function you are transforming and the corresponding parity function (you are finding the coefficient of)…see O'Donnell.

Boolean Functions as Voting Schemes

We may think of a boolean function $f : \{-1,1\}^n \to \{-1,1\}$ as a voting scheme for an election with 2 candidates (± 1) and n voters $(x_1, \ldots, x_n)$. Many boolean functions are named after the voting schemes they correspond to: the i-th dictator $\mathsf{DICT}_i(x) = x_i$ (*i.e.* $\mathsf{DICT}_i \equiv \chi_i$); k-juntas (functions that depend only on k of its n variables, where we think of k as $\ll n$, or even a constant); the majority function $\mathsf{MAJ}(x) = \text{sgn}(x_1 + \ldots + x_n)$. The majority function is special instance of linear threshold functions $f(x) = \text{sgn}(a_0 + a_1 x_1 + \ldots + a_n x_n)$, $a_i \in \mathbb{R}$, also known as weighted-majority functions, or halfspaces. Another important voting scheme in boolean function analysis is $\mathsf{TRIBES}_{w,s} : \{-1,1\}^{ws} \to \{-1,1\}$, the s-way OR of w-way AND's of disjoint sets of variables (where we think of -1 as true and 1 as false). In $\mathsf{TRIBES}_{w,s}$, the candidate -1 is elected iff at least one member of each of the s disjoint tribes of w members votes for -1.

Properties of Voting Schemes…

The following are a few reasonable properties one may expect of a voting scheme:

- Monotone: if $x_i \le y_i$ for all $i \in [n]$ then $f(x) \le f(y)$.
- Symmetric: $f(\pi(x)) = f(x)$ for all permutations $\pi \in S_n$ and $x \in \{-1,1\}^n$.
- Transitive-symmetric (weaker than symmetric): for all $i, j \in [n]$ there exists a permutation $\pi \in S_n$ such that $\pi(i) = j$ and $f(x) = f(\pi(x))$ for all $x \in \{-1,1\}^n$.

The monotone property is natural because it says that if a candidate wins with some set of votes, then they should also win when they get more votes! Increasing your votes shouldn't turn a winning position into a losing position!
The symmetry condition is just saying how you order the voters in your listing of voters shouldn't count.

Influences of variables (voters) in a Boolean Function

Definition 22 (influence) *Let* $f : \{-1,1\}^n \to \{-1,1\}$*. We say that variable* $i \in [n]$ *is pivotal for* $x \in \{-1,1\}^n$ *if* $f(x) \neq f(x^{\oplus i})$*, where* $x^{\oplus i}$ *is the string* x *with its* i*-th bit flipped. The influence of variable* i *on* f*, denoted* $\mathrm{Inf}_i(f)$*, is the fraction of inputs for which* i *is pivotal. That is,* $\mathrm{Inf}_i(f) := \mathbf{Pr}[f(x) \neq f(x^{\oplus i})]$.

Note: when you “flip a bit” in the Boolean cube, you move along an edge of the Boolean cube *in that direction*. For
And n-dimensional Boolean cube there are 2^{n-1} edges in each direction. So… the above says that Influence of variable is is the count of edges, in the direction i, where the Boolean function changes value from one end of the edge to the other, and then normalizing that count by dividing by 2^{n-1} . In other words, the influence if variable i, is the probability that if we pick an edge of the Boolean cube *in direction i* (uniformly at random), that we would see the function change value over that chosen edge.

Some example influences…

- It should be obvious that the influences of Dict(x) is 1 if the variable is the dictator variable, 0 otherwise…

$$\mathrm{Inf}_i(\mathsf{MAJ}) = \binom{n-1}{(n-1)/2} \cdot 2^{-(n-1)}$$

(a voter will be pivotal iff all other voters are evenly split in their votes)

Exercise – check these formulae/facts for small number of variables/voters by drawing the Boolean cube and finding the edges where the function changes value….

Influences when the function is monotone

Proposition 25 (influence of monotone functions) *Let* $f : \{-1,1\}^n \to \{-1,1\}$ *be a monotone function. Then* $\mathrm{Inf}_i(f) = \hat{f}(i)$.

For *monotone Boolean functions* the influences are just the 1st-order Fourier coefficients

Total Influence

Definition 26 (total influence) *Let* $f : \{-1,1\}^n \to \mathbb{R}$. *The total influence of* f *is* $\mathrm{Inf}(f) := \sum_{i=1}^n \mathrm{Inf}_i(f)$.

If f is a boolean function, then

$$\mathrm{Inf}(f) = \sum_{i=1}^n \mathbf{Pr}[f(x) \neq f(x^{\oplus i})] = \sum_{i=1}^n \mathbf{E}[\mathbf{1}(f(x) \neq f(x^{\oplus i}))] = \mathbf{E}\Big[\sum_{i=1}^n \mathbf{1}(f(x) \neq f(x^{\oplus i}))\Big].$$

Since each influence is a value in [0,1], total influence is a value in [0,n]

Proposition 27 (MAJ maximizes sum of linear coefficients) *Let* n *be odd. Among all boolean functions* $f : \{-1,1\}^n \to \{-1,1\}$, *the quantity* $\sum_{i=1}^n \hat{f}(i)$ *is maximized by* $\mathsf{MAJ}(x) = \mathrm{sgn}(x_1 + \ldots + x_n)$. *Consequently, if* f *is monotone then* $\mathrm{Inf}(f) \leq \mathrm{Inf}(\mathsf{MAJ}) \sim \sqrt{2n/\pi}$.

Read on……

- The above is all we will need for what I want to cover in this course.
- BUT I encourage you to look at the references (or web search) for more on this topic. Voting schemes and weighted aggregation of “scores” are how we “decide” a lot of things – from social decisions (group of friends with different opinions on which restaurant to visit, what score/grade you will get for your course – given your different scores on assigments/exams etc., decisions made by artificial intelligence (a neuron is something that more or less just takes a weighted sum of its inpute and makes a decision – whether to turn on…) etc.
- Applies to ANY Boolean function – hence computer computation – whether you naturally think of that as a voting procedure or not…