On decision trees, influences, and learning monotone decision trees

Ryan O'Donnell*
School of Mathematics
Institute for Advanced Study
Princeton, NJ
odonnell@ias.edu

Rocco Servedio
Dept. of Computer Science
Columbia University
New York, NY
rocco@cs.columbia.edu

May 26, 2004

Abstract

In this note we prove that a monotone boolean function computable by a decision tree of size s has average sensitivity at most $\sqrt{\log_2 s}$. As a consequence we show that monotone functions are learnable to constant accuracy under the uniform distribution in time polynomial in their decision tree size.

1 Decision trees

Let $f\colon \{-1,1\}^n \to \{-1,1\}$ be a boolean function.

Fourier notions: Throughout this paper we view $\{-1,1\}^n$ as a probability space under the uniform distribution. Recall f's *Fourier expansion*,

$$f(x) = \sum_{S\subseteq[n]} \hat{f}(S)\chi_S(x),$$

where $\chi_S(x) = \prod_{i\in S} x_i$ and $\hat{f}(S) = \mathbf{E}_x[f(x)\chi_S(x)]$. We also recall the notions of influence and average sensitivity: The *influence of i on f* is $\mathrm{Inf}_i(f) = \Pr_x[f(x) \neq f(x^{(i)})]$, where $x^{(i)}$ denotes x with the ith bit flipped; if f is a monotone function then $\mathrm{Inf}_i(f) = \hat{f}(\{i\})$. We shall henceforth write $\hat{f}(i)$ in place of $\hat{f}(\{i\})$. The *average sensitivity of f* is $\mathrm{I}(f) = \sum_{i=1}^n \mathrm{Inf}_i(f)$.

Decision trees: Suppose we have a decision tree computing $f\colon \{-1,1\}^n \to \{-1,1\}$; we will always assume (without loss of generality) that no variable appears more than once on any path of the tree. Note that picking a uniformly random input $x \in \{-1,1\}^n$ is equivalent to the following two-step procedure: First, pick a uniformly random path P in the tree by starting at the root and assigning to the variables encountered uniformly at random until a leaf is

*This material is based upon work supported by the National Science Foundation under agreement No. CCR-0324906. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

reached. Second, assign uniformly at random to those variables as yet unset. Corresponding to the first step of this process we define a collection of random variables $P_1, \ldots, P_n$ as follows:

$$P_i = \begin{cases} 1 & \text{if the variable } i \text{ is encountered on the random path and } x_i \text{ is chosen to be } 1, \\ -1 & \text{if the variable } i \text{ is encountered on the random path and } x_i \text{ is chosen to be } -1, \\ 0 & \text{if the variable } i \text{ is not encountered on the random path.} \end{cases}$$

For each i we have $\mathbf{E}[P_i] = 0$; a slight amount of reflection reveals that also $\mathbf{E}[P_i \mid P_j] = 0$ for all $i \neq j$. Hence while P_i and P_j are not independent we do have $\mathbf{E}[P_iP_j] = 0$ for all $i \neq j$. We write ΣP for $\sum_{i=1}^n P_i$, the sum of the bit assignments made along the random path P, and we also write $\text{len}(P)$ for the length of the random path P; another way of expressing $\text{len}(P)$ is $\sum_{i=1}^n (P_i)^2$.

Consider the two-step procedure for choosing $x \in \{-1,1\}^n$ at random: first choose P at random, assigning randomly to the variables on the path; then choose the remaining unset variables uniformly at random. Since the value of $f(x)$ is fixed after the first step in the procedure, we may denote this value by $f(P)$.

Proposition 1 *Let $f\colon \{-1,1\}^n \to \{-1,1\}$ be computed by a decision tree with paths P. Then*

$$\sum_{i=1}^n \hat{f}(i) = \mathop{\mathbf{E}}_P[f(P) \cdot \Sigma P].$$

Proof:

$$\begin{aligned} \sum_{i=1}^n \hat{f}(i) &= \sum_{i=1}^n \mathop{\mathbf{E}}_{x\in\{-1,1\}^n} [f(x)x_i] \\ &= \mathop{\mathbf{E}}_{x\in\{-1,1\}^n} \left[f(x) \sum_{i=1}^n x_i\right] \\ &= \mathop{\mathbf{E}}_{P;\ x_j:\ P_j=0} \left[f(P) \left(\sum_{i:\ P_i\neq 0} x_i + \sum_{j:\ P_j=0} x_j \right)\right] \\ &= \mathop{\mathbf{E}}_P \left[f(P)\left(\Sigma P + \mathop{\mathbf{E}}_{x_j:\ P_j=0}\left[\sum_{j:\ P_j=0} x_j\right]\right)\right] \\ &= \mathop{\mathbf{E}}_P[f(P) \cdot \Sigma P]. \end{aligned}$$

The final equality holds since $\mathbf{E}[x_j | P_j = 0] = 0$ for each j. □

Theorem 1 *Let $f\colon \{-1,1\}^n \to \{-1,1\}$.*

1. *If f is computed by a decision tree of size[1] s then $\sum_{i=1}^n \hat{f}(i) \le \sqrt{\log_2 s}$.*
2. *If f is computed by a decision tree of depth d then $\sum_{i=1}^n \hat{f}(i) \le \mathrm{I}(\mathsf{Maj}_d) \sim \sqrt{\frac{2}{\pi}}\sqrt{d} \le \sqrt{d}$.*

If f is monotone we can replace $\sum_{i=1}^n \hat{f}(i)$ by $\mathrm{I}(f)$.

Proof: Since f is ± 1-valued, from the Proposition it is clear that

$$\sum_{i=1}^n \hat{f}(i) \le \mathop{\mathbf{E}}_P[|\Sigma P|]$$

[1]Number of leaves.

with equality iff f computes the majority of the bits along each of its decision tree paths — i.e., $\mathrm{sgn}(\Sigma P)$ (f may output anything if the bits split evenly.)

In case (1), we proceed as follows:

$$\mathop{\mathbf{E}}_P[|\Sigma P|] \leq \sqrt{\mathop{\mathbf{E}}_P[|\Sigma P|^2]} = \sqrt{\mathop{\mathbf{E}}_P\Big[\sum_{i,j=1}^n P_i P_j\Big]} = \sqrt{\mathop{\mathbf{E}}_P[\mathrm{len}(P)] + \sum_{i\neq j}\mathop{\mathbf{E}}_P[P_i P_j]} = \sqrt{\mathop{\mathbf{E}}_P[\mathrm{len}(P)]}.$$

(At this point we have proved the upper bound of $\sqrt{d}$ in case (2).) It remains to show that $\mathbf{E}_P[\mathrm{len}(P)] \leq \log_2 s$; we use induction on s. The result is obvious when $s = 2$; for larger s, suppose we have a size-s tree in which the left subtree of the root has size s_1 and the right subtree of the root has size s_2, with $s = s_1 + s_2$. The expected length of a random path in such a tree is 1 plus half the expected length in the left subtree plus half the expected length in the right subtree. By induction this is at most $1 + \frac{1}{2}\log_2 s_1 + \frac{1}{2}\log_2 s_2 = \log_2(2\sqrt{s_1 s_2}) \leq \log_2(s_1 + s_2) = \log_2 s$ where we have used the AM-GM inequality.

In case (2), we instead note in upper-bounding $\mathbf{E}_P[|\Sigma P|]$ it doesn't hurt to assume that the tree is a full depth-d tree; this is because if we have a path of depth less than d, we can extend it redundantly, querying an irrelevant variable — if ΣP for the path was nonzero then $\mathbf{E}_P[|\Sigma P|]$ is unchanged, if ΣP for the path was zero then $\mathbf{E}_P[|\Sigma P|]$ will increase. Now note that $\mathbf{E}_P[|\Sigma P|]$ does not depend on the names of the variables labeling the nodes of the tree; hence we may assume that all nodes at level ℓ read x_ℓ, for $\ell = 1 \dots d$. But now equality in $\sum_{i=1}^n \hat{f}(i) \leq \mathbf{E}_P[|\Sigma P|]$ occurs if the decision tree computes Maj_d, as claimed. The asymptotic formula $\mathrm{I}(\mathsf{Maj}_d) = (1 + o(1))\sqrt{\frac{2}{\pi}}\sqrt{d}$ is well known. □

Remarks:

1. In the case of monotone functions with depth-d decision trees, Theorem 1 generalizes the well-known edge-isoperimetric inequality on the discrete n-cube, which says that for monotone $f\colon \{-1,1\}^n \to \{-1,1\}$, $\mathrm{I}(f) \leq \mathrm{I}(\mathsf{Maj}_n)$.
2. We conjecture that if $s = s(d)$ is the minimal size of a decision tree computing Maj_d, then every function f computable by a decision tree of size s has $\sum_{i=1}^n \hat{f}(i) \leq \sum_{i=1}^n \widehat{\mathsf{Maj}_d}(i)$.

2 Learning monotone decision trees

The following result is immediate from inspecting the proof of Friedgut's '98 theorem about functions with low average sensitivity [Fri98]:

Theorem 2 *Let $f\colon \{-1,1\}^n \to \{-1,1\}$, $\epsilon > 0$, $t = 2\mathrm{I}(f)/\epsilon$, and $J = \{i : \mathrm{Inf}_i(f) \geq t3^{-t}\}$. Then $|J| \leq 3^t$, and furthermore*

$$\sum_{S\colon\, S\subseteq J,\ |S|\leq t} \hat{f}(S)^2 \geq 1 - \epsilon.$$

Combining Theorems 1 and 2 with the idea behind the monotone DNF learning algorithm of [Ser01], we get the following uniform distribution algorithm for learning monotone functions in time polynomial in their decision tree size:

Theorem 3 *The class of monotone functions can be learned under the uniform distribution to accuracy ϵ (with confidence $1 - \delta$) in time $s^{O(1/\epsilon^2)} \cdot \mathrm{poly}(n) \cdot \log(1/\delta)$, where s represents decision tree size.*

Proof: Let f be the unknown monotone function to be learned. We may assume that the algorithm knows s, the decision tree size of f, by a standard doubling argument. From Theorem 1 we know that $\mathrm{I}(f) \leq \sqrt{\log_2 s}$; let $t = 2\sqrt{\log_2 s}/\epsilon$. Since f is monotone, its influences are equal to its degree-one Fourier coefficients and thus can be accurately estimated from uniformly random samples. The algorithm first determines the set $J = \{i : \mathrm{Inf}_i(f) \geq t3^{-t}\}$. It then estimates all Fourier coefficients $\hat{f}(S)$ such that $|S| \leq t$ and $S \subseteq J$. By Theorem 2 this gives the algorithm all but ϵ of f's spectrum; it is well known that this is sufficient for learning f to accuracy ϵ. To conclude we note that up to the $\mathrm{poly}(n) \cdot \log(1/\delta)$ the running time is dominated by the number of Fourier coefficients estimated, which is at most $|J|^t = 3^{t^2} = s^{O(1/\epsilon^2)}$. $\square$

References

[Fri98] E. Friedgut. Boolean functions with low average sensitivity depend on few coordinates. *Combinatorica*, 18(1):474–483, 1998.

[Ser01] R. Servedio. On learning monotone DNF under product distributions. *14th Ann. Conference on Comp. Learning Theory*, 558–573, 2001.