6.895 Randomness and Computation	March 10, 2008
Lecture 10	
Lecturer: Ronitt Rubinfeld	*Scribe: Alex Cornejo*

Last Lecture

Definition 1 (L1 Norm) *Let $f : \{\pm 1\}^n \to \mathbb{R}$, then $L_1(f) = \sum_S \left|\hat{f}(S)\right|$.*

Claim 2 *Given ε, $S_\varepsilon = \left\{S \subseteq [n] : \left|\hat{f}(s)\right| = \frac{\varepsilon}{L_1(f)}\right\}$, we have*

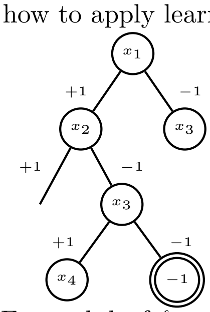
1. $|S_\varepsilon| \leq \frac{(L_1(f))^2}{\varepsilon}$
2. $\sum_{S \in S_\varepsilon} \hat{f}(S)^2 \geq 1 - \varepsilon$

Theorem 3 *Boolean functions can be learned to ε-accuracy with* $\mathrm{poly}(n, L_1(f), 1/\varepsilon)$ *queries under uniform distribution.*

Definition 4 (Monotone functions) *We assume the following ordering of vectors in $\{\pm 1\}^n$: $x \leq y$ if and only if for all coordinates i, $x_i \leq y_i$. A function f is* monotone *if $\forall x \leq y$, $f(x) \leq f(y)$.*

1 Learning Decision Trees

We show how to apply learning with ε-accuracy using $\mathrm{poly}(n, L_1(f), 1/\varepsilon)$ queries to decision trees.



Theorem 5 *If f has a size t decision tree then $L_1(f) \leq t$, where t is the number of nodes in the tree.*

Notice that Theorem 5 together with Theorem 3 imply that we can learn decision trees with $\mathrm{poly}(n, t, 1/\varepsilon)$ queries.

Proof For each leaf ℓ, we define the following function

$$g_\ell(x) = \begin{cases} 1 & \text{if } x \text{ reaches } \ell, \\ 0 & \text{otherwise.} \end{cases}$$

W.l.o.g. the variables on the path to ℓ are $x_1, \ldots, x_k$ and they always take the "-1" direction. Then

$$g_\ell(x) = \left(\frac{1 - x_1}{2}\right)\left(\frac{1 - x_2}{2}\right) \cdots \left(\frac{1 - x_k}{2}\right) = \sum_{S \subseteq [k]} \frac{(-1)^{|S|}}{2^k} \chi_S$$

Notice that $L_1(g_\ell) = \sum_{S \subseteq [k]} \frac{1}{2^k} = 1$.

We now proceed to define f in terms of g.

$$f(x) = \sum_{\text{paths } \ell} g_\ell(x) \cdot \underbrace{\begin{pmatrix} \text{output of leaf} \\ \text{at end of } \ell \end{pmatrix}}_{\pm 1}$$

$$\hat{f}(S) = \sum_{\text{paths } \ell} \hat{g}_\ell(S) \cdot \underbrace{\begin{pmatrix} \text{output of leaf} \\ \text{at end of } \ell \end{pmatrix}}_{\pm 1}$$

Finally, we evaluate the L_1 norm of f.

$$\begin{aligned} L_1(f) &= \sum_S \left|\hat{f}(S)\right| \\ &= \sum_S \left| \sum_{\text{paths } \ell} \pm \hat{g}_\ell(S) \right| \\ &\leq \sum_{\text{paths } \ell} \underbrace{\sum_S |\hat{g}_\ell(S)|}_{L_1(g_\ell)=1} \\ &= \text{number of paths} \\ &\leq t \end{aligned}$$

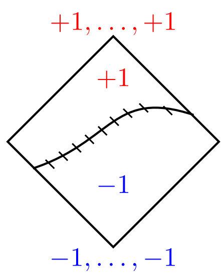
$\blacksquare$

2 Learning monotone functions

Comment: *You can improve on the algorithm we are about to describe by restricting the set of our potential hypothesis g to ± 1 and majority functions (instead of dictators). Instead of $\Omega(\frac{1}{\sqrt{n}})$ advantage, this would give $\Omega(\frac{1}{\sqrt{n}})$ advantage. It is also possible to remove the queries using the low degree algorithm and sampling on the order of $2^{\sqrt{n}}$.*

Throughout the following we assume that we want to learn a function with respect to the uniform distribution. We also assume access to queries. Furthermore, we call each pair $(x_1, \ldots, x_{k-1}, -1, x_{k+1}, \ldots, x_n)$ and $(x_1, \ldots, x_{k-1}, +1, x_{k+1}, \ldots, x_n)$ an *edge* in the hypercube $\{\pm 1\}^n$.

Theorem 6 *For each monotone function $f : \{\pm 1\}^n \to \{\pm 1\}$, there exists a function $g \in \{\pm 1, x_1, \ldots, x_n\}$ such that $\Pr_x[f(x) = g(x)] \geq \frac{1}{2} + \Omega(\frac{1}{n})$.*



This figure represents a Boolean hypercube.
2^n nodes.
2^{n-1} edges in direction i
$n2^{n-1}$ total edges.
A *cut edge* connects a red node to a blue node.

Definition 7 (Influence of the i^{th} variable)

$$\text{Inf}_i(f) = \underbrace{\hat{f}(\{i\})}_{\textit{Homework 2}} = \underbrace{2\Pr[f(x) \neq x_i] - 1}_{\textit{a previous lecture}}$$

$$\text{Inf}_i(f) = \frac{\textit{\# of cut edges in } i^{th} \textit{ direction}}{2^{n-1}}$$

Definition 8 (Total influence)

$$
\begin{aligned}
\mathrm{Inf}(f) &= \sum_{i=1}^{n} \mathrm{Inf}_i(f) \\
&= \frac{\#\ \textit{of cut edges}}{2^{n-1}}
\end{aligned}
$$

Plan of attack. To show that $\mathrm{Inf}_i(f)$ is $\Omega(1/n)$, we will first define the concept of a canonical path and use it to prove a lower bound.

Definition 9 (Canonical path) *For all (x, y) such that x is red and y is blue, a* canonical path *from x to y scans bits from left to right, flipping bits where needed. Each flip corresponds to a step in the path.*

$x =$	-1	+1	+1	+1	+1
	-1	**-1**	+1	+1	+1
	-1	-1	**-1**	+1	+1
$y =$	-1	-1	-1	+1	-1

Observation 10 *It is clear that since the start of a canonical path is red and the end is blue, then there exists at least one edge (u, v) in the path such that u is red and v is blue.*

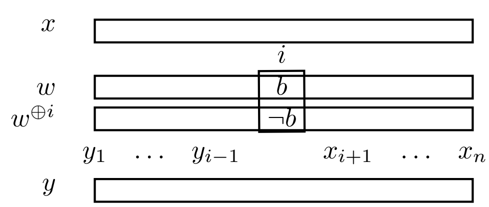
We can assume that $\Pr[f(x) = 1] \in \left[\frac{1}{4}, \frac{3}{4}\right]$ since otherwise we could use one of the constant ± 1 functions to approximate f. Under this assumption, how many red-blue (x, y) pairs can we expect?

$$
\geq \left(\frac{1}{4} 2^n\right)^2 = \frac{1}{16} 2^{2n}
$$

Lemma 11 *For any given edge, there are $\leq 2^n$ canonical paths which cross it.*

Proof Consider an edge $(w, w^{\oplus i})$, a part of a canonical path from x to y. Notice that w and $w^{\oplus i}$ have Hamming distance one and therefore only differ in one bit (the i^{th} bit).

We argue that due to the definition of canonical paths, there are a limited number of paths that can share an edge.



For any canonical path between (x', y') that crosses the edge $(w, w^{\oplus i})$, the prefix $y'_1 \ldots y'_{i-1}$ of y' has to be the same as the prefix of w. This gives us at most 2^{n-i} choices for the last $n-1$ bits of y'. Analogously, the suffix $x'_{i+1} \ldots x'_n$ of x' has to be the same as the suffix of w, and we have at most 2^{i-1} choices for the first bits of x'. Therefore there are $\leq 2^n$ settings of x' and y' consistent with the edge. ■

Since we know that each canonical path has at least one red-blue edge, using lemma 11, we can now give a lower bound on the number of red-blue edges.

$$
\#\text{ of red-blue edges} \geq \frac{\frac{1}{16} 2^{2n}}{2^n} = \frac{1}{16} 2^n
$$

Therefore by the pigeon-hole principle, there is i such that $\geq \frac{1}{16n}2^n$ red-blue edges exist in direction i. Finally, using the definition for the influence of a variable,

$$\mathrm{Inf}_i(f) \geq \frac{\frac{1}{16n}2^n}{2^{n-1}} = \frac{1}{8n}.$$

Since $\Pr[f(x) \neq x_i] = \frac{1}{2} + \mathrm{Inf}_i(f)/2$, we have that

$$\begin{aligned}\Pr[f(x) \neq x_i] &\geq \frac{1}{2} + \frac{1}{16n} \\ &= \frac{1}{2} + \Omega\left(\frac{1}{n}\right).\end{aligned}$$

This completes our proof of Theorem 6.

3 Next Lecture: Boosting PAC Learners

Definition 12 (PAC learning) *An algorithm* $\mathcal{A}$ PAC learns *a concept class* $\mathbf{C}$ *if* $\forall c \in \mathbf{C}$, $\forall$ *distributions* $\mathbf{D}$, $\forall \varepsilon, \delta > 0$, *given examples of* c *using* $\mathbf{D}$, *then* $\mathcal{A}$ *outputs a hypothesis* h *such that with probability* $\geq 1 - \delta$

$$\Pr_{\mathbf{D}}[c(x) \neq h(x)] \leq \varepsilon.$$

Definition 13 (Weak PAC learning) *An algorithm* $\mathcal{WL}$ weakly PAC learns *a concept class* $\mathbf{C}$ *with parameter* τ *if* $\forall c \in \mathbf{C}$, $\forall$ *distributions* $\mathbf{D}$, $\forall \delta > 0$, *given examples of* c *according to distribution* $\mathbf{D}$ *algorithm* $\mathcal{A}$ *outputs a hypothesis* h *such that with probability* $\geq 1 - \delta$

$$\Pr_{\mathbf{D}}[c(x) \neq h(x)] \leq \frac{1}{2} - \tau.$$

The parameter τ *is referred to as the* advantage of the weak learner.

For some years it was thought that these two problems where separate, but Schapire proved that it is possible to boost weak PAC learners to "strong" PAC learners.

Theorem 14 (Shapire) *If* $\mathbf{C}$ *can be weakly learned, then* $\mathbf{C}$ *can be "strongly" learned.*

Notice that that we cannot apply these definitions to boost the algorithm presented in the previous section, since the algorithm we developed relied on the assumption of a uniform distribution, and thus it is not a proper "weak PAC learner".