COMS 6998-2 Advanced Topics in Computational Learning Theory **Spring 2005**

Lecture 1 : January 20, 2005

Lecturer: Rocco A. Servedio *Scribe: Alp Atici*

1 Introduction

Computational learning theory has the following goals:

- Develop automated ways to construct computer programs from data.
- Extract useful information efficiently from raw data.

There are two kinds of learning:

Supervised: Each example consists of a pair (x, y) where x denotes the instance and y its label. The goal is to infer the process being used to label the data.

Unsupervised: No labels/indexing associated to elements. Consider the example of a botanist on a new island.

This course focuses on supervised learning. For conducting supervised learning there are several different points of view:

Statistical/Theoretical: This approach aims to build statistical models for data and provide a refined analysis of generalization of error.

Applied/Machine Learning: This approach focuses on the question "How does this algorithm work on my data?" from a rather practical viewpoint and generally involves sophisticated algorithms.

Computational Learning Theory/Theoretical CS: This perspective, which is adopted by this course, focuses on well-defined, abstract learning scenarios typically involving learning Boolean functions (i.e. the domain $X = \{0, 1\}^n$). Generally, the given function f is assumed to be represented by a nice structure, such as decision trees, DNF formulas or Boolean circuits. The goal is to explain the behaviour of learning algorithms in this abstract context, and the results are presented in a theorem-proof style.

The emphasis of this course will be exploring algorithms for "hard learning problems" and we shall mainly be concerned about the asymptotic behaviour of our algorithms. Therefore the course focuses on efficient extraction of the structural properties of functions but with a flavor of complexity theory.

2 Concept Classes

Reading: "Classes of Boolean Functions" by N. Bshouty (check class webpage for links).

In the basic learning setting, we are interested in the concept class C which is a set of functions of the form $f : \{0,1\}^n \to \{0,1\}$. The learning algorithm is provided with a target concept $c_\circ \in C$ and is expected to determine or approximate the function $c_\circ \in C$ 'efficiently'. Efficiency is most often described in terms of n and $\mathbf{size(c)}$, which is some measure of the size of each concept in $c \in C$. We now describe some classes of boolean functions that we will be concerned with in our learning models.

2.1 The concept class of functions represented by a DNF

Disjunctive **N**ormal **F**orm is the form in which a function is expressed as an OR of ANDs of literals where the literals could be x_i or $\neg x_i$ (x_i denotes the i-th position of an element in the domain $X = \{0,1\}^n$).
Example: $(x_1\bar{x_3}x_4) \vee (x_2x_5x_6) \vee (\bar{x_4}\bar{x_7})$.
The size of a DNF is measured by the number of terms in this expression.

From a combinatorial viewpoint if one considers the elements of $\{0,1\}^n$ as the vertices of an n dimensional cube, then each term of a DNF could be considered as a subcube of $\{0,1\}^n$ corresponding to those $x \in \{0,1\}^n$ which satisfy the term. For instance the term $x_1\bar{x_3}x_4$ in the above DNF could be considered as the subcube $1 * 01 * * \ldots *$. Therefore the positive instances of an s-term DNF corresponds to a union of s subcubes of $\{0,1\}^n$.

Fact 1 *DNFs are a universal representation scheme. In other words any* $f : \{0,1\}^n \to \{0,1\}$ *has an equivalent DNF.*

It is straightforward to see why this fact holds: Note there is a unique term corresponding to each positive instance of a given function f where the term only satisfies that instance and none other. Hence the disjunction of such terms yields f if one considers all its positive instances.

Clearly there could be more than one DNF representation for the same function. Given any $f : \{0,1\}^n \to \{0,1\}$, the DNF-size of f is the number of terms in the smallest DNF that computes f. It is clear that a function $f : \{0,1\}^n \to \{0,1\}$ has DNF size at most 2^n. However a combinatorial argument gives rise to the following stronger fact.

Fact 2 *Any* $f : \{0,1\}^n \to \{0,1\}$ *has DNF size at most* 2^{n-1}.

This bound is also tight because the smallest DNF computing PARITY over n bits has exactly 2^{n-1} many terms. Note each term in the DNF representing PARITY should have length n because flipping any one of the literals should also flip the value of the function computed. Therefore each term in such a DNF could only cover one instance at which PARITY is 1. Hence one needs precisely 2^{n-1} terms to express PARITY in DNF.

A k-DNF is a DNF where each term consists of at most k literals. Therefore any k-DNF has at most $\sum_{i=1}^{k} 2^i \binom{n}{i} \approx n^k$ terms.

2.2 The concept class of functions represented by a decision tree

A **D**ecision **T**ree is a rooted binary tree where each internal node is labeled with one of $x_1, \ldots, x_n$ and has two children. We shall assume the left child always corresponds to an occurrence of a 0. Given a DT it is straightforward to observe the function it computes: given an input one can traverse the tree until one reaches a leaf (which always corresponds to a 0 or 1) based on the conditions each internal node poses. [**Andrew: For example if we are traversing the tree in figure 1 on input** $x = 00110$ **we first go the left child of** x_1 **because** $x_1 = 0$**, then we go to the right child of** x_3 **because** $x_3 = 1$**, and finally output 0 because** $x_5 = 0$.]

The size of a DT is measured by the number of leaves in the DT (note that this is one more than the number of internal nodes). The following fact is also clear.

Fact 3 *DTs are a universal representation scheme. In other words any* $f : \{0,1\}^n \to \{0,1\}$ *has an equivalent DT.*

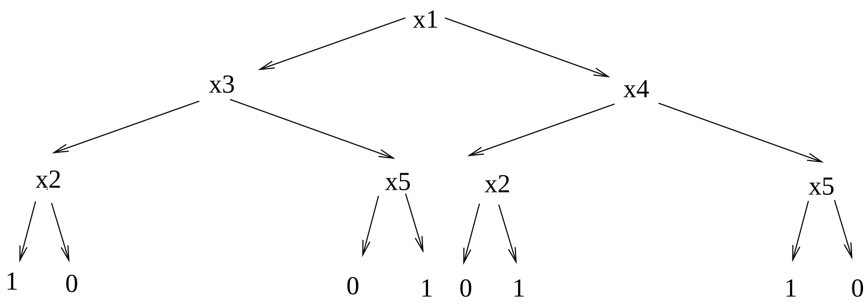

Figure 1: A decision tree

The DT size of $f : \{0,1\}^n \to \{0,1\}$ is the size of the smallest DT representing f. Clearly DT size is at most 2^n since the number of levels could not be more than n. Moreover PARITY over n bits has DT size exactly 2^n.

Note that given a DT each path to a leaf corresponds to a subcube therefore an s-leaf DT gives a partition of $\{0,1\}^n$ into s disjoint monochromatic subcubes.

Lemma 1 *If $c : \{0,1\}^n \to \{0,1\}$ has a size s DT, then*

- *c has an s term DNF.*
- *c has an s clause CNF.*

Proof: One can construct a DNF out of the DT associated to c as follows: For each of the paths to a leaf with a 1 in the DT there corresponds a conjunction of literals that imposes the same conditions. Therefore if one considers these as the terms of a DNF and forms a disjunction it gives rise to an s term DNF computing c. This gives the first result. The second result is also clear after this observation because given a size s DT, by negating only the leaf nodes one derives a size s DT for $\neg c$. Using this one can construct an s term DNF for $\neg c$ as described and negate it using DeMorgan's Law to obtain an s clause CNF for c. ■

The converse of this fact is not true: There exist functions with small size DNFs but large DTs.
Example: $(x_1 \dots x_{\log n}) \vee (x_{\log n+1} \dots x_{2\log n}) \vee \dots$ has a $n/\log n$ term DNF but any DT representing this function must have size $2^{n^{\Omega(1)}}$.

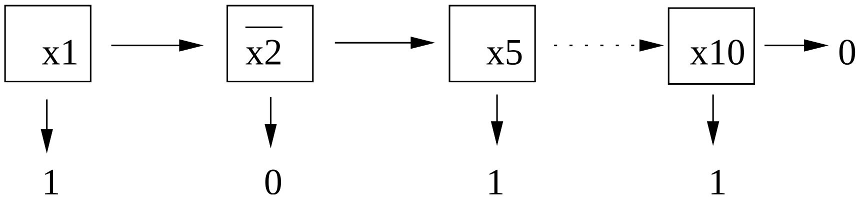

Figure 2: A decision list

The *depth* of a DT is measured by the length of the longest root-to-leaf path. Thus any depth-d DT has size at most 2^d and also has a d-DNF since one can express the function as a disjunction of terms with length d using the paths in the DT.

A *Decision List* is a DT with only one branch.

2.3 The concept class of functions represented by a linear threshold function

In this section we will assume the range of our functions to be $\{-1, 1\}$ instead of $\{0, 1\}$ for simplicity.

Definition 1 *$c : \{0,1\}^n \to \{-1,1\}$ is a **L**inear **T**hreshold **F**unction if there exist $\omega_1, \ldots, \omega_n, \theta \in \mathbb{R}$ such that for all $x \in \{0,1\}^n$, $c(x) = sign(\omega_1 x_1 + \ldots + \omega_n x_n - \theta)$.*

Example: $n = 4$, $c(x) = sign(3x_1 - 4x_2 + 2x_4 - 1)$ determines a LTF.

As earlier if one considers $\{0,1\}^n$ as an n dimensional cube then associated to such a function $\omega_1 x_1 + \ldots + \omega_n x_n - \theta$ is a hyperplane that would divide the cube into (at most) two monochromatic pieces. By using this idea it becomes clear that LTFs are not a universal representation scheme: Because Parity could not be expressed in the desired form.

Clearly one can perturb the coefficients $\omega_1, \ldots, \omega_n, \theta$ without affecting the LTF c, thus we have:

Fact 4 *Any LTF $c : \{0,1\}^n \to \{-1,1\}$ has infinitely many representations.*

Hence one can perturb the coefficients for the LTF in such a way that $\omega_1, \ldots, \omega_n, \theta \in \mathbb{Q}$ and then multiply all these coefficients with the common denominator to obtain the following fact.

Fact 5 *Any LTF has a representation with integer coefficients.*

Definition 2 *Given a LTF c, the* weight *of c is measured by the minimum value of $\sum_i |\omega_i|$ among all possible integer representations.*

Example: $c = x_1 \vee \ldots \vee x_k$ is an LTF since it could be expressed as $sign(x_1 + \ldots + x_k - 1)$. Moreover the weight of c is k.

Fact 6 (Muroga'72) *Every LTF on $\{0,1\}^n$ has weight at most $2^{O(n \log n)}$.*

This upper bound is tight up to a constant in the exponent thanks to the following result which requires sophisticated techniques.

Fact 7 (Hastad'94) *There are LTFs of weight $2^{\Omega(n \log n)}$.*

Question: Can you think of a simple LTF with weight $2^{\Theta(n)}$?