Active Sampling for Multiple Output Identification

Shai Fine[1]* and Yishay Mansour[2]**

[1] IBM Research Laboratory in Haifa, Israel.
[2] School of Computer Science, Tel Aviv University, Tel Aviv, Israel.

Abstract. We study functions with multiple output values, and use active sampling to identify an example for each of the possible output values. Our results for this setting include: (1) Efficient active sampling algorithms for simple geometric concepts, such as intervals on a line and axis parallel boxes. (2) A characterization for the case of binary output value in a transductive setting. (3) An analysis of active sampling with uniform distribution in the plane. (4) An efficient algorithm for the Boolean hypercube when each output value is a monomial.

1 Introduction

Active sampling is much about "hitting" low probability events. In active learning the active sampling is used to guide the learning process to learn a high accuracy hypothesis while using a limited number or examples [16, 9, 8, 10, 6, 3, 1, 2]. While in many applications the goal of an accurate hypothesis is the most natural one, there are other applications which require only examples of those low probability events. Example of such application areas include hardware and software verification, fault tolerance, network security, data mining etc. The usage of such examples in each of the applications can be very different: in fault tolerance one would like to simulate the performance of the system in extreme conditions (e.g., very high load), in network security one would like to have examples of potential intruders, while in data mining one would like to find new interesting relationships, which are not explained by the existing ones. Our original motivation, though, stem from dynamic hardware verification and from software testing. In both domains, the main industrial vehicles are simulation-based methods which aimed at exciting (and impacting) the occurrence of events and scenarios of desired functional behaviors that need to be verified [19, 4]. Coverage [12] is an information collection mechanism that is

* Email: shai@il.ibm.com
** Email: mansour@post.tau.ac.il. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778, by a grant no. 1079/04 from the Israel Science Foundation, by a grant from BSF and an IBM faculty award. This publication only reflects the authors' views.

often used to monitor the progress of the verification process, and point to areas in the design that have not been properly tested.

The analysis of coverage reports, and their translation to a set of directives that guides the implementation of the test plan, result in major manual bottlenecks in the otherwise highly automated verification process. A verification methodology called *coverage directed test generation* (CDG) aims to resolve this problem, either by utilizing a mechanism that can directly translate a verification task into a simulation test (CDG by Construction, cf. [17]), or by extracting useful information from the observed events, and use it to bias future simulation runs (CDG by Feedback, cf. [7] and ref. therein). However, too often, neither an accurate translation mechanism nor well structured coverage model can be provided, and we end up with the following naive, yet difficult, scenario: The verification team is given a list of events that should be covered, and the goal is to provide multiple sets of directives (inputs) that will tune the test generator to produce patterns that hit all items in the list. In these situations the common practice is "trial & error".

We abstracted the above motivation in the following learning model. There is an unknown target function f which maps every input in $\mathcal{X}$ to one of m output values. (For example, in the verification setting the output values would be desired scenarios for the coverage generator.) The *output identification task* is to find m inputs, one for each output value.

The output identification algorithm is given some information about the target function. First, like much of the computational learning literature, it knows that the target function f is in a given function class $\mathcal{F}$. Second, it is given the number of output values, i.e., m (hence it knows when to terminate).

We assume that there is an unknown distribution D over the inputs. The algorithm has an access to an *induced distribution example oracle* which allows it to sample from sub-regions of the domain. (Namely, the algorithm specifies a subset $Y \subset \mathcal{X}$, and the oracle returns an example from Y, distributed according to the induced distribution of D over Y.) The goal of the output identification algorithm is to minimize the number of oracle samples it requires until a representative for each output value is found. The performance is measured as a function of the number of output values m and ϵ, a lower bound on the probability of each output value. (We remark that only for simplicity we assume that m and ϵ are known to the output identification algorithm, both of the requirements can be easily relaxed and similar results hold.)

We show efficient algorithms for many classes of functions. We start by showing efficient output identification algorithms for a few simple geometric classes. For the function class of m intervals on a line we show an expected active sample bound of $O(m \log 1/\epsilon)$. For m axis parallel boxes in $\mathbb{R}^d$ we give an expected active sample bound of $O(md \log 1/\epsilon)$. We also derive lower bounds that exhibit classes with a constant VC dimension, such as a linear separator in the plane, which require $\Omega(1/\epsilon)$ active samples.

Our main result is a characterization for the case of binary outputs, i.e., $m = 2$. We define a separation dimension and show that if the separation dimension

is d then the function class can be output identified in $O(d^2 \log^2 s)$ queries in a transductive setting (where the output identification algorithm is given a set of s unlabeled examples in advance). In addition, we show that for any function class with separation dimension d, there is an unlabeled sample for which no algorithm can output identify it in less than $\Omega(\min\{d, s\})$ queries.

The separation dimension is similar in flavor to the VC dimension[18]. It requires d points such that the function class induced on them has all the d singleton functions (rather than all 2^d possible functions in the VC dimension). We show that when a class has separation dimension d, then some input in the unlabeled sample can be queried and guarantee that either we terminate (finding a representative for each output value) or we reduce the space of consistent functions considerably. Using this property we derive an efficient output identification algorithm.

We also study the case of specific distributions, namely the uniform distribution in the plane. Using classical results from computational geometry we can show that many classes are efficiently learnable under the uniform distribution. Specifically, we show that the class of linear separators in the plane can be output identified with expected $O(m^2 \log^2 1/\epsilon)$ active samples with respect to the uniform distribution over the unit square.

We conclude with a concept class defined over the Boolean hypercube $\{0,1\}^n$. We show that the class where each output value is represented by a monomial can be output identify in mn active samples.

Our model has obvious connections to active learning [16, 9, 8, 10, 6, 3, 1, 2]. In some sense the output identification task is much simpler than the usual learning task, since we do not need to find an accurate hypothesis, only to target one example for each output value. Still it seems that the techniques we present here and the active learning techniques share much in common. In both cases the goal is to reduce uncertainty, however since the tasks are different (learning vs. identification), so does the choice which of the samples to query – while an active learner will choose to query for the label of the sample which maximizes disagreement between the consistent hypotheses in the version space (cf. QBC [16, 8]), an output identifier will select to query for the label of the sample which maximize the probability of an unseen output value (ideally have the probability be almost one). Another major difference is that active learning is interested mainly in binary classification while the main motivation for output identification are cases with a large number of possible output values.

There are simple cases in which active learning fails to achieve a significant sample improvement, for example linear separator in the plane [1]. In such cases one should expect the output identification task to suffer from similar drawbacks (and indeed some of our lower bounds are much in that spirit).

A somewhat related question was discuss in [11] where efficient deterministic constructions for combinatorial hitting sets are given. A hitting set for a domain $\{1, \ldots, m\}^d$ guarantees to "hit" any combinatorial rectangle of volume at least ϵ, i.e., any combinatorial rectangle that includes at least ϵm^d points would intersect the hitting set in at least one point. The main contribution of [11] is to construct

such a hitting set *deterministically* (i.e., without any randomization) and have its size and computation time be polynomial in m, d and $1/\epsilon$.

2 The Model

A function f is m-valued if f maps inputs from a domain $\mathcal{X}$ to an m valued range $\{1, \ldots, m\}$. An m-valued function class $\mathcal{F}$ is a set of m valued functions.

The *output identification task* for an m valued function class $\mathcal{F}$ is as follows. There is an unknown m valued *target function* $f \in \mathcal{F}$ and the goal of the output identification algorithm is to identify an input for each the m output values, i.e., find examples, $x^1, \ldots, x^m \in \mathcal{X}$ such that $f(x^i) = i$.

The output identification algorithm has access to examples of the target function as follows. There is some unknown distribution D over the domain $\mathcal{X}$. Given a subset of the domain $Y \subset \mathcal{X}$ let D_Y be the distribution D induces over Y. An *induced distribution example oracle* receives as an input a subset $Y \subset \mathcal{X}$ and returns a pair $< x, f(x) >$, where x is distributed according to D_Y and f is the target function. (We assume that Y has a non-empty intersection with the support of D, otherwise the oracle would generate an error.)

At each time step t, the output identification algorithm specifies a subset $Y_t \subset \mathcal{X}$ to the induced distribution example oracle and received an example $< x_t, f(x_t) >$. The process terminates when the algorithm has an example for each of the m output values. (I.e., $x^1, \ldots, x^m \in \mathcal{X}$ such that $f(x^i) = i$.)

In order to measure the complexity of an output identification algorithm we assume that each output value has a probability of at least ϵ under (the unknown) distribution D.

The *active sample complexity* of an output identification algorithm with respect to a distribution D and function $f \in \mathcal{F}$ is the number of examples it requests, i.e., the number of times it accesses the induced distribution example oracle. The active sample complexity of an output identification algorithm for a function class $\mathcal{F}$ is the worse case over all $f \in \mathcal{F}$ and distributions D of its active sample complexity. The active sample complexity of a function class $\mathcal{F}$ is the least active sample complexity of any output identification algorithm for $\mathcal{F}$.

3 Simple Geometric Concepts

In this section we consider simple geometric concepts where the domain is $\mathcal{X} = \mathbb{R}^d$. We start with the line ($d = 1$) and consider the case where each of the m output values is represented by an interval. For this case we give an expected $O(m \log 1/\epsilon)$ active sample complexity output identification algorithm. We then extend our result to the case of axis parallel boxes and give an output identification algorithm whose expected active sample complexity is $O(dm \log 1/\epsilon)$. We end with a few simple lower bounds, showing examples of concept classes with a finite VC dimensions which require $\Omega(1/\epsilon)$ active sample complexity.

3.1 Generic Consistency Algorithm

We assume that the points mapped to a specific output value belong to some function class $\mathcal{A}$. (For intervals $A \in \mathcal{A}$ would be a single interval and for axis parallel boxes it would be a single axis parallel box.)

The *generic consistency algorithm* works as follows. Initially we have $C_0 = \emptyset$. At phase $t \geq 1$ we sample the induced distribution example oracle with the subset $\mathcal{X} - C_{t-1}$ and receive an example $< x_t, k_t >$. For each output value k let S_t^k be the set of examples sampled with output value k until time t. Let C_t^k be the *minimal concept* in A which is consistent with the examples in S_t^k (in our applications such a concept always exists). Let C_t be the union of all the C_t^k and proceed to phase $t+1$.

The generic consistency algorithm starts at phase $t = 1$ and terminate at the first phase where for every output value k we have $S_t^k \neq \emptyset$ (note that in such a case we have for each of the m output values at least one example).

To be more specific about the minimal consistent concept we define it for the case of intervals and axis parallel boxes. In the case that A is an interval then C_t^k is an interval $[\lambda_-^k, \lambda_+^k]$ such that $\lambda_-^k = \min\{x \in S_t^k\}$ and $\lambda_+^k = \max\{x \in S_t^k\}$. In the case that A is an axis parallel box then C_t^k is $([c_-^1, c_+^1], \ldots, [c_-^d, c_+^d])$ where $c_-^i = \min\{x_i : x \in S_t^k, x = x_1, \cdots x_d\}$ and $c_+^i = \max\{x_i : x \in S_t^k, x = x_1, \cdots x_d\}$.

The correctness of the generic consistency algorithm is obvious from its termination condition, the main interest in the analysis would be on the expected number of examples until termination, i.e., the expected active sample complexity.

3.2 Intervals on a line

In this function class the domain is $\mathcal{X} = [0, 1]$, the examples corresponding to an output value k are in an interval A^k (which can be either open or closed interval), and the intervals are a partition of the domain $\mathcal{X} = [0, 1]$, i.e. $\cup_{k=1}^m A^k = [0, 1]$ and $A^i \cap A^j = \emptyset$ for $i \neq j$.

For the analysis of the generic consistency algorithm we introduce some additional notation. At time t, we have a set KN_t of output values which we have already sampled and UKN_t which are output values we have not been sampled. For each output $k \in KN_t$ let B_-^k and B_+^k be the points in A^k below and above C^k (respectively). I.e., if $C_t^k = [\lambda_-^k, \lambda_+^k]$ and $A^k = [\rho_-, \rho_+]$ then $B_-^k = [\rho_-, \lambda_-)$ and $B_+^k = (\lambda_+, \rho_+]$.

Given a distribution D over $[0, 1]$ let $D(I)$ be the probability of the interval I. At time t let $\beta_t = \sum_{k \in UKN_t} D(A^k)$ and let $\gamma_t = \sum_{k \in KN_t} D(B_-^k) + D(B_+^k)$. Let $\alpha_t = \beta_t + \gamma_t$, i.e., $\alpha_t = D(\mathcal{X} - C_t)$. Let $\mathcal{H}_t$ include all the history of the execution of the algorithm until and including time t.

Our analysis uses a potential function $\Phi_t = \alpha_t + \beta_t = \gamma_t + 2\beta_t$, and shows that Φ_t decreases by a certain factor each time step. Specifically we show that,

$$E[\Phi_t - \Phi_{t+1} | \mathcal{H}_t] = \frac{\beta_t}{\alpha_t} \sum_{k \in UKN_t} \frac{D(A^k)}{\beta_t} D(A^k) + \frac{\gamma_t}{\alpha_t} \sum_{k \in KN_t, b \in \{+,-\}} \frac{D(B_b^k)}{\gamma_t} \cdot \frac{D(B_b^k)}{2}$$

The first part follows from the fact that with probability $\frac{\beta_t}{\alpha_t}$ we sample an example with output value in UKN_t. Given that we sample such a point, the probability that the interval is A^k is $\frac{D(A^k)}{\beta_t}$. Given that we sample from A^k we reduce β_t by $2D(A^k)$ and increase γ_t by $D(A^k)$, so the net reduction in the potential is $D(A^k)$.

The second part follows from the fact that with probability $\frac{\gamma_t}{\alpha_t}$ we sample an example with output value in KN_t. Given that we sample such a point, the probability that it is in the interval B_b^k is $\frac{D(B_b^k)}{\gamma_t}$, where $b \in \{-,+\}$. Given that we sample from B_b^k the expected reduction is $D(B_b^k)/2$. Therefore,

$$E[\Phi_t - \Phi_{t+1}|\mathcal{H}_t] = \frac{1}{\alpha_t}[\sum_{k \in UKN_t} D(A^k)^2 + \sum_{k \in KN_t, b \in \{+,-\}} \frac{1}{2} D(B_b^k)^2]$$

Note that

$$\alpha_t = \beta_t + \gamma_t = \sum_{k \in UKN_t} D(A^k) + \sum_{k \in KN_t, b \in \{+,-\}} D(B_b^k).$$

Using the general inequality $\sum_{i=1}^n X_i^2 \geq (1/n)(\sum_{i=1}^n X_i)^2$, and since there are at most $2m$ elements in the summation (each output value appears only in one of the two summations), we have that

$$\sum_{k \in UKN} D(A^k)^2 + \sum_{k \in KN, b \in \{+,-\}} \frac{1}{2} D(B_b^k)^2 \geq \frac{1}{4m} \alpha_t^2$$

Since by definition $\beta_t \leq \alpha_t$, this implies that

$$E[\Phi_t - \Phi_{t+1}|\mathcal{H}_t] \geq \frac{1}{4m} \alpha_t \geq \frac{1}{8m}[\alpha_t + \beta_t] = \frac{1}{8m} \Phi_t$$

By averaging over $\mathcal{H}_t$ we have that

$$E[\Phi_{t+1}] \leq (1 - \frac{1}{8m}) E[\Phi_t] \leq (1 - \frac{1}{8m})^t \Phi_1$$

Initially we have $\gamma_1 = 0$ and $\beta_1 = 1$, therefore the initial potential is $\Phi_1 = \gamma_1 + 2\beta_1 = 2$. After $t = O(m \log(1/\epsilon))$ samples, the expected value of the potential is less than $\epsilon/2$. This implies that with probability at least $1/2$ its value is less than ϵ. Once the value of the potential is less than ϵ we are guarantee to hit each output value (since each output value has probability at least ϵ). This establishes the following theorem.

Theorem 1. *The class of m intervals can be output identified in expected active sample complexity of $O(m \log 1/\epsilon)$.*

3.3 Axis parallel boxes

We extend the results from intervals to axis parallel boxes, where $\mathcal{X} = [0,1]^d$, each output value k is represented by an axis parallel box A^k, and the collection of A^k are a partition of $\mathcal{X}$. Again, we use the generic consistent algorithm.

The analysis is similar in spirit to that of the intervals on a line and it appears in the appendix, where we establish the following theorem:

Theorem 2. *The class of m axis parallel boxes can be output identified in expected active sample complexity of $O(dm \log 1/\epsilon)$.*

3.4 Lower bounds

In this section we derive two simple lower bounds.
Example 1: Let $\mathcal{X} = [0,1]$ and U be the uniform distribution on $\mathcal{X}$. Consider the following function class $\mathcal{F}^{\eta}_{seg}$ that includes functions $f_z(x) = 1$ if $x \in [z, z+\eta]$ and otherwise $f_z(x) = 0$ (where $z \in [0, 1-\eta]$). Note that $\mathcal{F}^{\eta}_{seg}$ has a VC dimension equals to 2, and in addition, the points with output value 0 are not a convex set. We show the following lower bound.

Claim. Any output identification algorithm for $\mathcal{F}^{\eta}_{seg}$ with the uniform distribution U requires an expected active sample complexity of $\Omega(1/\epsilon)$, when $\epsilon = \eta$.

Example 2: Consider a linear separator in the plane (there are only two output values). Let $\mathcal{X} = [-1,1]^2$ and let $\mathcal{F}_{ls}$ include all linear separators. Namely, for each $f_{\alpha,\beta} \in \mathcal{F}$ we have $f_{\alpha,\beta}(x) = 1$ if $\alpha x_1 + \beta < x_2$ and otherwise $f_{\alpha,\beta}(x) = 0$ (where $\alpha, \beta \in \mathbb{R}$).

Following Dasgupta [1], we consider a distribution U_o whose support is the unit circle, e.g., $(x_1)^2 + (x_2)^2 = 1$ and it is uniform over it. Similar to the lower bound for active learning [1], we show the following lower bound for output identification.

Claim. Any output identification algorithm for $\mathcal{F}_{ls}$ with distribution U_o requires an expected active sample complexity of at least $\Omega(1/\epsilon)$.

4 Transductive setting: Binary output values

In the transductive setting the algorithm is given in advance a set of unlabeled examples $S = \{x_1, \ldots x_s\}$. The goal of the output identification algorithm is to find a subset $S' \subset S$ of size at most m, such that each output value that appears in S has an example in S'. I.e., let $S^k = \{x \in S : f(x) = k\}$, we require that if $S^k \neq \emptyset$ then $S^k \cap S' \neq \emptyset$. The active sample complexity in the transductive setting is the number of queries the algorithm makes (i.e., the number of unlabeled examples from S for which it asks a label).

We give a characterization for the case of binary output values, i.e., $m = 2$. We first define the notion of a *separation dimension* of a function class $\mathcal{F}$. Then we show that if a function class $\mathcal{F}$ has separation dimension d then there is

an algorithm that queries only $O(d^2 \log^2 s)$ examples. In addition we show that if a function class has separation dimension d then the expected number of examples queried is $\Omega(\min\{d, s\})$. This implies that the for the binary case we have a complete characterization when can a function class be output identified with a poly-logarithmic number of queries.

4.1 Separation dimension: definition

We start by defining the notion of separation dimension. Let the separation dimension of a function class $\mathcal{F}$ be the following. A function class $\mathcal{F}$ is said to b-separate the set $\{x_1, \ldots x_d\} \subset \mathcal{X}$ if there are functions $f_1, \ldots f_d \in \mathcal{F}$ such that $f_i(x_i) = b$ and $f_i(x_j) = 1 - b$, for $j \neq i$, where $b \in \{0, 1\}$.

The b-separation dimension of $\mathcal{F}$ is the size of the largest set that $\mathcal{F}$ b-separates (or infinity, if it can b-separate sets of arbitrary size). The separation dimension of $\mathcal{F}$ is the maximum of the 0-separation dimension and the 1-separation dimension.

Separation dimension - examples: Let us start with a few examples of the separation dimension. Consider the function class $\mathcal{F}_{pre}$ over $[0, 1]$ such that $f_z(x) = 1$ if $z \leq x$ and $f_z(x) = 0$ otherwise (where $z \in [0, 1]$). The separation dimension of $\mathcal{F}_{pre}$ is 1 since for any two points $x_1 < x_2$ no function f_z can have $f_z(x_1) = 1$ and $f_z(x_2) = 0$.

A simple extension of $\mathcal{F}_{pre}$ is the function class $\mathcal{F}_{pre+suf}$ where $f_{z,b}(x) = b$ if $z \leq x$ and $f_z(x) = 1 - b$ otherwise (where $z \in [0, 1]$ and $b \in \{0, 1\}$). The separation dimension of $\mathcal{F}_{pre+suf}$ is 2. (Given, for example, $x_1 = 1/3$ and $x_2 = 2/3$, the functions $f_{1/2,1}$ and $f_{1/2,0}$ achieve a b-separation of $\{x_1, x_2\}$ for both $b = 1$ and $b = 0$. However, for any three points $x_1 < x_2 < x_3$ no function $f_{z,b}$ can have $f_{z,b}(x_2) = b$ and $f_{z,b}(x_1) = f_{z,b}(x_3) = 1 - b$, neither for $b = 1$ nor $b = 0$.)

Recall the function class $\mathcal{F}^{\eta}_{seg}$ that includes functions $f_z(x) = 1$ if $x \in [z, z+\eta]$ and otherwise $f_z(x) = 0$ (where $z \in [0, 1 - \eta]$). The function class $\mathcal{F}^{\eta}_{seg}$ has separation dimension of $\Theta(1/\eta)$.

An example of a function class with an infinite separation dimension is $\mathcal{F}_{ind}$, where for every $z \in \mathcal{X}$ we have an indicator function $f_z \in \mathcal{F}_{ind}$ (i.e., $f_z(x) = 1$ for $x = z$ and otherwise $f_z(x) = 0$), and $\mathcal{X}$ is infinite. Since for any set of s distinct points $x_1, \ldots, x_s \in \mathcal{X}$ the indicator functions $f_{x_1}, \ldots, f_{x_s}$ are a 1-separation, this implies that $\mathcal{F}_{ind}$ has an infinite separation dimension.

Separation dimension - Number of consistent functions: Given a set of points $S = \{x_1, \ldots x_s\}$ let $\mathcal{F}^S$ be the function class $\mathcal{F}$ restricted to the set S. We would like to bound the number of consistent functions $\mathcal{F}^S$ as a function of $s = |S|$ and separation dimension of the function class $\mathcal{F}$.

It is obvious that if a function class $\mathcal{F}$ has a separation dimension of d then it has a VC dimension [18] of at most d. Therefore, using Sauer Lemma [15], we can bound $|\mathcal{F}^S|$. We can also show a function class for which this bound is almost tight also for separation dimension.

Lemma 1. *Let $\mathcal{F}$ be a function class with separation dimension d, then $|\mathcal{F}^S| \leq \sum_{i=0}^{d} \binom{|S|}{i}$. In addition, there is a function class $\mathcal{F}$ of separation dimension d such that $|\mathcal{F}^S| \geq (|S|/d)^d$.*

4.2 Separation dimension: Lower bound

In this section we give a lower bound based on the separation dimension.

Theorem 3. *Let $\mathcal{F}$ be a function class with separation dimension d, then its expected active sample complexity is $\Omega(\min\{d, s\})$. In addition, if $\mathcal{F}$ has an infinite separation dimension then its expected active sample complexity is $\Omega(s)$.*

Proof. Assume that $\mathcal{F}$ has 1-separation dimension of d (the other case is identical). Then there are $k = \min\{d, s\}$ inputs $x_1, \ldots x_k \in \mathcal{X}$ such that for any i there is an $f_i \in \mathcal{F}$ that $f_i(x_i) = 1$ and $f_i(x_j) = 0$, for $j \neq i$. (When $k < s$, we extend the the set of k points to s points by duplicating point x_1 for $s - k$ times.)

Assume that we select the target function f to be f_i with probability $1/k$. Then any output identification algorithm would have to make at least $\Omega(k)$ queries before hitting the input which is labeled 1.

If $\mathcal{F}$ has an infinite separation dimension, then for every s, there are s inputs $x_1, \ldots x_s \in \mathcal{X}$ and functions $f_1, \ldots f_s \in \mathcal{F}$ such that $f_i(x_i) = 1$ and $f_i(x_j) = 0$, for $j \neq i$. Again, this implies that the expected number of queries is $\Omega(s)$. □

4.3 Separation dimension: Upper Bound

In this section we derive an upper bound on the sample complexity based of the separation dimension. Assume that the first point we sampled has label 0, and therefore the output identification task reduces to finding an input with label 1. (For this reason we will also concentrate on the 1-separation dimension.)

We start with a few notations. Let $\texttt{sup}(f, S) = \{x_i \in S : f(x_i) = 1\}$, i.e., the set of points in S on which f is 1. Given a set of functions $\mathcal{F}$, let $\texttt{deg}(x_i, \mathcal{F}^S) = \{f \in \mathcal{F}^S : f(x_i) = 1\}$, i.e., the set of functions which classify x_i as 1. We partition $\mathcal{F}^S$ according to the size of $\texttt{sup}(f, S)$, and define $\mathcal{F}_k^S = \{f \in \mathcal{F}^S : k \leq |\texttt{sup}(f, S)| < 2k\}$. Let $\ell = |\mathcal{F}|$ and $\ell_k = |\mathcal{F}_k^S|$.

Lemma 2. *Let $\mathcal{F}$ be a function class with 1-separation dimension d. Given an unlabeled sample S, for every $k \geq 1$, there exists an input $x_i \in S$ such that $|\texttt{deg}(x_i, \mathcal{F}_k^S)| \geq \lfloor \ell_k / 8d \rfloor$.*

Proof. For contradiction, assume that for some $k \geq 1$ no such $x_i \in S$ exists. We will show that the 1-separation dimension is at least $d + 1$, which would be a contradiction.

Since we assume that no such x_i exists for k, then for every $x_i \in S$ we have $|\texttt{deg}(x_i, \mathcal{F}_k^S)| < \ell_k / 8d$. By definition of $\mathcal{F}_k^S$, for every $f \in \mathcal{F}_k^S$ we have $|\texttt{sup}(f, S)| \in [k, 2k)$. Consider the set of pairs Z that includes all the pairs (f, x) such that $x \in S$, $f \in \mathcal{F}_k^S$ and $f(x) = 1$. There are at least $k\ell_k$ such pairs in Z.

We would like to find a subset of pairs $(f_1, x_1) \ldots, (f_{d+1}, x_{d+1})$ in Z such that for any $i \neq j$ we have $f_j(x_i) = 0$. Recall that by the definition of the pairs in Z we have $f_i(x_i) = 1$. Therefore, such a subset would imply that the separation dimension of $\mathcal{F}$ is at least $d + 1$.

For the first pair we pick any (f, x) in Z. We would like to delete some of the pairs in Z such that any remaining pair (h, z) has the property that $f(z) = 0$ and $h(x) = 0$. This would guarantee that the subset that we select would have the required property.

Formally, we delete from Z both the set $\{(g, y) | g \in \mathcal{F}_k^S, y \in \texttt{sup}(f, S)\}$ and the set $\{(g, y) | g \in \mathcal{F}_k^S, g \in \texttt{deg}(x, \mathcal{F}_k^S)\}$. The deletion of the first set guarantees that any remaining pair (h, z) would have $f(z) = 0$ while the deletion of the second set guarantees that for any remaining pair (h, z) we have $h(x) = 0$. The size of the first set is at most $\sum_{y \in \texttt{sup}(f,S)} \texttt{deg}(y, \mathcal{F}_k^S) \leq k\ell_k/4d$ while the size of the second set is at most $\sum_{h \in \texttt{deg}(x, \mathcal{F}_k^S)} \texttt{sup}(h, S) \leq k\ell_k/4d$. This implies that we delete at most $k\ell_k/2d$ pairs.

By iteratively selecting a pair from Z and deleting from Z the two related sets, this implies that we can select $2d \geq d+1$ such pairs. This is a contradiction to the assumption that separation dimension is d. □

We can now use Lemma 2 to derive an output identification algorithm in the transductive setting.

Theorem 4. *Let $\mathcal{F}$ be a function class of separation dimension d. For any unlabeled sample S, it can be output identified in active sample complexity of $O(d \log |S| \log |\mathcal{F}^S|)$.*

Proof. Initially we query an arbitrary point in S. W.l.o.g., assume that the first point has a label 0. This implies that we need to search for a point in S with a label of 1. Note that the 1-separation dimension of $\mathcal{F}$ is at most d.

We run the algorithm in rounds, where in each round we select at most $\log |S|$ inputs (one for to each $\mathcal{F}_{2^i}^S$, where $i \in [0, \log |S|])$. In each round for each $\mathcal{F}_{2^i}^S$ we select the input x which maximizes $\texttt{deg}(x, \mathcal{F}_{2^i}^S)$. By Lemma 2, since $\mathcal{F}$ has 1-separation dimension of at most d, there exists an input x_i such that $|\texttt{deg}(x, \mathcal{F}_{2^i}^S)| \geq \ell_k/8d$. Therefore, in each round, the number of possible target functions in each $\mathcal{F}_{2^i}^S$ shrink by a factor of $(1-1/8d)$. After at most $O(d \log |\mathcal{F}^S|)$ rounds we will either: (1) find an $x \in S$ with label 1, i..e, $f(x) = 1$, or, (2) the only remaining consistent function in $\mathcal{F}^S$ is the all zero function, i.e., there are no points in S with a label of 1. □

Since, by Lemma 1, a d separation dimension implies that $|\mathcal{F}^S| = O(|S|^d)$, we have,

Corollary 1. *If $\mathcal{F}$ has separation dimension d then any unlabeled sample S can be output identified with an active sample complexity of $O(d^2 \log^2 |S|)$.*

5 Uniform distribution

In this section we discuss active sample complexity for specific distributions. We will concentrate on the case that the input distribution is uniform over the unit square, i.e., $\mathcal{X} = [0,1]^2$. In this case we will be able to show that many natural geometric concepts, which for a general distribution they require $\Omega(1/\epsilon)$, are efficiently output identified with respect to the uniform distribution.

Generic Convex-Hull Algorithm: Our algorithm would be a generic consistency algorithm (Section **??**) for the case where the domain of each output value is convex. Initially we have $S_0 = C_0 = \emptyset$. At time t we sample the induced distribution example oracle with $\mathcal{X} - C_t$ and receive an example $< x_t, k_t >$. For each output value k let S_t^k be the set of points sampled with output value k. Let C_t^k be the convex hull of the points in S_t^k, and let C_t be the union of all those sets. We terminate when for every output value k we have $S_t^k \neq \emptyset$.

Review from computational geometry: There are classical results in computational geometry regarding uniformly sampling from the plane. The results related the area of the convex-hull to the total area of the convex domain from which the points are sampled at uniform. For a set of points Z let $ConvexHull(Z)$ be their convex hull and for a body Y let $area(Y)$ be its area. The following theorem summarizes the related results (see, cf., [13]).

Theorem 5 ([14, 5]). *Let G be an r-gon in $[0,1]^2$, and S_n be a sample of size n sampled uniformly from G. Then, $E[area(G - ConvexHull(S_n))] = \Theta(\frac{r \ln n}{n})$.*

Triangles in the plane: Let $\mathcal{X} = [0,1]^2$ be the unit square. Consider the case that there are m output values such that the domain of each output value is a triangle and their union is the unit square.

It would be more beneficial in this case to consider an alternative way of sampling. Assume that each time we access the induced distribution example oracle with $\mathcal{X} - C_t$, the oracle samples from $\mathcal{X}$ until it hits a point $x \notin C_t$. In our analysis let us consider also the extra points that oracle samples (but we will give them zero weight). Assume that in this process the total number of samples the oracle makes is T. (Recall that the generic convex hull algorithm terminates when each of the m output values is sampled at least once.) Let T_i be the number of times the oracle samples output value i (out of the T samples). Let X_j^i be a random variable which is 1 if the j-th point with output value i is outside the convex hull of the previous $j-1$ points, and 0 otherwise. This implies that the expected number of samples of the generic convex hull algorithm is $E_T E_{T_1 \ldots T_m}[\sum_{i=1}^{m} \sum_{j=1}^{T_i} X_j^i]$.[3]

[3] This follows, since an equivalent way of sampling, is to sample always points uniformly from the unit square, and to request a label (and charge) only for points not in C_t.

From Theorem 5, applied to triangles (i.e., $r = 3$), we can deduce that, $E[X^i_{j+1}] = O(\frac{\ln j}{j})$. Therefore, $\sum_{j=1}^{T_i} E[X^i_{j+1}] = \sum_{j=1}^{T_i} O(\frac{\ln j}{j}) \leq \alpha \log^2 T_i$, for some constant $\alpha > 0$. Summing over all possible output values we have

$$E_T E_{T_1,\ldots T_m} E[\sum_{i=1}^{m} \sum_{j=1}^{T_i} X^i_j] \leq E_T \left[\alpha \sum_{i=1}^{m} E_{T_i}[\log^2 T_i] \right] \leq E_T[\alpha m \log^2 \frac{T}{m}] \leq \alpha m \log^2 \frac{1}{\epsilon}$$

where the second inequality follows since $\sum_{i=1}^{m} T_i = T$, and the last inequality uses the fact that $E[T] \leq m/\epsilon$ and the concavity of the logarithm function. This implies the following theorem,

Theorem 6. *Let $\mathcal{F}_\Delta$ be a function class such that every $f \in \mathcal{F}_\Delta$ partitions the unit square to m triangles each of area at least ϵ. Then for the uniform distribution U, the expected active sample complexity of the generic convex hull algorithm is $O(m \log^2 1/\epsilon)$.*

Lines in the plane: Consider the case where there are k lines in the plane that partition the unit square. Each of the m output values is one of the cells created by the intersection of the lines. We can perform a triangulation of the cells and have at most $O(k^2)$ triangles. (By performing a triangulation we are only increasing the running time of the generic convex hull algorithm, since each output value is de-composed to smaller convex hulls.) Since k lines in the plane create at least k cells, this implies that $m \geq k$. By using Theorem 6, we can show the following,

Theorem 7. *Let $\mathcal{F}_{k-line}$ be a function class such that every $f \in \mathcal{F}_{k-line}$ is represented by at most k lines in the plane. If each of the m cells has area at least ϵ then for the uniform distribution U the expected active sample complexity of the generic convex hull algorithm is $O(m^2 \log^2 1/\epsilon)$.*

Note that $\mathcal{F}_{ls}$, for which we showed a lower bound of $\Omega(1/\epsilon)$ with respect to an arbitrary distribution, is simply $\mathcal{F}_{1-line}$ and $m = 2$.

6 Monomials in a hypercube

In this section we will concentrate on the Boolean cube and consider the case that each output value is a monomial. Formally, the domain $\mathcal{X}$ is $\{0,1\}^n$. The function class $\mathcal{F}_{mon}$ includes functions of the following type: For every output k there is a monomial M_k such that $f(x) = k$ iff $M_k(x) = 1$, where $f \in \mathcal{F}_{mon}$. (Note that the monomials M_i are a partition of the hypercube.)

For any i let x^i be the input $x \in \{0,1\}^n$ with the i-th bit flipped. Note that if $M_k(x) = 1$ then $M_k(x^i) = 0$ iff M_k depends on attribute x_i. This would be the basic property that will allow us to efficiently output identify $\mathcal{F}_{mon}$.

Monomial Output Identification algorithm: The algorithm performs an exact identification using (essentially) membership queries. Let KN be the set of known output values and CH be a set of inputs we need to "check". Initially both $KN = \emptyset$ and $CH = \emptyset$. In the first phase, we sample any x, get its value $f(x) = k$ and add k to KN and x to CH. In every phase we take an input x from CH, and for every i such that $f(x) \neq f(x^i)$, if $f(x^i) \notin KN$ then we add the output value $f(x^i)$ to KN, add the input x^i to CH, and continue to the next input from CH. We terminate either when we have processed all the values in CH or have already recovered m output values (i.e., $|KN| = m$).

Theorem 8. *For any distribution which has non-zero probability for every example $\{0,1\}^n$, the class $\mathcal{F}_{mon}$ can be output identified with active sample complexity of mn.*

Note that in this case we also end with a complete model of the target function f in addition to the m output values.

References

1. S. Dasgupta. Analysis of a greedy active learning strategy. In *Advances in Neural Information Processing Systems (NIPS)*, 2004.
2. S. Dasgupta. Coarse sample complexity bounds for active learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2005.
3. S. Dasgupta, A. Kalai, and C. Monteleoni. Analysis of perceptron-based active learning. In *Eighteenth Annual Conference on Learning Theory (COLT)*, 2005.
4. O. Edelstein, E. Farchi, Y. Nir G Ratzaby, and S Ur. Multithreaded java program test generation. *IBM Systems Journal*, 41(3):111–125, 2002.
5. B. Efron. The convex hull of a rndom set of points. *Biometrika*, 52:331–343, 1965.
6. S. Fine, R. Gilad-Bachrach, and E. Shamir. Query by committee, linear separation and random walks. *Theoretical Computer Science*, 284(1), 2002. (A preliminary version appeared in EuroColt 1999.).
7. S. Fine and A. Ziv. Coverage directed test generation for functional verification using Bayesian networks. In *Proceedings of the 40th Design Automation Conference*, pages 286–291, June 2003.
8. Y. Freund, H. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2/3):133–168, 1997.
9. S. R. Kulkarni, S. K. Mitter, and J. N. Tsitsiklis. Active learning using arbitrary binary valued queries. *Machine Learning*, 11:23–35, 1993.
10. R. Liere and P. Tadepalli. Active learning with committees for text categorization. In *AAAI-97*, 1997.
11. N. Linial, M. Luby, M. Saks, and D. Zuckerman. Efficient construction of a small hitting set for combinatorial rectangles in high dimension. *Combinatorica*, 17(2):215–234, 1997. (A preliminary version appeard in STOC 1993).
12. Andrew Piziali. *Functional Verification Coverage Measurement and Analysis*. Springer, 2004.
13. F. P. Preparata and M. I Shamos. *Computational Geometry: An introduction*. Springer-Verlag, 1985.
14. A. Rényi and R. Sulamke. Uber die konvexe hulle von n zufallig gewahlten punkten. *Z. Wahrschein*, 2:75–84, 1963.

15. N. Sauer. On the density of family of sets. *J. of Combinatorial Theory*, Ser. A 13:145–147, 1972.
16. H. S. Seung, M. Opper, and H. Sompolinsky. Query by committe. In *Proceedings of the Fith Workshop on Computational Learning Theory*, pages 287–294. Morgan Kaufman, San Mateo, CA, 1992.
17. S. Ur and Y. Yadin. Micro-architecture coverage directed generation of test programs. In *Proceedings of the 36th Design Automation Conference*, pages 175–180, June 1999.
18. V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its applications*, XVI(2):264–280, 1971.
19. B. Wile, J. C. Goss, and W. Roesner. *Comprehensive Functional Verification The Complete Industry Cycle*. Elsevier, 2005.

A Axis parallel boxes: Analysis

The analysis of the axis parallel boxes would be similar to the analysis of intervals on a line. At time t, we have a set KN_t of output values which we have already sampled and UKN_t which are output values we have not sampled. For each output value $k \in KN_t$ we define $2d$ boxes which extend the current C_t^k to A^k, in each of the d dimensions both above and below. (Note that the boxes overlap unlike in the intervals case.)

Formally, let $C_t^k = ([c_-^1, c_+^1], \ldots, [c_-^d, c_+^d])$ and $A^k = ([a_-^1, a_+^1], \ldots, [a_-^d, a_+^d])$. Since $C_t^k \subset A^k$, then $a_-^i \leq c_-^i \leq c_+^i \leq a_+^i$. The $2d$ boxes are

$$B_-^{k,i} = [a_-^1, a_+^1], \ldots [a_-^{i-1}, a_+^{i-1}], [a_-^i, c_-^i), [a_-^{i+1}, a_+^{i+1}], \ldots, [a_-^d, a_+^d]).$$

and

$$B_+^{k,i} = [a_-^1, a_+^1], \ldots [a_-^{i-1}, a_+^{i-1}], (c_+^i, a_+^i], [a_-^{i+1}, a_+^{i+1}], \ldots, [a_-^d, a_+^d]).$$

Let, $\gamma_t = \sum_{k \in KN_t} \sum_{i=1}^{d} D(B_+^{k,i}) + D(B_-^{k,i})$, $\beta_t = \sum_{k \in UKN_t} D(A^k)$, and $\alpha_t = D(\mathcal{X} - C_t)$. The potential function would be $\Phi_t = \gamma_t + 2d\beta_t$. For intuition, note that when we sample an output value $k \in UNK_t$ for the first time, we decrease β_t by $D(A^k)$ and increase γ_t by at most $dD(A^k)$, so the net reduction in the potential is at least $dD(A^k)$. Let $\mathcal{H}_t$ include all the history of the execution of the algorithm until and including time t.

We will show that the potential Φ_t decreases by a certain factor each time step. Specifically we show that

$$E[\Phi_t - \Phi_{t+1} | \mathcal{H}_t] = \frac{\beta_t}{\alpha_t} \sum_{k \in UKN_t} \frac{D(A^k)}{\beta_t} dD(A^k) + \frac{\alpha_t - \beta_t}{\alpha_t} \sum_{k \in KN_t, i \in [1,d], b \in \{+,-\}} \frac{D(B_b^{k,i})}{\alpha_t - \beta_t} \cdot \frac{D(B_b^{k,i})}{2}$$

The first part follows from the fact that with probability $\frac{\beta_t}{\alpha_t}$ we sample an example with output value in UKN_t. Given that we sample such an example, the

probability that the output value is k is $\frac{D(A^k)}{\beta_t}$. Given that we sample from A^k we reduce β_t by $D(A^k)$ and increase γ_t by at most $dD(A^k)$, so the net reduction in the potential is at least $dD(A^k)$.

The second part follows from the fact that with probability $\frac{\alpha_t-\beta_t}{\alpha_t}$ we sample an example with output value in KN_t. Given that we sample such an example, the probability that the example is in box $B_b^{k,i}$ is $\frac{D(B_b^{k,i})}{\alpha_t-\beta_t}$. (Note that a point can be in more than one box. We used here the linearity of expectations, to be able to consider each box separately.) Given that we sample from $B_b^{k,i}$ the expected reduction is $D(B_b^{k,i})/2$. Therefore,

$$E[\Phi_t - \Phi_{t+1}|\mathcal{H}_t] = \frac{1}{\alpha_t}[\sum_{k\in UKN_t} dD(A^k)^2 + \sum_{k\in KN_t, i\in[1,d], b\in\{+,-\}} \frac{1}{2}D(B_b^{k,i})^2]$$

Again, we use the general inequality $\sum_{i=1}^n X_i^2 \geq (1/n)(\sum_{i=1}^n X_i)^2$, for each summation separately,

$$\sum_{k\in UKN_t} dD(A^k)^2 + \sum_{k\in KN_t, i\in[1,d] b\in\{+,-\}} \frac{1}{2}D(B_b^{k,i})^2 \geq \frac{1}{m}d\beta_t^2 + \frac{1}{4dm}\gamma_t^2$$
$$= \frac{1}{4dm}(4d^2\beta_t^2 + \gamma_t^2)$$

Therefore,

$$E[\Phi_t - \Phi_{t+1}|\mathcal{H}_t] \geq \frac{1}{4dm}\frac{4d^2\beta_t^2 + \gamma_t^2}{\alpha_t}$$

Since each point can be counted at most d times in γ_t, we have that,

$$\beta_t + \gamma_t \geq \alpha_t = D(\mathcal{X} - C_t) \geq \beta_t + \gamma_t/d$$

This implies that

$$E[\Phi_t - \Phi_{t+1}|\mathcal{H}_t] \geq \frac{1}{4dm}\frac{4d^2\beta_t^2 + \gamma_t^2}{\beta_t + \gamma_t} \geq \frac{1}{8dm}[2d\beta_t + \gamma_t] = \frac{1}{8dm}\Phi_t,$$

where in the second inequality we use that $X^2 + Y^2 \geq \frac{1}{2}(X+Y)^2$. By averaging over $\mathcal{H}_t$ we have that

$$E[\Phi_{t+1}] \leq (1 - \frac{1}{8dm})E[\Phi_t] \leq (1 - \frac{1}{8dm})^t\Phi_1$$

Initially we have $\gamma_1 = 0$ and $\beta_1 = 1$, therefore the initial potential is $\Phi_1 = 2d$. After $t = O(dm\log(1/\epsilon))$ samples the expected potential is less than $\epsilon/2$. Therefore with probability at least $1/2$ the potential is less than ϵ. If the potential is less than ϵ, this implies that we hit every output value (since we assume that each output value has probability of at least ϵ). Therefore we have establishes Theorem 2.