MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY
and
CENTER FOR BIOLOGICAL AND COMPUTATIONAL LEARNING
DEPARTMENT OF BRAIN AND COGNITIVE SCIENCES

A.I. Memo No. 1514 March, 1995
C.B.C.L. Paper No. 113

Active Learning by Sequential Optimal Recovery

Partha Niyogi

Abstract

In most classical frameworks for learning from examples, it is assumed that examples are randomly drawn and presented to the learner. In this paper, we consider the possibility of a more *active* learner who is allowed to choose his/her own examples. Our investigations are carried out in a function approximation setting. In particular, using arguments from optimal recovery (Micchelli and Rivlin, 1976), we develop an adaptive sampling strategy (equivalent to adaptive approximation) for arbitrary approximation schemes. We provide a general formulation of the problem and show how it can be regarded as sequential optimal recovery. We demonstrate the application of this general formulation to two special cases of functions on the real line 1) monotonically increasing functions and 2) functions with bounded derivative. An extensive investigation of the sample complexity of approximating these functions is conducted yielding both theoretical and empirical results on test functions. Our theoretical results (stated in PAC-style), along with the simulations demonstrate the superiority of our active scheme over both passive learning as well as classical optimal recovery. The analysis of active function approximation is conducted in a worst-case setting, in contrast with other Bayesian paradigms obtained from optimal design (Mackay, 1992).

This report describes research done at the Center for Biological and Computational Learning and the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Center is provided in part by a grant from the National Science Foundation under contract ASC–9217041.

1 Introduction

In the classical paradigm of learning from examples, the data or examples, are typically drawn according to some fixed, unknown, arbitrary probability distribution. This is the case for a wide class of problems in the PAC (Valiant, 1984) framework, as well as other familiar frameworks in the connectionist (Rumelhart et al, 1986) and pattern recognition literature. In this important sense, the learner is merely a *passive* recipient of information about the target function. In this paper, we consider the possibility of a more *active* learner. There are of course a myriad of ways in which a learner could be more active. Consider, for example, the extreme pathological case where the learner simply asks for the true target function which is duly provided by an obliging oracle. This, the reader will quickly realize is hardly interesting. Such pathological cases aside, this theme of activity on the part of the learner has been explored (though it is not always conceived as such) in a number of different settings (PAC-style concept learning, boundary-hunting pattern recognition schemes, adaptive integration, optimal sampling etc.) in more principled ways and we will comment on these in due course.

For our purposes, we restrict our attention in this paper to the situation where the learner is allowed to choose its own examples[1], for function approximation tasks. In other words, the learner is allowed to decide where in the domain D (for functions defined from D to Y) it would like to sample the target function. Note that this is in direct contrast to the passive case where the learner is presented with randomly drawn examples. Keeping other factors in the learning paradigm unchanged, we then compare in this paper, the active and passive learners who differ only in their method of collecting examples. At the outset, we are particularly interested in whether there exist principled ways of collecting examples in the first place. A second important consideration is whether these ways allow the learner to learn with a fewer number of examples. This latter question is particularly important as one needs to assess the advantage, from an information-theoretic point of view, of active learning.

Are there principled ways to choose examples? We develop a general framework for collecting (choosing) examples for approximating (learning) real-valued functions. This can be viewed as a sequential version of optimal recovery (Michhelli and Rivlin, 1977): a scheme for optimal sampling of functions. Such an active learning scheme is consequently in a worst-case setting, in contrast with other schemes (Mackay, 1992; Cohn, 1993; Sollich, 1993) that operate within a Bayesian, average-case paradigm. We then demonstrate the application of sequential optimal recovery to some specific classes of functions. We obtain theoretical bounds on the sample complexity of the active and passive learners, and perform some empirical simulations to demonstrate the superiority of the active learner.

[1] This can be regarded as a computational instantiation of the psychological practice of *selective attention* where a human might choose to selectively concentrate on interesting or confusing regions of the feature space in order to better grasp the underlying concept. Consider, for example, the situation when one encounters a speaker with a foreign accent. One cues in to this foreign speech by focusing on and then adapting to its distinguishing properties. This is often accomplished by asking the speaker to repeat words which are confusing to us.

2 A General Framework For Active Approximation

2.1 Preliminaries

We need to develop the following notions:

$\mathcal{F}$: Let $\mathcal{F}$ denote a class of functions from some domain D to Y where Y is a subset of the real line. The domain D is typically a subset of R^k though it could be more general than that. There is some unknown target function $f \in \mathcal{F}$ which has to be approximated by an approximation scheme.

$\mathcal{D}$: This is a data set obtained by sampling the target $f \in \mathcal{F}$ at a number of points in its domain. Thus,

$$\mathcal{D} = \{(x_i, y_i) | x_i \in D, y_i = f(x_i),\ i = 1 \dots n\}$$

Notice that the data is uncorrupted by noise.

$\mathcal{H}$: This is a class of functions (also from D to Y) from which the learner will choose one in an attempt to approximate the target f. Notationally, we will use $\mathcal{H}$ to refer not merely to the class of functions (hypothesis class) but also the algorithm by means of which the learner picks an approximating function $h \in \mathcal{H}$ on the basis of the data set $\mathcal{D}$. In other words, $\mathcal{H}$ denotes an approximation scheme which is really a tuple $< \mathcal{H}, A >$. A is an algorithm that takes as its input the data set $\mathcal{D}$, and outputs an $h \in \mathcal{H}$.

Examples: If we consider real-valued functions from R^k to R, some typical examples of $\mathcal{H}$ are the class of polynomials of a fixed order (say q), splines of some fixed order, radial basis functions with some bound on the number of nodes, etc. As a concrete example, consider functions from $[0, 1]$ to R. Imagine a data set is collected which consists of examples, i.e., (x_i, y_i) pairs as per our notation. Without loss of generality, one could assume that $x_i \leq x_{i+1}$ for each i. Then a cubic (degree-3) spline is obtained by interpolating the data points by polynomial pieces (with the pieces tied together at the data points or "knots") such that the overall function is twice-differentiable at the knots. Fig. 1 shows an example of an arbitrary data set fitted by cubic splines.

$\mathbf{d_C}$: We need a metric to determine how good the approximation learner's approximation is. Specifically, the metric d_C measures the approximation error on the region C of the domain D. In other words, d_C, takes as its input any two functions (say f_1 and f_2) from D to R and outputs a real number. It is assumed that d_C satisfies all the requisites for being a real distance metric on the appropriate space of functions. Since the approximation error on a larger domain is obviously going to be greater than that on the smaller domain, we can make the following two observations: 1) for any two sets C_1 and C_2 such

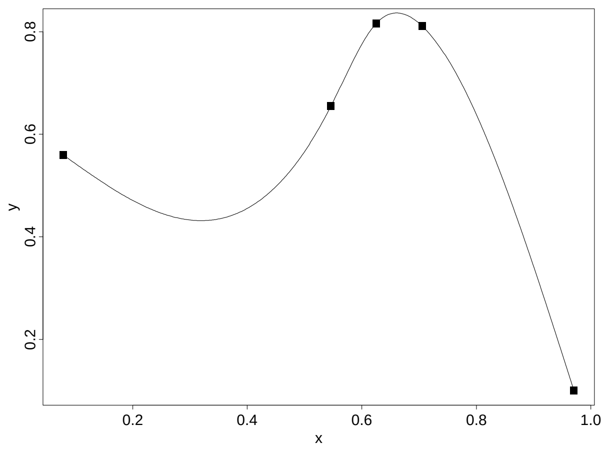

Figure 1: An arbitrary data set fitted with cubic splines

that $C_1 \subset C_2$, $d_{C_1}(f_1, f_2) \leq d_{C_2}(f_1, f_2)$, 2) $d_D(f_1, f_2)$ is the total approximation on the entire domain; this is our basic criterion for judging the "goodness" of the learner's hypothesis.
Examples: For real-valued functions from R^k to R, the L^p_C metric defined as $d_C(f_1, f_2) = (\int_C |f_1 - f_2|^p dx)^{1/p}$ serves as a natural example of an error metric.

$\mathcal{C}$: This is a collection of subsets C of the domain. We are assuming that points in the domain where the function is sampled, divide (partition) the domain into a collection of disjoint sets $C_i \in \mathcal{C}$ such that $\cup_{i=1}^n C_i = D$.
Examples: For the case of functions from $[0, 1]$ to R, and a data set $\mathcal{D}$, a natural way in which to partition the domain $[0, 1]$ is into the intervals $[x_i, x_{i+1})$, (here again, without loss of generality we have assumed that $x_i \leq x_{i+1}$). The set $\mathcal{C}$ could be the set of all (closed, open, or half-open and half-closed) intervals $[a, b] \subset [0, 1]$.

The goal of the learner (operating with an approximation scheme $\mathcal{H}$) is to provide a hypothesis $h \in \mathcal{H}$ (which it chooses on the basis of its example set $\mathcal{D}$) as an approximator of the unknown target function $f \in \mathcal{F}$. We now need to formally lay down a criterion for assessing the competence of a learner (approximation scheme). In recent times, there has been much use of PAC (Valiant 1984) like criteria to assess learning algorithms. Such a criterion has been used largely for concept learning but some extensions to the case of real valued functions exist (Haussler 1989). We adapt here for our purposes a PAC like criterion to judge the efficacy of approximation schemes of the kind described earlier.

Definition 1 *An approximation scheme is said to P-PAC learn the function $f \in \mathcal{F}$ if for every $\epsilon > 0$ and $1 > \delta > 0$, and for an arbitrary distribution P on D, it collects a data set $\mathcal{D}$, and computes a hypothesis $h \in \mathcal{H}$ such that $d_D(h, f) < \epsilon$ with probability greater than $1 - \delta$. The function class $\mathcal{F}$ is P-PAC learnable if the approximation scheme can P-PAC learn every function in $\mathcal{F}$. The class $\mathcal{F}$ is PAC learnable if the approximation scheme can P-PAC learn the class for every distribution P.*

There is an important clarification to be made about our definition above. Note that the distance metric d is arbitrary. It need not be naturally related to the distribution P according to which the data is drawn. Recall that this is not so in typical distance metrics used in classical PAC formulations. For example, in concept learning, where the set $\mathcal{F}$ consists of indicator functions, the metric used is the $L_1(P)$ metric given by $d(1_A, 1_B) = \int_D |1_A - 1_B| P(x) dx$. Similarly, extensions to real-valued functions typically use an $L_2(P)$ metric. The use of such metrics imply that the training error is an empirical average of the true underlying error. One can then make use of convergence of empirical means to true means (Vapnik, 1982) and prove learnability. In our case, this is not necessarily the case. For example, one could always come up with a distribution P which would never allow a passive learner to see examples in a certain region of the domain. However, the arbitrary metric d might weigh this region heavily. Thus the learner would never be able to learn such a function class for this metric. In this sense, our model is more demanding than classical PAC. To make matters easy, we will consider here the case of $P - PAC$ learnability alone, where P is a known distribution (uniform in the example cases studied). However, there is a sense in which our notion of PAC is easier —the learner knows the true metric d and given any two functions, can compute their relative distance. This is not so in classical PAC, where the learner cannot compute the distance between two functions since it does not know the underlying distribution.

We have left the mechanism of data collection undefined. Our goal here is the investigation of different methods of data collection. A baseline against which we will compare all such schemes is the *passive* method of data collection where the learner collects its data set by sampling D according to P and receiving the point $(x, f(x))$. If the learner were allowed to draw its own examples, are there principled ways in which it could do this? Further, as a consequence of this flexibility accorded to the learner in its data gathering scheme, could it learn the class $\mathcal{F}$ with fewer examples? These are the questions we attempt to resolve in this paper, and we begin by motivating and deriving in the next section, a general framework for *active* selection of data for arbitrary approximation schemes.

2.2 The Problem of Collecting Examples

We have introduced in the earlier section, our baseline algorithm for collecting examples. This corresponds to a *passive* learner that draws examples according to the probability distribution P on the domain D. If such a passive learner collects examples and produces an output h such that $d_D(h, f)$ is less than ϵ with probability greater than $1 - \delta$, it P-PAC learns the function. The number of examples that a learner needs before it produces such an (ϵ-good,δ-confidence) hypothesis is called its *sample complexity*.

Against this baseline passive data collection scheme, lies the possibility of allowing the learner to choose its own examples. At the outset it might seem reasonable to believe that a data set would provide the learner with some information about the target function; in particular, it would probably inform it about the "interesting" regions of the function, or regions where the approximation error is high and need further sampling. On the basis of this kind of information (along with other information about the class of functions in general) one might be able to decide where to sample next. We formalize this notion as follows:

Let $\mathcal{D} = \{(x_i, y_i); i = 1 \ldots n\}$ be a data set (containing n data points) which the learner has access to. The approximation scheme acts upon this data set and picks an $h \in \mathcal{H}$ (which best fits the data according to the specifics of the algorithm A inherent in the approximation scheme). Further, let $C_i; i = 1, \ldots, K(n)^2$ be a partition of the domain D into different regions on the basis of this data set. Finally let

$$\mathcal{F}_{\mathcal{D}} = \{f \in \mathcal{F} | f(x_i) = y_i \ \forall (x_i, y_i) \in \mathcal{D}\}$$

This is the set of all functions in $\mathcal{F}$ which are consistent with the data seen so far. The target function could be any one of the functions in $\mathcal{F}_{\mathcal{D}}$.

We first define an error criterion e_C (where C is any subset of the domain) as follows:

$$e_C(\mathcal{H}, \mathcal{D}, \mathcal{F}) = \sup_{f \in \mathcal{F}_{\mathcal{D}}} d_C(h, f)$$

Essentially, e_C is a measure of the maximum possible error the approximation scheme could have (over the region C) given the data it has seen so far. It clearly depends on the data, the approximation scheme, and the class of functions being learned. It does not depend upon the target function (except indirectly in the sense that the data is generated by the target function after all, and this dependence is already captured in the expression). We thus have a scheme to measure *uncertainty* (maximum possible error) over the different regions of the input space D. One possible strategy to select a new point might simply be to sample the function in the region C_i where the error bound is the highest. Let us assume we have a procedure $\mathcal{P}$ to do this. $\mathcal{P}$ could be to sample the region C at the centroid of C, or sampling C according to some distribution on it, or any other method one might fancy. This can be described as follows:

Active Algorithm A

1. **[Initialize]** Collect one example (x_1, y_1) by sampling the domain D once according to procedure $\mathcal{P}$.
2. **[Obtain New Partitions]** Divide the domain D into regions $C_1, \ldots, C_{K(1)}$ on the basis of this data point.
3. **[Compute Uncertainties]** Compute e_{C_i} for each i.
4. **[General Update and Stopping Rule]** In general, at the jth stage, suppose that our partition of the domain D is into $C_i, i = 1 \ldots K(j)$. One can compute e_{C_i} for each i and sample the region with maximum uncertainty (say C_l) according to procedure $\mathcal{P}$. This would provide a new data point (x_{j+1}, y_{j+1}). The new data point would re-partion the domain D into new regions. At any stage, if the maximum uncertainty over the entire domain e_D is less than ϵ stop.

The above algorithm is one possible active strategy. However, one can carry the argument a little further and obtain an optimal sampling strategy which would give us a precise location for the next sample point. Imagine for a moment, that the learner asks for the value of the function at a point $x \in D$. The value returned obviously belongs to the set

$$\mathcal{F}_{\mathcal{D}}(x) = \{f(x) | f \in \mathcal{F}_{\mathcal{D}}\}$$

Assume that the value observed was $y \in \mathcal{F}_{\mathcal{D}}(x)$. In effect, the learner now has one more example, the pair (x, y), which it can add to its data set to obtain a new, larger data set $\mathcal{D}'$ where

$$\mathcal{D}' = \mathcal{D} \cup (x, y)$$

Once again, the approximation scheme $\mathcal{H}$ would map the new data set $\mathcal{D}'$ into a new hypothesis h'. One can compute

$$e_C(\mathcal{H}, \mathcal{D}', \mathcal{F}) = \sup_{f \in \mathcal{F}_{\mathcal{D}'}} d(h', f)$$

Clearly, $e_D(\mathcal{H}, \mathcal{D}', \mathcal{F})$ now measures the maximum possible error after seeing this new data point. This depends upon (x, y) (in addition to the usual $\mathcal{H}, \mathcal{D}$, and $\mathcal{F}$). For a fixed x, we don't know the value of y we would observe if we had chosen to sample at that point. Consequently, a natural thing to do at this stage is to again take a worst case bound, i.e., assume we would get the most unfavorable y and proceed. This would provide the maximum possible error we could make if we had chosen to sample at x. This error (over the entire domain) is

$$\sup_{y \in \mathcal{F}_{\mathcal{D}}(x)} e_D(\mathcal{H}, \mathcal{D}', \mathcal{F}) = \sup_{y \in \mathcal{F}_{\mathcal{D}}(x)} e_D(\mathcal{H}, \mathcal{D} \cup (x, y), \mathcal{F})$$

Naturally, we would like to sample the point x for which this maximum error is minimized. Thus, the optimal point to sample by this argument is

$$x_{new} = \arg\min_{x \in D} \sup_{y \in \mathcal{F}_{\mathcal{D}}(x)} e_D(\mathcal{H}, \mathcal{D} \cup (x, y), \mathcal{F}) \qquad (1)$$

This provides us with a principled strategy to choose our next point. The following optimal active learning algorithm follows:

Active Algorithm B (Optimal)

1. **[Initialize]** Collect one example (x_1, y_1) by sampling the domain D once according to procedure $\mathcal{P}$. We do this because without any data, the approximation scheme would not be able to produce any hypothesis.

[2] The number of regions $K(n)$ into which the domain D is partitioned by n data points depends upon the geometry of D and the partition scheme used. For the real line partitioned into intervals as in our example, $K(n) = n + 1$. For k-cubes, one might obtain Voronoi partitions and compute $K(n)$ accordingly.

2. **[Compute Next Point to Sample]** Apply eq. 1 and obtain x_2. Sampling the function at this point yields the next data point (x_2, y_2) which is added to the data set.
3. **[General Update and Stopping Rule]** In general, at the jth stage, assume we have in place a data set $\mathcal{D}_j$ (consisting of j data). One can compute x_{j+1} according to eq. 1 and sampling the function here one can obtain a new hypothesis and a new data set $\mathcal{D}_{j+1}$. In general, as in Algorithm A, stop whenever the total error $e_D(\mathcal{H}, \mathcal{D}_k, \mathcal{F})$ is less than ϵ.

By the process of derivation, it should be clear that if we chose to sample at some point other than that obtained by eq. 1, an adversary could provide a y value and a function consistent with all the data provided (including the new data point), that would force the learner to make a larger error than if the learner chose to sample at x_{new}. In this sense, algorithm B is optimal. It also differs from algorithm A, in that it does not require a partition scheme, or a procedure $\mathcal{P}$ to choose a point in some region. However, the computation of x_{new} inherent in algorithm B is typically more intensive than computations required by algorithm A. Finally, it is worthwhile to observe that crucial to our formulation is the derivation of the error bound $e_D(\mathcal{H}, \mathcal{D}, \mathcal{F})$. As we have noted earlier, this is a measure of the maximum possible error the approximation scheme $\mathcal{H}$ could be forced to make in approximating functions of $\mathcal{F}$ using the data set $\mathcal{D}$. Now, if one wanted an approximation scheme independent bound, this would be obtained by minimzing e_D over all possible schemes, i.e.,

$$\inf_{\mathcal{H}} e_D(\mathcal{H}, \mathcal{D}, \mathcal{F})$$

Any approximation scheme can be forced to make at least as much error as the above expression denotes. Another bound of some interest is obtained by removing the dependence of e_D on the data. Thus given an approximation scheme $\mathcal{H}$, if data $\mathcal{D}$ is drawn randomly, one could compute

$$P\{e_D(\mathcal{H}, \mathcal{D}, \mathcal{F}) > \epsilon\}$$

or in an approximation scheme-independent setting, one computes

$$P\{\inf_{\mathcal{H}} e_D(\mathcal{H}, \mathcal{D}, \mathcal{F}) > \epsilon\}$$

The above expressions would provide us PAC-like bounds which we will make use of later in this paper.

2.3 In Context

Having motivated and derived two possible active strategies, it is worthwhile at this stage to comment on the formulation and its place in the context of previous work in similar vein executed across a number of disciplines.

1) Optimal Recovery: The question of choosing the location of points where the unknown function will be sampled has been studied within the framework of optimal recovery (Micchelli and Rivlin, 1976; Micchelli and Wahba, 1981; Athavale and Wahba, 1979). While work of this nature has strong connections to our formulation, there remains a crucial difference. Sampling schemes motivated by optimal recovery are not adaptive. In other words, given a class of functions $\mathcal{F}$ (from which the target f is selected), optimal sampling chooses the points $x_i \in D, i = 1, \ldots, n$ by optimizing over the entire function space $\mathcal{F}$. Once these points are obtained, then they remain fixed irrespective of the target (and correspondingly the data set $\mathcal{D}$). Thus, if we wanted to sample the function at n points, and had an approximation scheme $\mathcal{H}$ with which we wished to recover the true target, a typical optimal recovery formulation would involve sampling the function at the points obtained as a result of optimizing the following objective function:

$$\arg\min_{x_1,\ldots,x_n} \sup_{f\in\mathcal{F}} d(f, h(\mathcal{D} = \{(x_i, f(x_i))_{i=1\ldots n}\})) \quad (2)$$

where $h(\mathcal{D} = \{(x_i, f(x_i))_{i=1\ldots n}\}) \in \mathcal{H}$ is the learner's hypothesis when the target is f and the function is sampled at the x_i's. Given no knowledge of the target, these points are the optimal to sample.

In contrast, our scheme of sampling can be conceived as an iterative application of optimal recovery (one point at a time) by conditioning on the data seen so far. Making this absolutely explicit, we start out by asking for one point using optimal recovery. We obtain this point by

$$\arg\min_{x_1} \sup_{f\in\mathcal{F}} d(f, h(\mathcal{D}_1 = \{(x_1, f(x_1))\}))$$

Having sampled at this point (and obtained y_1 from the true target), we can now reduce the class of candidate target functions to $\mathcal{F}_1$, the elements of $\mathcal{F}$ which are consisitent with the data seen so far. Now we obtain our second point by

$$\arg\min_{x_2} \sup_{f\in\mathcal{F}_1} d(f, h(\mathcal{D}_2 = \{(x_1, y_1), (x_2, f(x_2))\}))$$

Note that the supremum is done over a restricted set $\mathcal{F}_1$ the second time. In this fashion, we perform optimal recovery at each stage, reducing the class of functions over which the supremum is performed. It should be made clear that this sequential optimal recovery is *not* a greedy technique to arrive at the solution of eq. 2. It will give us a different set of points. Further, this set of points will depend upon the target function. In other words,the sampling strategy adapts itself to the unknown target f as it gains more information about that target through the data. We know of no similar sequential sampling scheme in the literature.

While classical optimal recovery has the formulation of eq. 2, imagine a situation where a "teacher" who knows the target function and the learner, wishes to communicate to the learner the best set of points to minimize the error made by the learner. Thus given a function g, this best set of points can be obtained by the following optimization

$$\arg\min_{x_1,\ldots,x_n} d(g, h(\{(x_i, g(x_i))\}_{i=1\ldots n})) \quad (3)$$

Eq. 2 and eq. 3 provide two bounds on the performance of the active learner following the strategy of Algorithm B in the previous section. While eq. 2 chooses optimal points without knowing anything about the target, and, eq. 3 chooses optimal points knowing the target completely, the active learner chooses points optimally on the basis of partial information about the target (information provided by the data set).

2) Concept Learning: The PAC learning community (which has traditionally focused on concept learning) typically incorporates activity on the part of the learner by means of queries, the learner can make of an oracle. Queries (Angluin, 1988) range from membership queries (is x an element of the target concept c) to statistical queries (Kearns, 1993 ; where the learner can not ask for data but can ask for estimates of functionals of the function class) to arbitrary boolean valued queries (see Kulkarni etal for an investigation of query complexity). Our form of activity can be considered as a natural adaptation of membership queries to the case of learning real-valued functions in our modified PAC model. It is worthwhile to mention relevant work which touches the contents of this paper at some points. The most significant of these is an investigation of the sample complexity of active versus passive learning conducted by Eisenberg and Rivest (1990) for a simple class of unit step functions. It was found that a binary search algorithm could vastly outperform a passive learner in terms of the number of examples it needed to (ϵ, δ) learn the target function. This paper is very much in the spirit of that work focusing as it does on the sample complexity question. Another interesting direction is the transformation of PAC-learning algorithms from a batch to online mode. While Littlestone etal (1991) consider online learning of linear functions, Kimber and Long (1992) consider functions with bounded derivatives which we examine later in this paper. However the question of choosing one's data is not addressed at all. Kearns and Schapire (1990) consider the learning of p-concepts (which are essentially equivalent to learning classes of real-valued functions with noise) and address the learning of monotone functions in this context. Again, there is no active component on the part of the learner.

3)Adaptive Integration: The novelty of our formulation lies in its adaptive nature. There are some similarities to work in adaptive numerical integration which are worth mentioning. Roughly speaking, an adaptive integration technique (Berntsen etal 1991, book???) divides the domain of integration into regions over which the integration is done. Estimates are then obtained of the error on each of these regions. The region with maximum error is subdivided. Though the spirit of such an adaptive approach is close to ours, specific results in the field naturally differ because of differences between the integration problem (and its error bounds) and the approximation problem.

4) Bayesian and other formulations: It should be noted that we have a worst-case formulation (the supremum in our formulation represents the maximum possible error the scheme might have). Alternate bayesian schemes have been devised (Mackay, 1991; Cohn, 1994) from the perspective of optimal experiment design (Fedorov). Apart from the inherently different philosophical positions of the two schemes, an indepth treatment of the sample complexity question is not done. We will soon give two examples where we address this sample complexity question closely. In a separate piece of work (Sung and Niyogi, 1994) , the author has also investigated such bayesian formulations from such an information-theoretic perspective. Yet another average-case formulation comes from the information-complexity viewpoint of Traub and Wozniakovski (see Traub etal (1988) for details). Various interesting sampling strategies are suggested by research in that spirit. We do not attempt to compare them due to the difficulty in comparing worst-case and average-case bounds.

Thus, we have motivated and derived in this section, two possible active strategies. The formulation is general. We now demonstrate the usefulness of such a formulation by considering two classes of real-valued functions as examples and deriving specific active algorithms from this perspective. At this stage, the important question of sample complexity of active versus passive learning still remains unresolved. We investigate this more closely by deriving theoretical bounds and performing empirical simulation studies in the case of the specific classes we consider.

3 Example 1: A Class of Monotonically Increasing Bounded Functions

Consider the following class of functions from the interval $[0, 1] \subset \Re$ to $\Re$:

$$\mathcal{F} = \{f : 0 \leq f \leq M, \text{ and } f(x) \geq f(y) \forall x \geq y\}$$

Note that the functions belonging to this class need not be continuous though they do need to be measurable. This class is PAC- learnable (with an $L_1(P)$ norm, in which case our notion of PAC reduces to the classical notion) though it has infinite pseudo-dimension[3](in the sense of Pollard (1984)). Thus, we observe:

Observation 1 *The class $\mathcal{F}$ has infinite pseudo-dimension (in the sense of Pollard (1984); Haussler (1989),).*

Proof: To have infinite pseudo-dimension, it must be the case that for every $n > 0$, there exists a set of points $\{x_1, \ldots, x_n\}$ which is shattered by the class $\mathcal{F}$. In other words, there must exist a fixed translation vector $\mathbf{t} = (t_1, \ldots, t_n)$ such that for every boolean vector $\mathbf{b} = (b_1, \ldots, b_n)$, there exists a function $f \in \mathcal{F}$ which satisfies $f(x_i) - t_i > 0 \Leftrightarrow b_i = 1$. To see that this is indeed the case, let the n points be $x_i = i/(n+1)$ for i going from 1 to n. Let the translation vector then be given by $t_i = x_i$. For an arbitrary boolean vector $\mathbf{b}$ we can always come up with a monotonic function such that $f(x_i) = i/(n+1) - 1/3(n+1)$ if $b_i = 0$ and $f(x_i) = i/(n+1) + 1/3(n+1)$ if $b_i = 1$. □

We also need to specify the terms $\mathcal{H}$, d_C, the procedure $\mathcal{P}$ for partitioning the domain $D = [0, 1]$ and so on. For our purposes, we assume that the approximation scheme $\mathcal{H}$ is first order splines. This is simply finding the monotonic function which interpolates the data

[3]Finite pseudo-dimension is only a sufficient and not necessary condition for PAC learnability as this example demonstrates.

in a piece-wise linear fashion. A natural way to partition the domain is to divide it into the intervals $[0, x_1)$, $[x_1, x_2), \ldots, [x_i, x_{i+1}), \ldots, [x_n, 1]$. The metric d_C is an L_p metric given by $d_C(f_1, f_2) = (\int_0^1 |f_1 - f_2|^p dx)^{1/p}$.

Note that we are specifically interested in comparing the sample complexities of passive and active learning. We will do this under a uniform distributional assumption, i.e., the *passive* learner draws its examples by sampling the target function uniformly at random on its domain $[0, 1]$. In contrast, we will show how our general formulation in the earlier section translates into a specific *active* algorithm for choosing points, and we derive bounds on its sample complexity. We begin by first providing a lower bound for the number of examples a passive PAC learner would need to draw to learn this class $\mathcal{F}$.

3.1 Lower Bound for Passive Learning

Theorem 1 *Any passive learning algorithm (more specifically, any approximation scheme which draws data uniformly at random and interpolates the data by any arbitrary bounded function) will have to draw at least* $\frac{1}{2}(M/2\epsilon)^p \ln(1/\delta)$ *examples to P-PAC learn the class where P is a uniform distribution.*

Proof: Consider the uniform distribution on $[0, 1]$ and a subclass of functions which have value 0 on the region A $= [0, 1 - (2\epsilon/M)^p]$ and belong to $\mathcal{F}$. Suppose the passive learner draws l examples uniformly at random. Then with probability $(1 - (2\epsilon/M)^p)^l$, all these examples will be drawn from region A. It only remains to show that for the subclass considered, whatever be the function hypothesized by the learner, an adversary can force it to make a large error.

Suppose the learner hypothesizes that the function is h. Let the value of $(\int_{(1-(2\epsilon/M)^p,1)} |h(x)|^p dx)^{1/p}$ be χ. Obviously $0 \leq \chi \leq (M^p(2\epsilon/M)^p)^{1/p} = 2\epsilon$. If $\chi < \epsilon$, then the adversary can claim that the target function was really

$$g(x) = \begin{cases} 0 & \text{for } x \in [0, 1 - (2\epsilon/M)^p] \\ M & \text{for } x \in (1 - (2\epsilon/M)^p, 1] \end{cases}$$

If, on the other hand $\chi \geq \epsilon$, then the adversary can claim the function was really $g = 0$.

In the first case, by the triangle inequality,

$$d(h, g) = (\int_{[0,1]} |g - h|^p dx)^{1/p} \geq$$

$$(\int_{[1-(2\epsilon/M)^p,1]} |g - h|^p dx)^{1/p}$$

$$\geq (\int_{(1-(2\epsilon/M)^p,1)} M^p dx)^{1/p} - (\int_{(1-(2\epsilon/M)^p,1)} |h|^p dx)^{1/p}$$

$$= 2\epsilon - \chi > \epsilon$$

In the second case,

$$d(h, g) = (\int_{[0,1]} |g - h|^p dx)^{1/p} \geq$$

$$(\int_{(1-(2\epsilon/M)^p,1)} |0 - h|^p dx)^{1/p} = \chi > \epsilon$$

Now we need to find out how large l must be so that this particular event of drawing all examples in A is not very likely, in particular, it has probability less than δ.

For $(1 - (2\epsilon/M)^p)^l$ to be greater than δ, we need $l < \frac{1}{-\ln(1-(2\epsilon/M)^p)} \ln(\frac{1}{\delta})$. It is a fact that for $\alpha < 1/2$, $\frac{1}{2\alpha} \leq \frac{1}{-\ln(1-\alpha)}$. Making use of this fact (and setting $\alpha = (2\epsilon/M)^p$, we see that for $\epsilon < (\frac{M}{2})(\frac{1}{2})^{1/p}$, we have $\frac{1}{2}(M/2\epsilon)^p \ln(1/\delta) < \frac{1}{-\ln(1-(2\epsilon/M)^p)} \ln(\frac{1}{\delta})$. So unless l is greater than $\frac{1}{2}(M/2\epsilon)^p \ln(1/\delta)$, the probability that all examples are chosen from A is greater than δ. Consequently, with probability greater than δ, the passive learner is forced to make an error of atleast ϵ, and PAC learning cannot take place. $\square$

3.2 Active Learning Algorithms

In the previous section we computed a lower bound for passively PAC learning this class for a uniform distribution[4]. Here we derive an active learning strategy (the CLA algorithm) which would meaningfully choose new examples on the basis of information gathered about the target from previous examples. This is a specific instantiation of the general formulation, and interestingly yields a "divide and conquer" binary searching algorithm starting from a different philosophical standpoint. We formally prove an upper bound on the number of examples it requires to PAC learn the class. While this upper bound is a worst case bound and holds for all functions in the class, the actual number of queries (examples) this strategy takes differs widely depending upon the target function. We demonstrate empirically the performance of this strategy for different kinds of functions in the class in order to get a feel for this difference. We derive a classical non-sequential optimal sampling strategy and show that this is equivalent to uniformly sampling the target function. Finally, we are able to empirically demonstrate that the active algorithm outperforms both the passive and uniform methods of data collection.

3.2.1 Derivation of an optimal sampling strategy

Consider an approximation scheme of the sort described earlier attempting to approximate a target function $f \in \mathcal{F}$ on the basis of a data set $\mathcal{D}$. Shown in fig. 2 is a picture of the situation. We can assume without loss of generality that we start out by knowing the value of the function at the points $x = 0$ and $x = 1$. The points $\{x_i; i = 1, \ldots, n\}$ divide the domain into $n + 1$ intervals C_i (i going from 0 to n) where $C_i = [x_i, x_{i+1}] (x_0 = 0, x_{n+1} = 1)$.The monotonicity constraint on $\mathcal{F}$ permits us to obtain rectangular boxes showing the values that the target function could take at the points on its domain. The set of all functions which lie within these boxes as shown is $\mathcal{F}_{\mathcal{D}}$.

Let us first compute $e_{C_i}(\mathcal{H}, \mathcal{D}, \mathcal{F})$ for some interval C_i. On this interval, the function is constrained to lie in the appropriate box. We can zoom in on this box as shown in fig. 3.

The maximum error the approximation scheme could

[4] Naturally, this is a distribution-free lower bound as well. In other words, we have demonstrated the existence of a distribution for which the passive learner would have to draw at least as many examples as the theorem suggests.

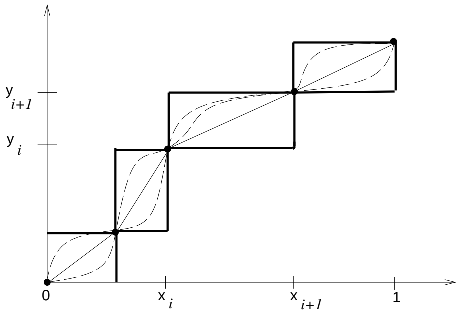

Figure 2: A depiction of the situation for an arbitrary data set. The set $\mathcal{F}_{\mathcal{D}}$ consists of all functions lying in the boxes and passing through the datapoints (for example, the dotted lines). The approximating function h is a linear interpolant shown by a solid line.

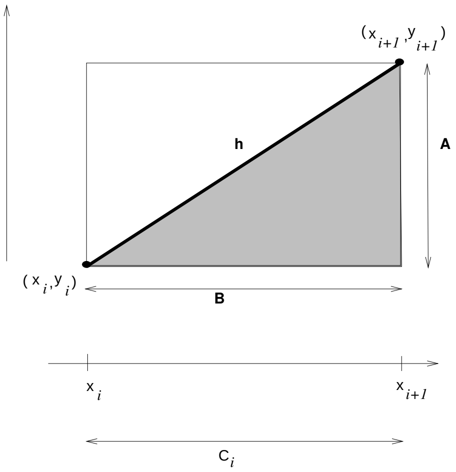

Figure 3: Zoomed version of interval. The maximum error the approximation scheme could have is indicated by the shaded region. This happens when the adversary claims the target function had the value y_i throughout the interval.

have (indicated by the shaded region) is clearly given by

$$(\int_{C_i} |h - f(x_i)|^p dx)^{1/p} = (\int_0^B (\frac{A}{B}x)^p dx)^{1/p} = AB^{1/p}/(p+1)^{1/p}$$

where $A = f(x_{i+1}) - f(x_i)$ and $B = (x_{i+1} - x_i)$.

Clearly the error over the entire domain e_D is given by

$$e_D^p = \sum_{i=0}^{n} e_{C_i}^p \qquad (4)$$

The computation of e_C is all we need to implement an active strategy motivated by Algorithm A in section 2. All we need to do is sample the function in the interval with largest error; recall that we need a procedure $\mathcal{P}$ to determine how to sample this interval to obtain a new data point. We choose (arbitrarily) to sample the midpoint of the interval with the largest error yielding the following algorithm.

The Choose and Learn Algorithm (CLA)

1. **[Initial Step]** Ask for values of the function at points $x = 0$ and $x = 1$. At this stage, the domain $[0, 1]$ is composed of one interval only, viz., $[0, 1]$. Compute $E_1 = \frac{1}{(p+1)^{1/p}}(1-0)^{1/p}|(f(1)-f(0))|$ and $T_1 = E_1$. If $T_1 < \epsilon$, stop and output the linear interpolant of the samples as the hypothesis, otherwise query the midpoint of the interval to get a partition of the domain into two subintervals $[0, 1/2)$ and $[1/2, 1]$.

2. **[General Update and Stopping Rule]** In general, at the kth stage, suppose that our partition of the interval $[0, 1]$ is $[x_0 = 0, x_1)$,$[x_1, x_2)$, $\ldots$, $[x_{k-1}, x_k = 1]$. We compute the normalized error $E_i = \frac{1}{(p+1)^{1/p}}(x_i - x_{i-1})^{1/p}|(f(x_i) - f(x_{i-1}))|$ for all $i = 1, .., k$. The midpoint of the interval with maximum E_i is queried for the next sample. The total normalized error $T_k = (\sum_{i=1}^k E_i^p)^{1/p}$ is computed at each stage and the process is terminated when $T_k \leq \epsilon$. Our hypothesis h at every stage is a linear interpolation of all the points sampled so far and our final hypothesis is obtained upon the termination of the whole process.

Now imagine that we chose to sample at a point $x \in C_i = [x_i, x_{i+1}]$ and received the value $y \in \mathcal{F}_{\mathcal{D}}(x)$ (i.e., y in the box) as shown in the fig. 4. This adds one more interval and divides C_i into two subintervals C_{i1} and C_{i2} where $C_{i1} = [x_i, x]$ and $C_{i2} = [x, x_{i+1}]$. We also correspondingly obtain two smaller boxes inside the larger box within which the function is now constrained to lie. The uncertainty measure e_C can be recomputed taking this into account.

Observation 2 *The addition of the new data point (x, y) does not change the uncertainty value on any of the other intervals. It only affects the interval C_i which got subdivided. The total uncertainty over this interval*

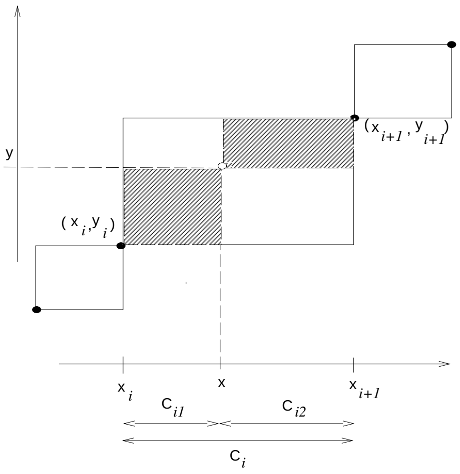

Figure 4: The situation when the interval C_i is sampled yielding a new data point. This subdivides the interval into two subintervals and the two shaded boxes indicate the new constraints on the function.

is now given by

$$e_{C_i}(\mathcal{H},\mathcal{D}',\mathcal{F}) = (\frac{1}{p+1})^{1/p}((x-x_i)(y-f(x_i))^p +$$

$$(x_{i+1}-x))((f(x_{i+1})-f(x_i))-y)^p)^{1/p} =$$

$$= G\,(zr^p + (B-z)(A-r)^p)^{1/p}$$

where for convenience we have used the substitution $z = x - x_i$, $r = y - f(x_i)$, *and* A *and* B *are* $f(x_{i+1}) - f(x_i)$ *and* $x_{i+1} - x_i$ *as above. Clearly* z *ranges from 0 to* B *while* r *ranges from* 0 *to* A.

We first prove the following lemma:

Lemma 1

$$B/2 = \arg\min_{z\in[0,B]}\sup_{r\in[0,A]} G\,(zr^p + (B-z)(A-r)^p)^{1/p}$$

Proof: Consider any $z \in [0,B]$. There are three cases to consider:

Case I $z > B/2$: let $z = B/2 + \alpha$ where $\alpha > 0$. We find

$$\sup_{r\in[0,A]} G\,(zr^p + (B-z)(A-r)^p)^{1/p} =$$

$$\left(\sup_{r\in[0,A]} G\,(zr^p + (B-z)(A-r)^p)\right)^{1/p}$$

Now,

$$\sup_{r\in[0,A]} G\,(zr^p + (B-z)(A-r)^p) =$$

$$\sup_{r\in[0,A]} G\,((B/2+\alpha)r^p + (B/2-\alpha)(A-r)^p)$$

$$= G \sup_{r\in[0,A]} B/2(r^p + (A-r)^p) + \alpha(r^p - (A-r)^p)$$

Now for $r = A$, the expression within the supremum $B/2(r^p + (A-r)^p) + \alpha(r^p - (A-r)^p)$ is equal to $(B/2 + \alpha)A^p$. For any other $r \in [0,A]$, we need to show that

$$B/2(r^p + (A-r)^p) + \alpha(r^p - (A-r)^p) \leq (B/2+\alpha)A^p$$

or

$$B/2((r/A)^p + (1-(r/A))^p) + \alpha((r/A)^p - (1-r/A)^p) \leq B/2 + \alpha$$

Putting $\beta = r/A$ (clearly $\beta \in [0,1]$, and noticing that $(1-\beta)^p \leq 1-\beta^p$ and $\beta^p - (1-\beta)^p \leq 1$ the inequality above is established. Consequently, we are able to see that

$$\sup_{r\in[0,A]} G\,(zr^p + (B-z)(A-r)^p)^{1/p} = G(B/2+\alpha)^{1/p}A$$

Case II Let $z = B/2 - \alpha$ for $\alpha > 0$. In this case, by a similar argument as above, it is possible to show that again,

$$\sup_{r\in[0,A]} G\,(zr^p + (B-z)(A-r)^p)^{1/p} = G(B/2+\alpha)^{1/p}A$$

Case III Finally, let $z = B/2$. Here

$$\sup_{r\in[0,A]} G\,(zr^p + (B-z)(A-r)^p)^{1/p} =$$

$$G(B/2)^{1/p} \sup_{r\in[0,A]} (r^p + (A-r)^p)^{1/p}$$

Clearly, then for this case, the above expression is reduced to $GA(B/2)^{1/p}$. Considering the three cases, the lemma is proved.□

The above lemma in conjunction with eq. 4 and observation 2 proves that if we choose to sample a particular interval C_i then sampling the midpoint is the optimal thing to do. In particular, we see that

$$\min_{x\in[x_i,x_{i+1}]} \sup_{y\in[f(x_i),f(x_{i+1})]} e_{C_i}(\mathcal{H},\mathcal{D}\cup(x,y),\mathcal{F}) =$$

$$(\frac{1}{p+1})^{1/p}(\frac{x_{i+1}-x_i}{2})^{1/p}(f(x_{i+1})-f(x_i)) = e_{C_i}(\mathcal{H},\mathcal{D},\mathcal{F})/2^{1/p}$$

In other words, if the learner were constrained to pick its next sample in the interval C_i, then by sampling the midpoint of this interval C_i, the learner ensures that the maximum error it could be forced to make by a malicious adversary is minimized. In particular, if the uncertainty over the interval C_i with its current data set $\mathcal{D}$ is e_{C_i}, the uncertainty over this region will be reduced after sampling its midpoint and can have a maximum value of $e_{C_i}/2^{1/p}$.

Now which interval must the learner sample to minimize the maximum possible uncertainty over the entire domain $D = [0,1]$. Noting that if the learner chose to sample the interval C_i then

$$\min_{x\in C_i=[x_i,x_{i+1}]} \sup_{y\in\mathcal{F}_\mathcal{D}} e_{D=[0,1]}(\mathcal{H},\mathcal{D}\cup(x,y),\mathcal{F}) =$$

$$\left(\sum_{j=0,j\neq i}^{n} e^p_{C_j}(\mathcal{H},\mathcal{D},\mathcal{F}) + \frac{e^p_{C_i}(\mathcal{H},\mathcal{D},\mathcal{F})}{2}\right)^{1/p}$$

From the decomposition above, it is clear that the optimal point to sample according to the principle embodied in Algorithm B is the midpoint of the interval C_j which has the maximum uncertainty $e_{C_j}(\mathcal{H},\mathcal{D},\mathcal{F})$ on the basis of the data seen so far, i.e., the data set $\mathcal{D}$. Thus we can state the following theorem

Theorem 2 *The CLA is the optimal algorithm for the class of monotonic functions*

Having thus established that our binary searching algorithm (CLA) is optimal, we now turn our efforts to determining the number of examples the CLA would need in order to learn the unknown target function to ϵ accuracy with δ confidence. In particular, we can prove the following theorem.

Theorem 3 *The CLA converges in at most $(M/\epsilon)^p$ steps. Specifically, after collecting at most $(M/\epsilon)^p$ examples, its hypothesis is ϵ close to the target with probability 1.*

Proof Sketch: The proof of convergence for this algorithm is a little tedious. However, to convince the reader, we provide the proof of convergence for a slight variant of the active algorithm. It is possible to show (not shown here) that convergence times for the active algorithm described earlier is bounded by the convergence time for the variant. First, consider a uniform grid of points $(\epsilon/M)^p$ apart on the domain $[0,1]$. Now imagine that the active learner works just as described earlier but with a slight twist, viz., it can only query points on this grid. Thus at the kth stage, instead of querying the true midpoint of the interval with largest uncertainty, it will query the gridpoint closest to this midpoint. Obviously the intervals at the kth stage are also separated by points on the grid (i.e. previous queries). If it is the case that the learner has queried all the points on the grid, then the maximum possible error it could make is less than ϵ. To see this, let $\alpha = \epsilon/M$ and let us first look at a specific small interval $[k\alpha, (k+1)\alpha]$. We know the following to be true for this subinterval:

$$f(k\alpha) = h(k\alpha) \leq f(x), h(x) \leq f((k+1)\alpha) = h((k+1)\alpha)$$

Thus

$$|f(x) - h(x)| \leq f((k+1)\alpha) - f(k\alpha)$$

and so over the interval $[k\alpha, (k+1)\alpha]$

$$\int_{k\alpha}^{(k+1)\alpha} |f(x) - h(x)|^p dx \leq$$

$$\int_{k\alpha}^{(k+1)\alpha} (f((k+1)\alpha) - f(k\alpha))^p dx$$

$$\leq (f((k+1)\alpha) - f(k\alpha))^p \alpha$$

It follows that

$$\int_{[0,1]} |f-h|^p dx =$$

$$\int_{[0,\alpha)} |f-h|^p dx + \ldots + \int_{[1-\alpha,1]} |f-h|^p dx \leq$$

$$\alpha((f(\alpha) - f(0))^p + (f(2\alpha) - f(\alpha))^p + \ldots +$$

$$(f(1-\alpha) - f(1-2\alpha))^p + (f(1) - f(1-\alpha))^p) \leq$$

$$\alpha(f(\alpha) - f(0) + f(2\alpha) - f(\alpha) + \ldots + f(1) - f(1-\alpha))^p$$

$$\leq \alpha(f(1) - f(0))^p \leq \alpha M^p$$

So if $\alpha = (\epsilon/M)^p$, we see that the L_p error would be at most $\left(\int_{[0,1]} |f-h|^p dx\right)^{1/p} \leq \epsilon$. Thus the active learner moves from stage to stage collecting examples at the grid points. It could converge at any stage, but clearly after it has seen the value of the unknown target at all the gridpoints, its error is provably less than ϵ and consequently it must stop by this time. □

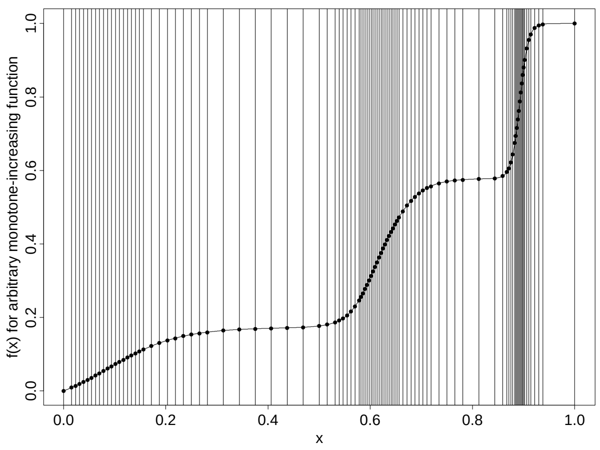

Figure 5: How the CLA chooses its examples. Vertical lines have been drawn to mark the x-coordinates of the points at which the algorithm asks for the value of the function.

3.3 Empirical Simulations, and other Investigations

Our aim here is to characterise the performance of CLA as an active learning strategy. Remember that CLA is an adaptive example choosing strategy and the number of samples it would take to converge depends upon the specific nature of the target function. We have already computed an upper bound on the number of samples it would take to converge in the worst case. In this section we try to provide some intuition as to how this sampling strategy differs from random draw of points (equivalent to passive learning) or drawing points on a uniform grid (equivalent to optimal recovery following eq. 2 as we shall see shortly). We perform simulations on arbitrary monotonic increasing functions to better characterize conditions under which the active strategy could outperform both a passive learner as well as a uniform learner.

3.3.1 Distribution of Points Selected

As has been mentioned earlier, the points selected by CLA depend upon the specific target function.Shown in fig. 3-5 is the performance of the algorithm for an arbitrarily constructed monotonically increasing function. Notice the manner in which it chooses its examples. Informally speaking, in regions where the function changes a lot (such regions can be considered to have high information density and consequently more "interesting"), CLA samples densely. In regions where the function doesn't change much (correspondingly low information density), it samples sparsely. As a matter of fact, the density of the points seems to follow the derivative of the target function as shown in fig. 6.

Consequently, we conjecture that

Conjecture 1 *The density of points sampled by the active learning algorithm is proportional to the derivative of the function at that point for differentiable functions.*

Remarks:

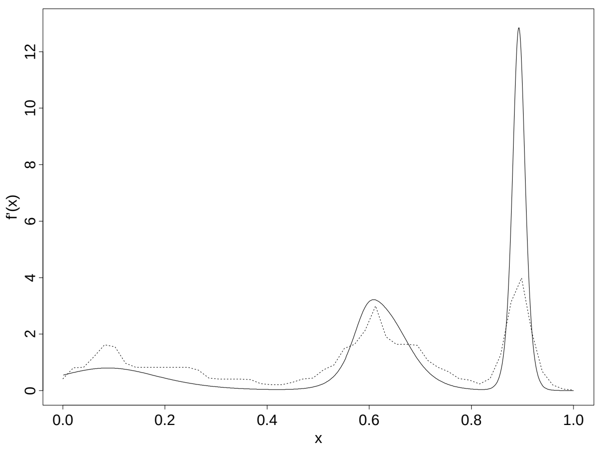

Figure 6: The dotted line shows the density of the samples along the x-axis when the target was the monotone-function of the previous example. The bold line is a plot of the derivative of the function. Notice the correlation between the two.

1. The CLA seems to sample functions according to its rate of change over the different regions. We have remarked earlier, that the best possible sampling strategy would be obtained by eq. 3 earlier. This corresponds to a teacher (who knows the target function and the learner) selecting points for the learner. How does the CLA sampling strategy differ from the best possible one? Does the sampling strategy converge to the best possible one as the data goes to infinity? In other words, does the CLA discover the best strategy? These are interesting questions. We do not know the answer.
2. We remarked earlier that another bound on the performance of the active strategy was that provided by the classical optimal recovery formulation of eq. 2. This, as we shall show in the next section, is equivalent to uniform sampling. We remind the reader that a crucial difference between uniform sampling and CLA lies in the fact that CLA is an adaptive strategy and for some functions might actually learn with very few examples. We will explore this difference soon.

3.3.2 Classical Optimal Recovery

For an L_1 error criterion, classical optimal recovery as given by eq. 2 yields a uniform sampling strategy. To see this, imagine that we chose to sample the function at points $x_i; i = 1, \ldots, n$. Pick a possible target function f and let $y_i = f(x_i)$ for each i. We then get the situation depicted in fig. 7. The n points divide the domain into $n+1$ intervals. Let these intervals have length a_i each as shown. Further, if $[x_{i-1}, x_i]$ corresponds to the interval of length a_i, then let $y_i - y_{i-1} = b_i$. In other words we would get $n+1$ rectangles with sides a_i and b_i as shown in the figure.

It is clear that choosing a vector $\mathbf{b} = (b_1, \ldots, b_{n+1})'$

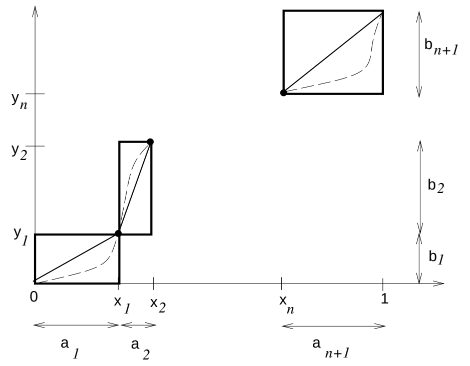

Figure 7: The situation when a function $f \in \mathcal{F}$ is picked, n sample points (the x's) are chosen and the corresponding y values are obtained. Each choice of sample points corresponds to a choice of the a's. Each choice of a function corresponds to a choice of the b's.

with the constraint that $\sum_{i=1}^{n+1} b_i = M$ and $b_i \geq 0$ is equivalent to defining a set of y values (in other words, a data set) which can be generated by some function in the class $\mathcal{F}$. Specifically, the data values at the respective sample points would be given by $y_1 = b_1$, $y_2 = b_1 + b_2$ and so on. We can define $\mathcal{F}_{\mathbf{b}}$ to be the set of monotonic functions in $\mathcal{F}$ which are consistent with these data points. In fact, every $f \in \mathcal{F}$ would map onto some $\mathbf{b}$, and thus belong to some $\mathcal{F}_{\mathbf{b}}$. Consequently,

$$\mathcal{F} = \cup_{\{\mathbf{b}: b_i \geq 0, \sum b_i = M\}} \mathcal{F}_{\mathbf{b}}$$

Given a target function $f \in \mathcal{F}_{\mathbf{b}}$, and a choice of n points x_i, one can construct the data set $\mathcal{D} = \{(x_i, f(x_i))\}_{i=1\ldots n}$ and the approximation scheme generates an approximating function $h(\mathcal{D})$. It should be clear that for an L_1 distance metric ($d(f,h) = \int_0^1 |f - h| dx$), the following is true:

$$\sup_{f \in \mathcal{F}_{\mathbf{b}}} d(f,h) = \frac{1}{2} \sum_{i=1}^{n+1} a_i b_i = \frac{1}{2} \mathbf{a}.\mathbf{b}$$

Thus, taking the supremum over the entire class of functions is equivalent to

$$\sup_{f \in \mathcal{F}} d(f, h(\mathcal{D})) = \sup_{\{\mathbf{b}: b_i \geq 0, \sum b_i = M\}} \frac{1}{2} \mathbf{a}.\mathbf{b}$$

The above is a straight forward linear programming problem and yields as its solution the result $\frac{1}{2} M \max\{a_i, i = 1, \ldots, (n+1)\}$.

Finally, every choice of n points $x_i, i = 1, \ldots, n$ results in a corresponding vector $\mathbf{a}$ where $a_i \geq 0$ and $\sum a_i = 1$. Thus minimizing the maximum error over all the choice of sample points (according to eq. 2) is equivalent to

$$\arg\min_{x_1,\ldots,x_n} \sup_{f \in \mathcal{F}} d(f, h(\mathcal{D} = \{(x_i, f(x_i))\}_{i=1\ldots n}) =$$

$$\arg\min_{\{\mathbf{a}: a_i \geq 0, \sum a_i = 1\}} \max\{a_i; i = 1 \ldots n+1\}$$

Clearly the solution of the above problem is $a_i = \frac{i}{n+1}$ for each i.

In other words, classical optimal recovery suggests that one should sample the function uniformly. Note that this is not an adaptive scheme. In the next section, we compare empirically the performance of three different schemes to sample. The passive, where one samples randomly, the non-sequential "optimal", where one samples uniformly, and the active which follows our sequentially optimal strategy.

3.3.3 Error Rates and Sample Complexities for some Arbitrary Functions: Some Simulations

In this section, we attempt to relate the number of examples drawn and error made by the learner for a variety of arbitrary monotone increasing functions. We begin with the following simulation:

Simulation A:

1. Pick an arbitrary monotone-increasing function.
2. Decide (N), the number of samples to be collected. There are three methods of collection of samples. The first is by randomly drawing N examples according to a uniform distribution on $[0, 1]$ (corresponding to the passive case). The second is by asking for function values on a uniform grid on $[0, 1]$ of grid spacing $1/N$. The third is the CLA.
3. The three learning algorithms differ only in their method of obtaining samples. Once the samples are obtained, all three algorithms attempt to approximate the target by the monotone function which is the linear interpolant of the samples.
4. This entire process is now repeated for various values of N for the same target function and then repeated again for different target functions.

Results: Let us first consider performance on the arbitrarily selected monotonic function of the earlier section. Shown in fig. 8 are performance for the three different algorithms. Notice that the active learning strategy (CLA) has the lowest error rate. On an average, the error rate of random sampling is 8 times the rate of CLA and uniform sampling is 1.5 times the rate of CLA.

Figure 9 shows four other monotonic functions on which we ran the same simulations comparing the three sampling strategies. The results of the simulations are shown in Fig. 10 and Table 3.3.3. Notice that the active strategy (CLA) far outperforms the passive strategy and clearly has the best error performance. The comparison between uniform sampling and active sampling is more interesting. For functions like function-2 (which is a smooth approximation of a step function), where most of the "information" is located in a small region of the domain, CLA outperforms the uniform learner by a large amount. Functions like function-3 which don't have any clearly identified region of greater information have the least difference between CLA and the uniform learner (as also between the passive and active learner). Finally on functions which lie in between these two extremes (like functions 4 and 5) we see decreased error-rates due to CLA which are in between the two extremes.

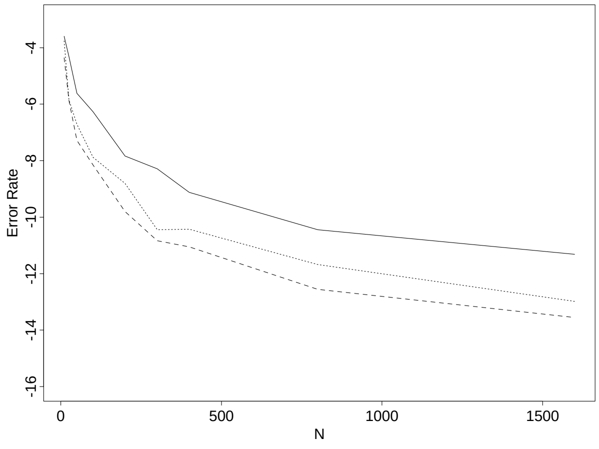

Figure 8: Error rates as a function of the number of examples for the arbitrary monotone function shown in a previous figure.

In conclusion, the active learner outperforms the passive learner. Further, it is even better than classical optimal recovery. The significant advantage of the active learner lies in its adaptive nature. Thus, for certain "easy" functions, it might converge very rapidly. For others, it might take as long as classical optimal recovery, though never more.

4 Example 2: A Class of Functions with Bounded First Derivative

Here the class of functions we consider are from $[0, 1]$ to R and of the form

$$\mathcal{F} = \{f | f(x) \text{ is differentiable and } |\frac{df}{dx}| \leq d\}$$

Notice a few things about this class. First, there is no direct bound on the values that functions in $\mathcal{F}$ can take. In other words, for every $M > 0$, there exists some function $f \in \mathcal{F}$ such that $f(x) > M$ for some $x \in [0, 1]$. However, there is a bound on the first derivative, which means that a particular function belonging to $\mathcal{F}$ cannot itself change very sharply. Knowing the value of the function at any point, we can bound the value of the function at all other points. So for example, for every $f \in \mathcal{F}$, we see that $|f(x)| \leq dxf(0) \leq df(0)$.

We observe that this class too has infinite pseudo-dimension. We state this without proof.

Observation 3 *The class $\mathcal{F}$ has infinite pseudo-dimension in the sense of Pollard.*

As in the previous example we would like to investigate the possibility of devising active learning strategies for this class. We first provide a lower bound on the number of examples a learner (whether passive or active) would need in order to ϵ identify this class. We then derive in the next section, an optimal active learning strategy (that is, an instantiation of the Active Algorithm B earlier). We also provide an upper bound on the number of examples this active algorithm would take.

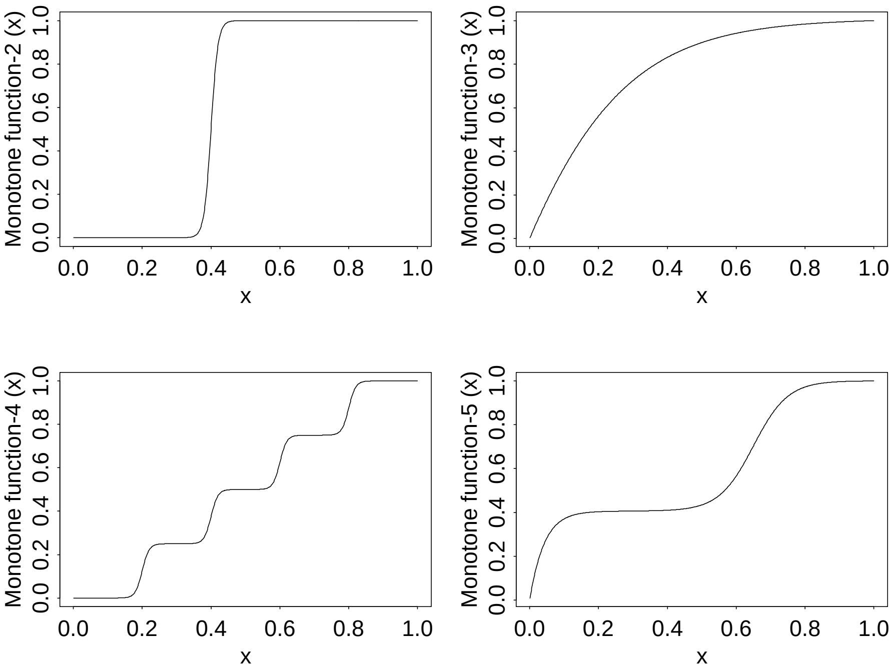

Figure 9: Four other monotonic functions on which simulations have been run comparing random, uniform, and active sampling strategies.

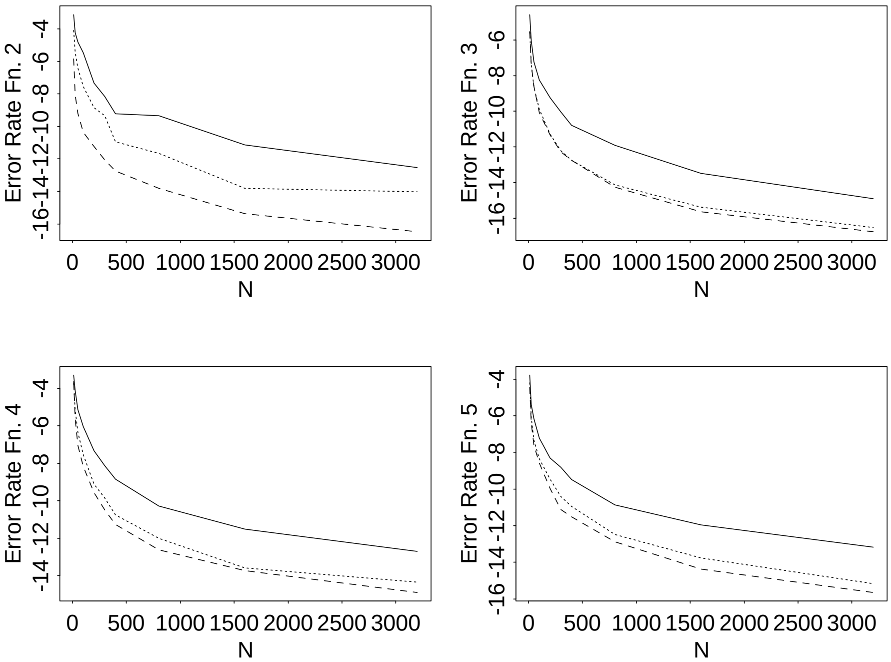

Figure 10: This figure plots the log of the error (L_1 error) against N the number of examples for each of the 4 monotonic functions shown in fig. 9. The solid line represents error rates for random sampling, the line with small dashes represents uniform sampling and the line with long dashes represents results for CLA. Notice how CLA beats random sampling by large amounts and does slightly better than uniform sampling.

Fn. No.	Avg. Random/CLA	Avg. Uniform/CLA
1	7.23	1.66
2	61.37	10.91
3	6.67	1.10
4	8.07	1.62
5	6.62	1.56

Table 1: Shown in this table is the average error rate of the random sampling and the uniform sampling strategies when as a multiple of the error rates due to CLA. Thus for the function 3 for example, uniform error rates are on an average 1.1 times CLA error rates. The averages are taken over the different values of N (number of examples) for which the simulations have been done. Note that this is not a very meaningful average as the difference in the error rates between the various strategies grow with N (as can be seen from the curves)if there is a difference in the order of the sample complexity. However they have been provided just to give a feel for the numbers.

We also need to specify some other terms for this class of functions. The approximation scheme $\mathcal{H}$ is a first order spline as before, the domain $D = [0,1]$ is partitioned into intervals by the data $[x_i, x_{i+1}]$ (again as before) and the metric d_C is an L_1 metric given by $d_C(f_1, f_2) = \int_C |f_1(x) - f_2(x)|dx$. The results in this section can be extended to an L_p norm but we confine ourselves to an L_1 metric for simplicity of presentation.

4.1 Lower Bounds

Theorem 4 *Any learning algorithm (whether passive or active) has to draw at least $\Omega((d/\epsilon))$ examples (whether randomly or by choosing) in order to PAC learn the class $\mathcal{F}$.*

Proof Sketch: Let us assume that the learner collects m examples (passively by drawing according to some distribution, or actively by any other means). Now we show that an adversary can force the learner to make an error of atleast ϵ if it draws less than $\Omega((d/\epsilon))$ examples. This is how the adversary functions.

At each of the m points which are collected by the learner, the adversary claims the function has value 0. Thus the learner is reduced to coming up with a hypothesis that belongs to $\mathcal{F}$ and which it claims will be within an ϵ of the target function. Now we need to show that whatever the function hypothesized by the learner, the adversary can always come up with some other function, also belonging to $\mathcal{F}$, and agreeing with all the data points, which is more than an ϵ distance away from the learner's hypothesis. In this way, the learner will be forced to make an error greater than ϵ.

The m points drawn by the learner, divides the region $[0,1]$ into (at most) $m+1$ different intervals. Let the length of these intervals be $b_1, b_2, b_3, ..., b_{m+1}$. The "true" function, or in other words, the function the adversary will present, should have value 0 at the endpoints of each of the above intervals. We first state the following lemma.

Lemma 2 *There exists a function $f \in \mathcal{F}$ such that f interpolates the data and*

$$\int_{[0,1]} |f|dx > \frac{kd}{4(m+1)}$$

where k is a constant arbitrarily close to 1.

Proof: Consider fig. 11. The function f is indicated by the dark line. As is shown, f changes sign at each $x = x_i$. Without loss of generality, we consider an interval $[x_i, x_{i+1}]$ of length b_i. Let the midpoint of this interval be $z = (x_i + x_{i+1})/2$. The function here has the values

$$f(x) = \begin{cases} d(x - x_i) & \text{for } x \in [x_i, z-\alpha] \\ -d(x - x_{i+1}) & \text{for } x \in [z+\alpha, x_{i+1}] \\ \frac{d(x-z)^2}{2\alpha} + \frac{d(b_i - \alpha)}{2} & \text{for } x \in [z-\alpha, z+\alpha] \end{cases}$$

Simple algebra shows that

$$\int_{x_i}^{x_{i+1}} |f|dx > d(\frac{b_i - \alpha}{2})^2 + \alpha d(\frac{b_i - \alpha}{2}) = d(b_i^2 - \alpha^2)/4$$

Clearly, α can be chosen small, so that

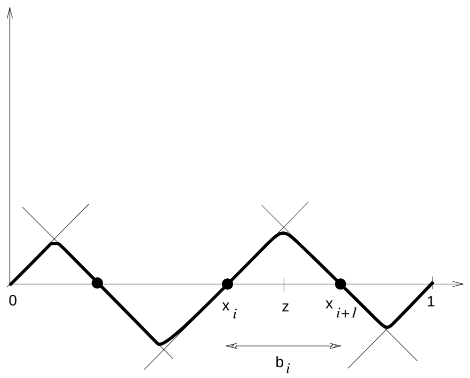

Figure 11: Construction of a function satisying Lemma 2.

$$\int_{x_i}^{x_{i+1}} |f|dx > \frac{kd{b_i}^2}{4}$$

where k is as close to 1 as we want. By combining the different pieces of the function we see that

$$\int_0^1 |f|dx > \frac{kd}{4} \sum_i^{m+1} b_i^2$$

Now we make use of the following lemma,

Lemma 3 *For a set of numbers $b_1, .., b_m$ such that $b_1 + b_2 + .. + b_m = 1$, the following inequality is true*

$$b_1^2 + b_2^2 + .. + b_m^2 \geq 1/m$$

Proof: By induction. □

Now it is easy to see how the adversary functions. Suppose the learner postulates that the true function is h. Let $\int_{[0,1]} |h|dx = \chi$. If $\chi > \epsilon$, the adversary claims that the true function was $f = 0$. In that case $\int_0^1 |h - f|dx =$

$\chi > \epsilon$. If on the other hand, $\chi < \epsilon$, then the adversary claims that the true function was f (as above). In that case,

$$\int_0^1 |f-h|dx \geq \int_0^1 |f|dx - \int_0^1 |h|dx = \frac{kd}{4(m+1)} - \chi$$

Clearly, if m is less than $\frac{kd}{8\epsilon} - 1$, the learner is forced again to make an error greater than ϵ. Thus in either case, the learner is forced to make an error greater than or equal to ϵ if less than $\Omega(d/\epsilon)$ examples are collected (howsoever these examples are collected). □

The previous result holds for all learning algorithms. It is possible to show the following result for a passive learner.

Theorem 5 *A Passive learner must draw at least* $\max(\Omega((d/\epsilon), \sqrt{(d/\epsilon)}\ln(1/\delta)))$ *to learn this class.*

Proof Sketch: The d/ϵ term in the lower bound follows directly from the previous theorem. We show how the second term is obtained.

Consider the uniform distribution on $[0,1]$ and a subclass of functions which have value 0 on the region A $= [0, 1-\alpha]$ and belong to $\mathcal{F}$. Suppose the passive learner draws l examples uniformly at random. Then with probability $(1-\alpha)^l$, all these examples will be drawn from region A. It only remains to show that for this event, and the subclass considered, whatever be the function hypothesized by the learner, an adversary can force it to make a large error.

It is easy to show (using the arguments of the earlier theorem) that there exists a function $f \in \mathcal{F}$ such that f is 0 on A and $\int_{1-\alpha}^1 |f|dx = \frac{1}{2}\alpha^2 d$. This is equal to 2ϵ if $\alpha = \sqrt{(4\epsilon/d)}$. Now let the learner's hypothesis be h. Let $\int_{1-\alpha}^1 |h|dx = \chi$. If χ is greater than ϵ, the adversary claims the target was $g = 0$. Otherwise, the adversary claims the target was $g = f$. In either case, $\int |g-h|dx > \epsilon$.

It is possible to show (by an identical argument to the proof of theorem 1), that unless $l \geq \frac{1}{4}\sqrt{(d/\epsilon)}\ln(1/\delta)$, all examples will be drawn from A with probability greater than δ and the learner will be forced to make an error greater than ϵ. Thus the second term appears indicating the dependence on δ in the lower bound.□

4.2 Active Learning Algorithms

We now derive in this section an algorithm which actively selects new examples on the basis of information gathered from previous examples. This illustrates how our formulation of section 3.1.1 can be used in this case to effectively obtain an optimal adaptive sampling strategy.

4.2.1 Derivation of an optimal sampling strategy

Fig. 12 shows an arbitrary data set containing information about some unknown target function. Since the target is known to have a first derivative bounded by d, it is clear that the target is constrained to lie within the parallelograms shown in the figure. The slopes of the lines making up the parallelogram are d and $-d$ appropriately. Thus, $\mathcal{F}_{\mathcal{D}}$ consists of all functions which lie within the parallelograms and interpolate the data set. We can now compute the uncertainty of the approximation scheme over any interval,C, (given by $e_C(\mathcal{H}, \mathcal{D}, \mathcal{F})$), for this case. Recall that the approximation scheme $\mathcal{H}$ is a first order spline, and the data $\mathcal{D}$ consists of (x, y) pairs. Fig. 13 shows the situation for a particular interval ($C_i = [x_i, x_{i+1}]$). Here i ranges from 0 to n. As in the previous example, we let $x_0 = 0$, and $x_{n+1} = 1$.

The maximum error the approximation scheme $\mathcal{H}$ could have on this interval is given by (half the area of the parallelogram).

$$e_{C_i}(\mathcal{H}, \mathcal{D}, \mathcal{F}) = \sup_{f \in \mathcal{F}_{\mathcal{D}}} \int_{C_i} |h-f|dx = \frac{(d^2 B_i^2 - A_i^2)}{4d}$$

where $A_i = |f(x_{i+1}) - f(x_i)|$ and $B_i = x_{i+1} - x_i$. Clearly, the maximum error the approximation scheme could have over the entire domain is given by

$$e_{D=[0,1]}(\mathcal{H}, \mathcal{D}, \mathcal{F}) = \sup_{f \in \mathcal{F}_{\mathcal{D}}} \sum_{j=0}^{n} \int_{C_j} |f-h|dx = \sum_{j=0}^{n} e_{C_j} \quad (5)$$

The computation of e_C is crucial to the derivation of the active sampling strategy. Now imagine that we chose to sample at a point x in the interval C_i and received a value y (belonging to $\mathcal{F}_{\mathcal{D}}(x)$). This adds one more interval and divides C_i into two intervals C_{i1} and C_{i2} as shown in fig. 14.. We also obtain two correspondingly smaller parallelograms within which the target function is now constrained to lie.

The addition of this new data point to the data set ($\mathcal{D}' = \mathcal{D} \cup (x, y)$) requires us to recompute the learner's hypothesis (denoted by h' in the fig. 14). Correspondingly, it also requires us to update e_C, i.e., we now need to compute $e_C(\mathcal{H}, \mathcal{D}', \mathcal{F})$. First we observe that the addition of the new data point does not affect the uncertainty measure on any interval other than the divided interval C_i. This is clear when we notice that the parallelograms (whose area denotes the uncertainty on each interval) for all the other intervals are unaffected by the new data point.

Thus,

$$e_{C_j}(\mathcal{H}, \mathcal{D}', \mathcal{F}) = e_{C_j}(\mathcal{H}, \mathcal{D}, \mathcal{F}) = \frac{1}{4d}(d^2 B_j^2 - A_j^2) \text{ for } j \neq i$$

For the ith interval C_i, the total uncertainty is now recomputed as (half the sum of the two parallelograms in fig. 14)

$$\begin{aligned} e_{C_i}(\mathcal{H}, \mathcal{D}', \mathcal{F}) &= \tfrac{1}{4d}((d^2u^2 - v^2) + \\ &(d^2(B_i - u)^2 - (A_i - v)^2)) \quad (6) \\ &= \tfrac{1}{4d}\left((d^2u^2 + d^2(B_i-u)^2) - (v^2 + (A-v)^2)\right) \end{aligned}$$

where $u = x - x_i$, $v = y - y_i$, and A_i and B_i are as before. Note that u ranges from 0 to B_i, for $x_i \leq x \leq x_{i+1}$. However, given a particular choice of x (this fixes a value of u), the possible values v can take are constrained by the geometry of the parallelogram. In particular, v can only lie within the parallelogram. For a particular x, we know that $\mathcal{F}_{\mathcal{D}}(x)$ represents the set of all possible y values we can receive. Since $v = y - y_i$, it is clear that $v \in \mathcal{F}_{\mathcal{D}}(x) - y_i$. Naturally, if $y < y_i$, we find that $v < 0$, and $A_i - v > A_i$. Similarly, if $y > y_{i+1}$, we find that $v > A_i$.

We now prove the following lemma:

Lemma 4 *The following two identities are valid for the appropriate mini-max problem.*
Identity 1:

$$\frac{B}{2} = \arg\min_{u\in[0,B]} \sup_{v\in\{\mathcal{F}_{\mathcal{D}}(x)-y_i\}} H_1(u,v)$$

$$\text{where } H_1(u,v) = ((d^2u^2 + d^2(B-u)^2) -$$

$$(v^2 + (A-v)^2))$$

Identity 2:

$$\frac{1}{2}(d^2B^2 - A^2) = \min_{u\in[0,B]} \sup_{v\in\{\mathcal{F}_{\mathcal{D}}(x)-y_i\}} H_2(u,v)$$

$$\text{where } H_2(u,v) = ((d^2u^2 + d^2(B-u)^2) -$$

$$(v^2 + (A-v)^2))$$

Proof: The expression on the right is a difference of two quadratic expressions and can be expressed as $q_1(u) - q_2(v)$. For a particular u, the expression is maximized when the quadratic $q_2(v) = (v^2 + (A-v)^2)$ is minimized. Observe that this quadratic is globally minimized at $v = A/2$. We need to perform this minimization over the set $v \in \mathcal{F}_{\mathcal{D}}(x) - y_i$ (this is the set of values which lie within the upper and lower boundaries of the parallelogram shown in fig. 15). There are three cases to consider.

Case I: $u \in [A/2d, B - A/2d]$
First, notice that for u in this range, it is easy to verify that the upper boundary of the parallelogram is greater than $A/2$ while the lower boundary is less than $A/2$. Thus we can find a value of v (viz. v = A/2) which globally minimizes this quadratic because $A/2 \in \mathcal{F}_{\mathcal{D}}(x) - y_i$. The expression thus reduces to $d^2u^2 + d^2(B-u)^2 - A^2/2$. Over the interval for u considered in this case, it is minimized at $u = B/2$ resulting in the value

$$(d^2B^2 - A^2)/2$$

Case II: $u \in [0, A/2d]$
In this case, the upper boundary of the parallelogram (which is the maximum value v can take) is less than $A/2$ and hence the $q_2(v)$ is minimized when $v = du$. The total expression then reduces to

$$d^2u^2 + d^2(B-u)^2 - ((du)^2 + (A-du)^2)$$

$$= d^2(B-u)^2 - (A-du)^2 = (d^2B^2 - A^2) - 2ud(dB - A)$$

Since, $dB > A$, the above is minimized on this interval by choosing $u = A/2d$ resulting in the value

$$dB(dB - A)$$

Case III: By symmetry, this reduces to case II.

Since $(d^2B^2 - A^2)/2 \le dB(dB - A)$ (this is easily seen by completing squares), it follows that $u = B/2$ is the global solution of the mini-max problem above. Further, we have shown that for this value of u, the sup term reduces to $(d^2B^2 - A^2)/2$ and the lemma is proved.□

Using the above lemma along with eq. 6, we see that

$$\min_{x\in C_i} \sup_{y\in\mathcal{F}_{\mathcal{D}}(x)} e_{C_i}(\mathcal{H}, \mathcal{D}\cup(x,y), \mathcal{F}) = \frac{1}{8d}(d^2B_i^2 - A_i^2)$$

$$= \frac{1}{2} e_{C_i}(\mathcal{H}, \mathcal{D}, \mathcal{F})$$

In other words, by sampling the midpoint of the interval C_i, we are guaranteed to reduce the uncertainty by 1/2. As in the case of monotonic functions now, we see that using eq. 5, we should sample the midpoint of the interval with largest uncertainty $e_{C_i}(\mathcal{H}, \mathcal{D}, \mathcal{F})$ to obtain the global solution in accordance with the principle of Algorithm B of section 2.

This allows us to formally state an active learning algorithm which is optimal in the sense implied in our formulation.

The Choose and Learn Algorithm - 2 (CLA-2)

1. **[Initial Step]** Ask for values of the function at points $x = 0$ and $x = 1$. At this stage, the domain $D = [0,1]$ is composed of one interval only, viz., $C_1 = [0,1]$. Compute $e_{C_1} = \frac{1}{4d}\left(d^2 - |f(1) - f(0)|^2\right)$ and $e_D = e_{C_1}$. If $e_D < \epsilon$, stop and output the linear interpolant of the samples as the hypothesis, otherwise query the midpoint of the interval to get a partition of the domain into two subintervals $[0, 1/2)$ and $[1/2, 1]$.
2. **[General Update and Stopping Rule]** In general, at the kth stage, suppose that our partition of the interval $[0,1]$ is $[x_0 = 0, x_1)$,$[x_1, x_2), \ldots,$ $[x_{k-1}, x_k = 1]$. We compute the uncertainty $e_{C_i} = \frac{1}{4d}\left(d^2(x_i - x_{i-1})^2 - |y_i - y_{i-1}|^2\right)$ for each $i = 1, \ldots, k$. The midpoint of the interval with maximum e_{C_i} is queried for the next sample. The total error $e_D = \sum_{i=1}^{k} e_{C_i}$ is computed at each stage and the process is terminated when $e_D \le \epsilon$. Our hypothesis h at every stage is a linear interpolation of all the points sampled so far and our final hypothesis is obtained upon the termination of the whole process.

It is possible to show that the following upperbound exists on the number of examples CLA would take to learn the class of functions in consideration

Theorem 6 *The CLA-2 would PAC learn the class in at most $\frac{d}{4\epsilon} + 1$ examples.*

Proof Sketch: Following a strategy similar to the proof of Theorem 3, we show how a slight variant of CLA-2 would converge in at most $(d/4\epsilon + 1)$ examples. Imagine a grid of n points placed $1/(n-1)$ apart on the domain $D = [0,1]$ where the kth point is $k/(n-1)$ (for k going from 0 to $n-1$). The variant of the CLA-2 operates by confining its queries to points on this grid. Thus at the kth stage, instead of querying the midpoint of the interval with maximum uncertainty, it will query the gridpoint closest to this midpoint. Suppose it uses up all the gridpoints in this fashion, then there will be $n-1$ intervals and by our arguments above, we have seen that the maximum error on each interval is bounded by

$$\frac{1}{4d}(d^2(\frac{1}{n-1})^2 - |y_i - y_{i-1}|^2) \leq \frac{1}{4d}d^2(\frac{1}{n-1})^2$$

Since there are $n-1$ such intervals, the total error it could make is bounded by

$$(n-1)\frac{1}{4d}d^2(\frac{1}{n-1})^2 = \frac{1}{4d}(\frac{1}{n-1})$$

It is easy to show that for $n > d/4\epsilon + 1$, this maximum error is less than ϵ. Thus the learner need not collect any more than $d/4\epsilon + 1$ examples to learn the target function to within an ϵ accuracy. Note that the learner will have identified the target to ϵ accuracy with probability 1 (always) by following the strategy outlined in this variant of CLA-2. □

We now have both an upper and lower bound for PAC-learning the class (under a uniform distribution) with queries. Notice that here as well, the sample complexity of active learning does not depend upon the confidence parameter δ. Thus for δ arbitrarily small, the difference in sample complexities between passive and active learning becomes arbitrarily large with active learning requiring much fewer examples.

4.3 Some Simulations

We now provide some simulations conducted on arbitrary functions of the class of functions with bounded derivative (the class $\mathcal{F}$). Fig. 16 shows 4 arbitrary selected functions which were chosen to be the target function for the approximation scheme considered. In particular, we are interested in observing how the active strategy samples the target function for each case. Further, we are interested in comparing the active and passive techniques with respect to error rates for the same number of examples drawn. In this case, we have been unable to derive an analytical solution to the classical optimal recovery problem. Hence, we do not compare it as an alternative sampling strategy in our simulations.

4.3.1 Distribution of points selected

The active algorithm CLA-2 selects points adaptively on the basis of previous examples received. Thus the distribution of the sample points in the domain D of the function depends inherently upon the arbitrary target function. Consider for example, the distribution of points when the target function is chosen to be Function-1 of the set shown in fig. 16.

Notice (as shown in fig. 17) that the algorithm chooses to sample densely in places where the target is flat, and less densely where the function has a steep slope. As our mathematical analysis of the earlier section showed, this is well founded. Roughly speaking, if the function has the same value at x_i and x_{i+1}, then it could have a variety of values (wiggle a lot) within. However, if, $f(x_{i+1})$ is much greater (or less) than $f(x_i)$, then, in view of the bound, d, on how fast it can change, it would have had to increase (or decrease) steadily over the interval. In the second case, the rate of change of the function over the interval is high, there is less uncertainty in the values of the function within the interval, and consequently fewer samples are needed in between.

In example 1, for the case of monotone functions, we saw that the density of sample points was proportional to the first derivative of the target function. By contrast, in this example, the optimal strategy chooses to sample points in a way which is inversely proportional to the magnitude of the first derivative of the target function. Fig. 18 exemplifies this.

4.3.2 Error Rates:

In an attempt to relate the number of examples drawn and the error made by the learner, we performed the following simulation.

Simulation B:

1. Pick an arbitrary function from class $\mathcal{F}$.
2. Decide N, the number of samples to be collected. There are two methods of collection of samples. The first (*passive*) is by randomly drawing N examples according to a uniform distribution on $[0, 1]$. The second (*active*) is the CLA-2.
3. The two learning algorithms differ only in their method of obtaining samples. Once the samples are obtained, both algorithms attempt to approximate the target by the linear interpolant of the samples (first order splines).
4. This entire process is now repeated for various values of N for the same target function and then repeated again for the four different target functions of fig. 16

The results are shown in fig. 19. Notice how the active learner outperforms the passive learner. For the same number of examples, the active scheme having chosen its examples optimally by our algorithm makes less error.

We have obtained in theorem 6, an upper bound on the performance of the active learner. However, as we have already remarked earlier, the number of examples the active algorithm takes before stopping (i.e., outputting an ϵ-good approximation) varies and depends upon the nature of the target function. "Simple" functions are learned quickly, "difficult" functions are learned slowly. As a point of interest, we have shown in fig. 20, how the actual number of examples drawn varies with ϵ. In order to learn a target function to ϵ-accuracy, CLA-2 needs at most $n_{\max}(\epsilon) = d/4\epsilon + 1$ examples. However, for a particular target function, f, let the number of examples it actually requires be $n_f(\epsilon)$. We plot $\frac{n_f(\epsilon)}{n_{\max}(\epsilon)}$ as a function of ϵ. Notice, first, that this ratio is always much less than 1. In other words, the active learner stops before the worst case upper bound with a guaranteed ϵ-good hypothesis. This is the significant advantage of an adaptive sampling scheme. Recall that for uniform sampling (or classical optimal recovery even) we would have no choice but to ask for $d/4\epsilon$ examples to be sure of having an ϵ-good hypothesis. Further, notice that that as ϵ gets smaller, the ratio gets smaller. This suggests that for these functions, the sample complexity of the active learner is of a different order (smaller) than the worst case bound. Of course, there always exists some function in $\mathcal{F}$ which would force the active learner to perform at its worst case sample complexity level.

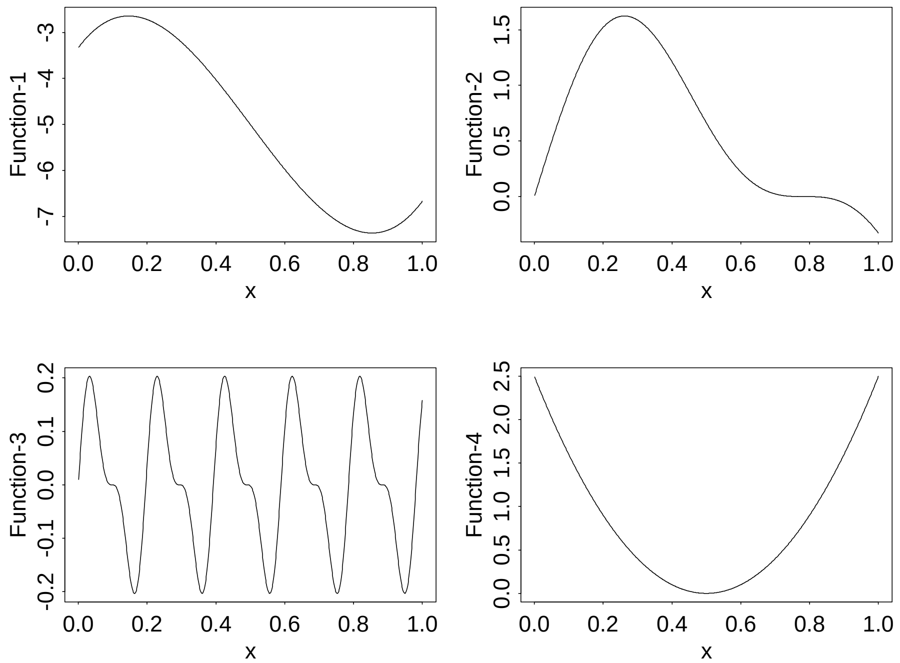

Figure 16: Four functions with bounded derivative considered in the simulations. The uniform bound on the derivative was chosen to be $d = 10$.

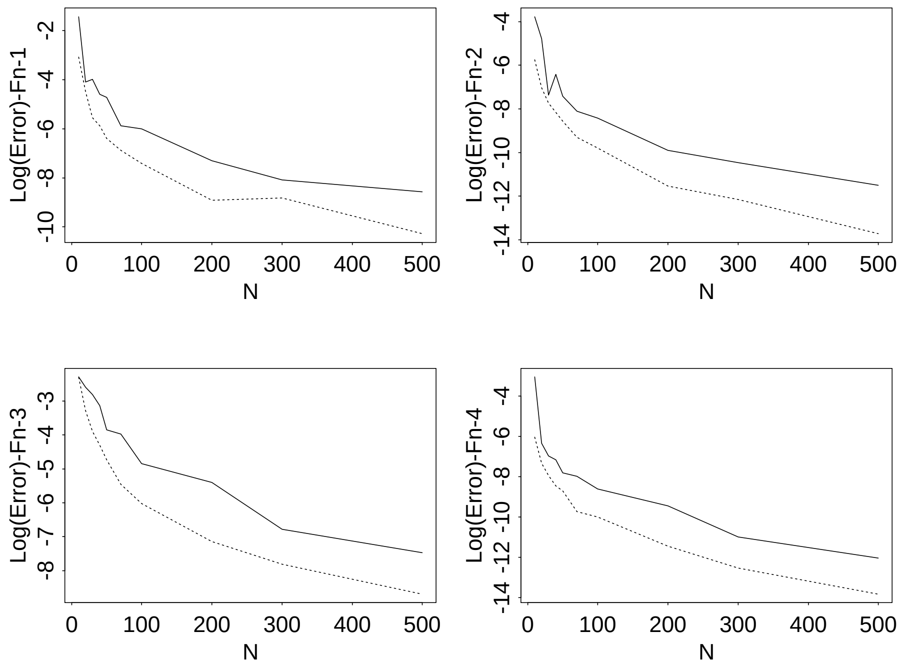

Figure 19: Results of Simulation B. Notice how the sampling strategy of the active learner causes better approximation (lower rates) for the same number of examples.

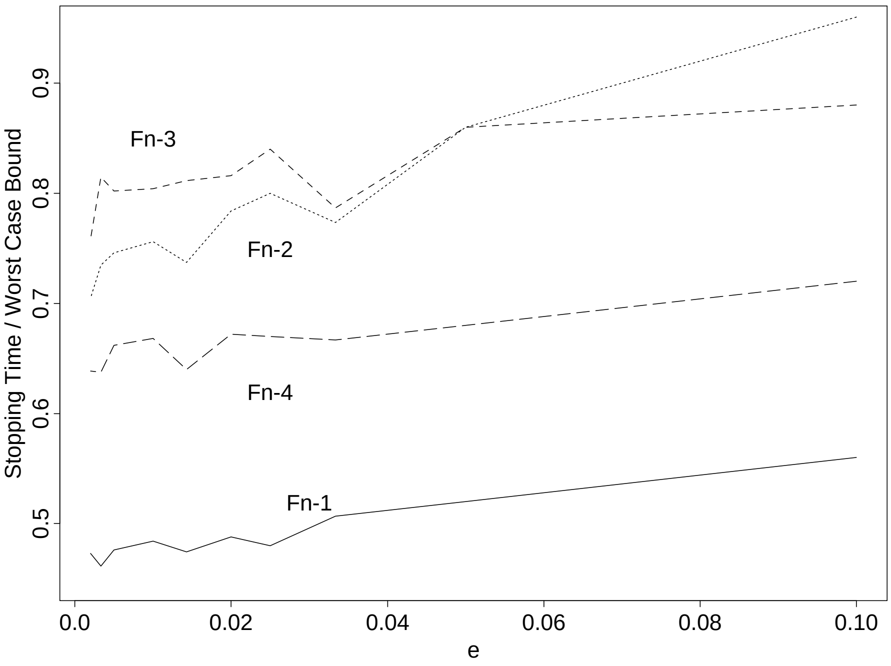

Figure 20: Variation with epsilons.

5 Conclusions, Extensions, and Open Problems

This paper focused on the possibility of devising *active* strategies to collect data for the problem of approximating real-valued function classes. We were able to derive a sequential version of optimal recovery. This sequential version, by virtue of using partial information about the target function is superior to classical optimal recovery. This provided us with a general formulation of an adaptive sampling strategy, which we then demonstrated on two example cases. Theoretical and empirical bounds on the sample complexity of passive and active learning for these cases suggest the superiority of the active scheme as far as the number of examples needed is concerned. It is worthwhile to observe that the same general framework gave rise to completely different sampling schemes in the two examples we considered. In one, the learner sampled densely in regions of high change. In the other, the learner did the precise reverse. This should lead us to further appreciate the fact that active sampling strategies are very task-dependent.

Using the same general formulation, we were also able to devise active strategies (again with superior sample complexity gain) for the following concept classes. 1) For the class of indicator functions $\{1_{[a,b]} : 0 < a < b < 1\}$ on the interval $[0,1]$, the sample complexity is reduced from $1/\epsilon \ln(1/\delta)$ for passive learning to $\ln(1/\epsilon)$ by adding membership queries. 2) For the class of half-spaces on a regular n-simplex, the sample complexity is reduced from $n/\epsilon \ln(1/\delta)$ to $n^2 \ln(s/\epsilon)$ by adding membership queries. Note that similar gains have been obtained for this class by Eisenberg (1992) using a different framework.

There are several directions for further research. First, one could consider the possibility of adding noise to our formulation of the problem. Noisy versions of optimal recovery exist and this might not be conceptually a very difficult problem. Although the general formulation (at least in the noise-free case) is complete, it might not be possible to compute the uncertainty bounds e_C for a variety of function classes. Without this, one could not actually use this paradigm to obtain a specific algorithm. A natural direction to pursue would be to investigate other classes (especially in more dimensions than 1) and other distance metrics to obtain further specific results. We observed that the active learning algorithm lay between classical optimal recovery and the optimal teacher. It would be interesting to compare the exact differences in a more principled way. In particular, an interesting open question is whether the sampling strategy of the active learner converges to that of the optimal teacher as more and more information becomes available. It would not be unreasonable to expect this, though precise results are lacking. In general, on the theme of better characterizing the conditions under which active learning would vastly outperform passive learning for function approximation, much work remains to be done. While active learning might require fewer examples to learn the target function, its computational burden is significantly larger. It is necessary to explore the information/computation trade-off with active learning schemes. Finally, we should note, that we have adopted in this part, a model of learning motivated by PAC but with a crucial difference. The distance metric, d, is not necessarily related to the distribution according to which data is drawn (in the passive case). This prevents us from using traditional uniform convergence (Vapnik, 1982) type arguments to prove learnability. The problem of learning under a different metric is an interesting one and merits further investigation in its own right.

References

[1] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

[2] M. L. Athavale and G. Wahba. Determination of an optimal mesh for a collocation-projection method for solving two-point boundary value problems. *Journal of Approximation Theory*, 25:38–49, 1979.

[3] R.L. Rivest B. Eisenberg. On the sample complexity of pac-learning using random and chosen examples. In *Proceedings of the 1990 Workshop on Computational LearningTheory*, pages 154–162, San Mateo, CA, 1990. Morgan Kaufmann.

[4] D. Cohn. Neural network exploration using optimal experiment design. AI memo 1491, Massachusetts Institute of Technology, 1994.

[5] B. Eisenberg. Sample complexity of active learning??? Master's thesis, Massachusetts Institute of Technology, Cambridge,MA, 1992.

[6] D. Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. Technical Report UCSC-CRL-91-02, University of California, Santa Cruz, 1989.

[7] T. O. Espelid J. Berntsen and A. Genz. An adaptive algorithm for the approximate calculation of multiple integrals. *ACM Transactions on Mathematical Software*, 17(4), December,1991.

[8] M. Kearns. Efficient noise-tolerant learning from statistical queries. In *Proceedings of the 1993 STOC*, pages 392–401, 1993.

[9] M. J. Kearns and R. E. Schapire. Efficient distribution-free learning of probabilistic concepts. In *Proceedings of the 1990 FOCS*, pages 382–391, 1990.

[10] D. Kimber and P. M. Long. The learning complexity of smooth functions of a single variable. In *Proceedings of the 1992 Workshop on Computational Learning Theory*, pages 153–159, San Mateo, CA, 1992. Morgan Kaufmann.

[11] S. Kulkarni, S.K. Mitter, and J.N. Tsitsiklis. Active learning using arbitrary binary valued queries. *Machine Learning*, (to appear).

[12] D. Mackay. *Bayesian Methods for Adaptive Models.* PhD thesis, California Institute of Technology, Pasadena, CA, 1991.

[13] C. A. Micchelli and G. Wahba. Design problems for optimal surface interpolation. In Z. Ziegler, editor, *Approximation theory and applications*, pages 329–348. Academic Press, New York, 1981.

[14] C.A. Micchelli and T.J. Rivlin. A survey of optimal recovery. In C.A. Micchelli and T.J. Rivlin, editors, *Optimal Estimation in Approximation Theory*, pages 1–54. Plenum Press, New York, 1976.

[15] P. M. Long N. Littlestone and M. K. Warmuth. Online learning of linear functions. In *Proceedings of the 1991 STOC*, pages 465–475, 1991.

[16] D. Pollard. *Convergence of stochastic processes.* Springer-Verlag, Berlin, 1984.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Parallel Distributed Processing.* MIT Press, Cambridge, MA, 1986.

[18] K. K. Sung and P. Niyogi. An active formulation for approximation of real valued functions. In *Advances in Neural information processings systems 7 (to appear)*, 1994.

[19] Wasilkowski G. W. Traub, J. F. and H. Wozniakovski. *Information Based Complexity.* Academic Press, New York, 1988.

[20] L.G. Valiant. A theory of learnable. *Proc. of the 1984 STOC*, pages 436–445, 1984.

[21] V. N. Vapnik. *Estimation of Dependences Based on Empirical Data.* Springer-Verlag, Berlin, 1982.

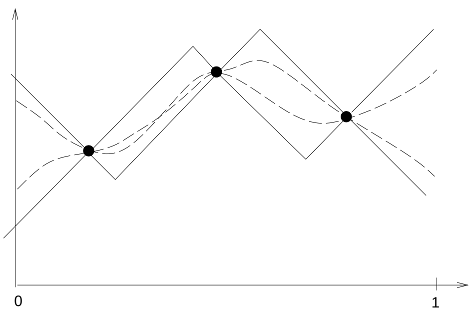

Figure 12: An arbitrary data set for the case of functions with a bounded derivative. The functions in $\mathcal{F}_\mathcal{D}$ are constrained to lie in the parallelograms as shown. The slopes of the lines making up the parallelogram are d and $-d$ appropriately.

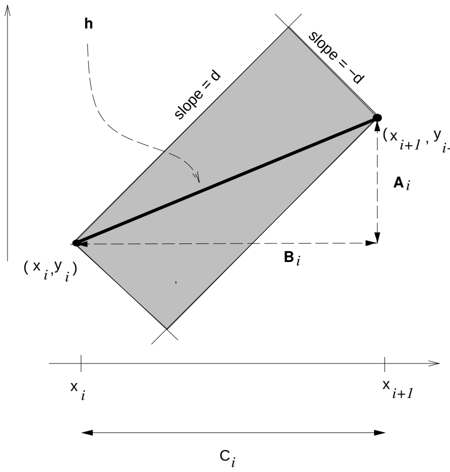

Figure 13: A zoomed version of the ith interval.

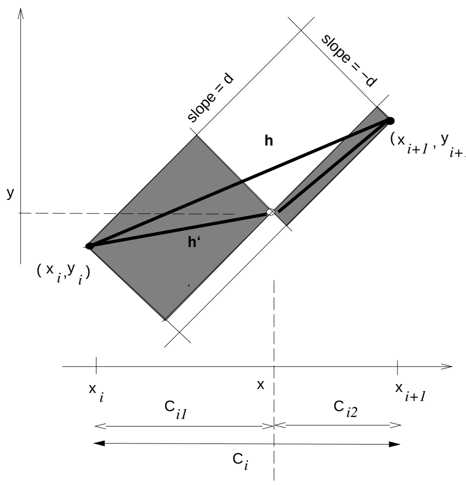

Figure 14: Subdivision of the ith interval when a new data point is obtained.

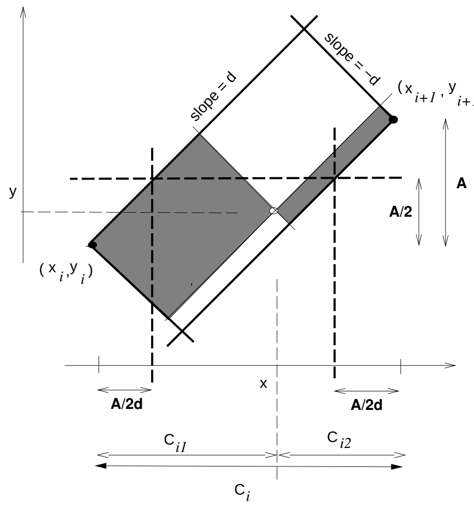

Figure 15: A figure to help the visualization of Lemma 4. For the x shown, the set $\mathcal{F}_\mathcal{D}$ is the set of all values which lie within the parallelogram corresponding to this x, i.e., on the vertical line drawn at x but within the parallelogram.

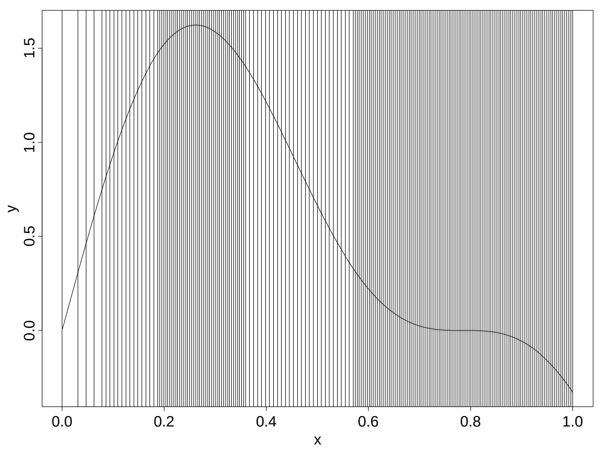

Figure 17: How CLA-2 chooses to sample its points. Vertical lines have been drawn at the x values where the CLA queried the oracle for the corresponding function value.

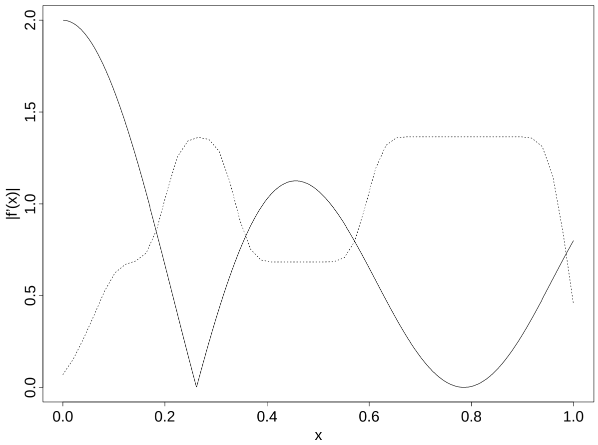

Figure 18: How CLA-2 chooses to sample its points. The solid line is a plot of $|f'(x)|$ where f is Function-1 of our simulation set. The dotted line shows the density of sample points (queried by CLA-2) on the domain.