Algorithms for Learning by Distances

Ronit Fraiman Shai Ben-David

July 23, 2001

Abstract

We consider the information complexity of learning in metric spaces. We discuss two models of such learning processes. The first one is the Learning By Distances (LBD) model of Ben-David et al [BIK]. In this model a concept is a point in a metric space, at each step of the learning process the student offers a hypothesis and receives from the teacher an approximation of its distance to the target. We also present a new *Relative Distances (RD)* model. In this model, at each step, the student presents two points and receives a bit indicating which of them is closer to the target.

We investigate the learning complexity in both models. We provide general lower and upper bounds on the complexity of learning concept classes in these models. We then analyze the complexity of several natural concept classes in two metric spaces; the space of boolean formulas with the metric induced by the number of satisfying assignments and spaces defined on graphs with the metric induced by the length of the shortest path between pairs of nodes.

1 Introduction

We investigate the complexity of learning in models of *learning by distances*. The *learning by distances* model was introduced in Ben-David et al. [BIK]. In this model concepts are just points in a metric space and the information conveyed to the student is in the form of (approximate) distances from certain points in the space to the target concept.

[BIK] concentrate on the scenarios where the interaction between the student and the world (or teacher) is through queries. The student asks about the distances of points of its choice to the target. They characterize the concept classes that are learnable in this model, and compare this model to more familiar models (PAC and Membership-Queries).

We take the next step and investigate the parameters that govern the complexity of the learning process in this model. We provide general lower and upper bounds on the information complexity of learning concept classes in this model (section 3). We then go on to investigate learnability of specific classes in two types of metric spaces. In section 4 we consider the space of boolean formulas with the metric induced by the number of satisfying assignments. We prove that the class of symmetric functions and the class of Majority functions are efficiently learnable in the LBD model.

In section 5 we endow graphs with the metric induced by the length of the shortest path between pairs of nodes. We introduce a general strategy for learning a vertex in such a space and apply this method to several natural families of graphs.

We also present a new, closely related, model - the *Relative Distances (RD)* model. In the *RD* model the student presents , at each step , a pair of points (from the underlying metric space) and the information he receives is a bit telling him which of the two is closer to the (unknown) target. In reality there are various situations in which it is much easier to acquire such relative information than to find actual distances. For an example, when trying to focus an optical device, say a camera lens, one has only a vague idea concerning the 'distance' of a current lens position to the optimum. Just the same we do 'learn' the correct focal distance through *comparing* the distances of close positions to the 'target' as reflected by the quality of the views they give rise to.

We conclude the paper with a comparison of these two models. In both models the concepts are points in a metric space, the student initiates the queries, and the teacher's responses are based on the distance from the target. The only difference is the kind of information supplied by the teacher; distances versus. relative distances.

It turns out, that in bounded metric spaces the same classes are learnable in both models. We also translate the upper and lower bounds achieved for LBD model to the RD model.

2 Preliminaries

In this section we summarize the [BIK] definitions of the learning-by-distances model.
Let (X, d) be a metric space. A *concept class* C is a subset of X. δ is a precision parameter for the teacher's answers and ϵ defines the accuracy required from the student.
Given ϵ, δ and a target $t \in C$, an (ϵ, δ)-*learning sequence* for t is a sequence $(< c_1, r(c_1) > \cdots < c_k, r(c_k) >)$ where $c_i \in X$ and $r(c_i)$ is an approximation of the distance between c_i and t, i.e., $d(c_i, t) - \delta \leq r(c_i) \leq d(c_i, t) + \delta$. In addition , if $d(c_i, t) \leq \epsilon$ then $r(c_i) = SUCCESS$. A student is a function from such finite sequences to points in the space - the next query. Learning sequences are generated by an interaction between a student and a teacher, the c_i *'s* are the student's queries and the $r(c_i)$*'s* are the teacher's responses.
A finite learning sequence is *successful* if it contains a pair of the form $< c, SUCCESS >$.
A concept class C is (ϵ, δ)-*learnable* if there exists a student S, and $l \in \mathcal{N}$ $(l(\epsilon, \delta, C))$ such that for every target $t \in C$, every (ϵ, δ)-learning sequence for t of length l is successful.
A concept class C is ϵ-*learnable* if it is (ϵ, δ)-learnable for all $\delta \leq \epsilon/2$.
A concept class C is $LBD-$*learnable* if it is ϵ-learnable for all $\epsilon > 0$.

2.1 Our Measures of Complexity

When we analyze a learning model, two questions immediately arise; which concept classes are learnable and how difficult is it to learn them?

[BIK] presented a complete characterization of the LBD learnable classes. In this paper we investigate the second question. There are several aspects to the complexity of learning.

We consider the communication complexity (i.e. the number of student queries and teacher responses) and the computational complexity of the student (resources needed during the learning process).

While analyzing the time complexity, we count every query and it's following answer as one step. Consequently, the time complexity of a learning algorithm is an upper bound on its information complexity.

We present our upper bounds in terms of time complexity, and our lower bounds in terms of information complexity. Hence , both kinds of bounds are valid for these two measures of learning complexity.

3 General Bounds on the Number of Queries

[BIK] presented lower and upper bounds for the information complexity, in terms of Kolmogorov's $\epsilon-$entropy of a metric space. We quote their bounds and present a new lower bound on the number of queries needed for successful learning.
The following notion of dimension is essential for analyzing learnability [KT].

Definition 1 *A set $A \subseteq X$ is an $\epsilon-$approximation of a set $B \subseteq C$ if for every $b \in B$ there exists $a \in A$ such that $d(a,b) \leq \epsilon$. The $\epsilon-$dimension of a set $B \subseteq C$ is defined as the size of a minimal $\epsilon-$approximation for B. I.e,*

$$DIM(B,\epsilon) = \min\{\| A \| \ : \ A \ \ is \ \ an \ \ \epsilon - approximation \ \ of \ \ B \ \}.$$

The following two theorems relate the information complexity needed for learning a concept class to its dimension.

Theorem 1 *[BIK] If a concept class C in a bounded metric space[1] (X,d) is $(\epsilon,\delta)-$learnable then it is learnable using at most $DIM(C,\epsilon)$ queries.*

Theorem 2 *[BIK] Let (X,d) be a bounded metric space and C a concept class in (X,d). If C is $(\epsilon,\delta)-$learnable by l queries then*

$$l \geq \log_q \ Dim(C,\epsilon)$$

where $q = \lceil \frac{b-\epsilon}{\delta} \rceil$ and $b = \sup\{d(s,t) : s,t \in X\}$.

The argument beyond the above theorem yields the following:

Theorem 3 *Let S_X denote the cardinality of the range of the distance function d on the space X. Any LBD algorithm for (X,d) needs at least $log_{S_X} \| C \|$ -many queries for a precise learning of C (i.e. $\epsilon = \delta = 0$) in X.*

[1] A (pseudo-)metric space, X , is called *bounded* if there exists b such that $d(x,y) \leq b$ for every $x,y \in X$.

Definition 2 *Let (X,d) be a metric space, $C \subseteq X$ and $\delta > 0$.*

1. *A set $A \subseteq C$ is $\delta-$*uniform *if for every $s \in X$ there exists $a_s \in A$ such that*

$$\forall x \forall y \ \in A \setminus \{a_s\} \quad |d(s,x) - d(s,y)| \ \leq \delta.$$

   (In the sequel A_s will denote $A \setminus \{a_s\}$.)

2. $U(C,X,d) \stackrel{\text{def}}{=} \sup \{\| A \| \ : \ A \subseteq C \text{ is } uniform\}$

When $X = C$ and d is clear from the context, we use $U(X)$ to denote $U(C,X,d)$.

Theorem 4 *For every metric space (X,d) and a concept class C, $U(C,X,d)$ is a lower bound on the number of queries needed for learning C in (X,d).*

Proof Let A be a uniform set of size m and let S be a student that learns C. For every hypothesis $h \in X$ let d_h denote $d(h,x)$ for some $x \in A_h$. Let SEQ_l be the learning sequence of length at most l generated by S, when for any hypothesis $h \in X$, the teacher's response is $r(h) = d_h$. If $l < m$ then there exists a hypothesis $a \in A$ that doesn't appear in SEQ_l and $\forall h \in SEQ_l \ \ d(h,a) = d_h$. If the target is such an a the student fails. □

In the following examples $X = C$ and $U(C,X,d) = \| C \|$.

Example : for all $n \in \mathcal{N}$, let $X = 2^{\{0,1\}^n}$ and the distance between every two sets $a, b \in X$ is $d_n(a,b) \stackrel{\text{def}}{=} \| a \Delta b \| \ \cdot 2^{-n} = P(a \Delta b)$ where P is the uniform distribution over $\{0,1\}^n$. Endowed with this metric, the concept class $C = LPARITY$ over $\{0,1\}^n$ defined below, becomes the uniform metric space. It follows that no LBD algorithm can do any better than exhaustive search for this class.

Definition 3 *Given $S \neq \emptyset$, $S \subseteq \{1, \cdots n\}$ let $C_S \stackrel{\text{def}}{=} \{v \in \{0,1\}^n : \sum_{i \in S} v_i \equiv 0 \pmod 2\}$ The class $LPARITY$ over $\{0,1\}^n$ is $\{C_S \ : \ S \neq \emptyset \ , \ S \subseteq \{1, \cdots n\} \ \}$.*

In the following sections we show that in many cases it's possible to obtain efficient algorithms that learn by distances.

4 Learning Boolean Functions

In this section we demonstrate the learning process . We present efficient learning algorithms for the class of symmetric functions and the class of majority functions .

We consider an infinite family of metric spaces $\{(X_n, d_n)\}_{n \in \mathcal{N}}$, where $X = 2^{\{0,1\}^n}$ and the distance between every two sets $a, b \in X_n$ is $d_n(a,b) \stackrel{\text{def}}{=} \| a \Delta b \| \ \cdot 2^{-n}$.

Definition 4 *Given $S \subseteq \{0, 1, \cdots n\}$ let $C_S \stackrel{\text{def}}{=} \{v \in \{0,1\}^n \ : \sum_{i=1}^{i=n} v_i \ \in S\}$. The class of* symmetric functions over $\{0,1\}^n$, (SYM_n) *is $\{C_S \ : \ S \subseteq \{0, 1, \cdots n\}\}$.*

Theorem 5 *for all $n \in \mathcal{N}$, $0 < \epsilon < 1$, $\delta \leq \epsilon/2(n+1)$, there is an LBD student that $(\epsilon, \delta)-$learns SYM_n in (X_n, d_n) using $n+3$ queries and linear computation time.*

Proof Given $n \in \mathcal{N}$, S_{LBD} initializes an empty stack S', and queries $\emptyset$ (the empty set). The teacher's answer $r(\emptyset)$ is an approximation of the size of t. Then, for each $0 \leq i \leq n$, S_{LBD} queries $c_i \stackrel{\text{def}}{=} \{v \in \{0,1\}^n \ : \sum_{j=1}^{j=n} v_j = i\}$. S_{LBD} stores i in S' if and only if $r(c_i) < r(\emptyset)$. The last hypothesis of S_{LBD} is $h \stackrel{\text{def}}{=} C_{S'}$, and the teacher's answer is $r(h) = SUCCESS$. □

Definition 5 *For any non-empty $S \subseteq \{1, \cdots n\}$, let $C_S \stackrel{\text{def}}{=} \{v \in \{0,1\}^n : \sum_{i=1}^{n} v_i \ > \| S \| /2\}$. The class $MAJORITY_S$ over $\{0,1\}^n$ is $\{C_S \ : \ S \neq \emptyset, \ S \subseteq \{1, \cdots n\}\}$.*

Theorem 6 *Let $X = \{0,1\}^n$, $\delta \leq 5\sqrt{2}/(24\sqrt{\pi n})$,$0 < \epsilon < 1$, then $MAJORITY_S$ is $(\epsilon, \delta)-$learnable (in (X_n, d_n)) by $n+1$ queries and $\mathcal{O}(n)$ computation time.*

Proof Given $n \in \mathcal{N}$, S_{LBD} initializes an empty stack S'. Then, for each $1 \leq i \leq n$, S_{LBD} queries $c_i \stackrel{\text{def}}{=} \{ \ v \in \{0,1\}^n \ : v_i = 1\}$. If $r(c_i) = SUCCESS$ then S_{LBD} returns $h = c_i$. Otherwise, S_{LBD} stores i in S' if and only if $r(c_i) < 1/2 - \delta$. The last hypothesis of S_{LBD} is $h \stackrel{\text{def}}{=} C_{S'}$, and the teacher's answer is $r(h) = SUCCESS$.
To verify that the above strategy indeed works, let S_t denote the index set of the target t, i.e, $t = C_{S_t}$. Clearly, when $\| S_t \| = 1$ there exists $1 \leq i \leq n$ and $t = c_i$. In this case the student finds the target within i queries. For the other case ($\| S_t \| > 1$) , the following property

$$\forall i \quad i \in \{1 \cdots n\} \quad i \in S_t \quad \Longleftrightarrow \quad r(c_i) \ < \ 1/2 - \delta$$

guarantees the student's success. □

5 Learning a Vertex in a Graph

In this section we analyze the problem of learning in metric spaces induces by the shortest-path metric over graphs. We present lower and upper bounds for the information complexity of such tasks, in terms of the graph topology.

5.1 Definition of the Problem

There is a graph $G = (V, E)$ (directed or undirected) known both to the teacher and the student. The concept class is V and the metric space is (V, d_G), where the distance between every two vertices $u, v \in V$ ($d_G(u, v)$) is the length of the shortest path from u to v in G. If no such a path exists, then the distance between u and v is ∞.
Note that for a directed graph d_G is not necessarily a symmetric function .

Since the length of every edge in G is 1, d_G is a function from $V \times V$ to the non-negative integers, and the distance between every two different vertices in G is at least 1. It follows

that for any $\epsilon < 1$ the student is required to find the exact target, and for any $\delta < 1/2$ it's always possible to round the teacher's answers to the correct distances . We shall only discuss this case - i.e. assume the accuracy of the teacher, and demand precision from the learner.

The following technical notation is useful for analyzing the above problem.
$Dist(G)$ is the set of all the possible teacher's responses. I.e.
$Dist(G) \stackrel{def}{=} \{d_G(u,v) \ : \ u,v \in V \}$ S_G is the cardinality of $Dist(G)$.
For an undirected graph $G = (V,E)$ and a node $v \in V$, $deg(v)$ denotes the number of neighbors of v.
The *degree of* G, D_G is $max\{deg(v) \ : \ v \in V\}$.
For a directed graph $G = (V,E)$ and a vertex v, $d_{in}(v)$ is the number of incoming edges to v and $d_{out}(v)$ is the number of outgoing edges from v. Given a directed tree $T = (V,E)$, the *degree* of T is $D_T \stackrel{def}{=} \max\{d_{out}(v) \ : \ v \in V\}$.
The *depth* of the tree is the maximal distance between the root and one of the leaves.

Example Consider the graph $G = (V,E)$. $v = \{v_1, \cdots v_n\}$,
$E = \{(v_i, v_{i+1}) \ : \ 1 \leq i \leq n-1\}$. Any graph G' isomorphic to G is called a *string* . Notice that for each $1 \leq i \leq n$ $d_G(v_1, v_i) = i - 1$. Hence a LBD-student can pick v_1 as his first hypothesis , and according to the teacher's answer identify the target t correctly.

On the other hand, there are some more difficult cases.

Example Consider the *full graph (clique)*. In this case, by theorem 4, $\| V \|$ is a lower bound on the number of queries needed for learning V in (V, d_G).

5.2 Lower Bounds

In this section, we apply the lower bounds presented in section 3 , and derive lower bounds for the information complexity of learning some families of graphs.

Lemma 1 *For any undirected tree T of degree $D_T = d$, $U(T) \geq d$.*

Proof T is of degree d, hence there is a vertex $u \in V$ such that u has d many sons in T. Let $A = \{u_1 \cdots u_d\}$ be the set of $u's$ sons. For each $v \in V$ we construct A_v as follows:

$$A_v = \begin{cases} A \setminus \{u_1\} & \text{if } v \text{ is not a descendant of } u \\ A \setminus \{u_i\} & \text{if } v \text{ is a descendant of } u_i \text{ or } v = u_i \end{cases}$$

□

Corollary 7 *For any undirected tree $T = (V,E)$ of degree $D_T = d$, d is a lower bound on the number of queries needed for learning V in (V, d_G).*

Definition 6 *A graph G is a* full bipartite *graph iff for some $k \in \mathcal{N}$, G is isomorphic to (V,E) where $V = \{v_1, \cdots v_{2k}\}$ and $E = \{(v_i, v_j) \ : \ i \neq j \pmod 2\}$.*

Lemma 2 *For any full bipartite graph $G = (V,E)$,$U(G) \geq \| V \| /2$.*

Proof It's easy to verify that the set $A \stackrel{def}{=} \{v_i \ : \ i \equiv 1 \pmod 2\}$ is uniform.

Corollary 8 *For any full bipartite* $G = (V, E)$, $\| V \| /2$ *is a lower bound on the number of queries needed for learning* V *in* (V, d_G).

We presented two lower bounds on the number of queries needed for learning a vertex in an undirected tree (theorem 3 and corollary 7). It's worth noting that each of these bounds is better (I.e. higher) for some graphs.
Consider the full tree $T_{d,h}$ of depth h and degree d. The number of nodes in T is $\frac{d^{h+1}-1}{d-1}$ and the longest distance in $T_{d,h}$ is $2 \cdot h$. Hence, d is the lower bound derived from corollary 7, and $\log_{2 \cdot h+1} \frac{d^{h+1}-1}{d-1}$ is the lower bound derived from theorem 3. When d is large and h is small (for example, a one level tree with a root and d children) the first bound is higher, and when d is small and h is large (for example, a full binary tree with depth $h \geq 7$) the second bound is higher.

5.3 Upper Bounds

In this section, we present a strategy for learning a vertex in a general graph. We show some graph families, in which using this strategy leads to efficient learning.
In our learning strategy, the student picks a start vertex $s \in V$, and searches for the target along the shortest path from s to t. The basic idea is as follows: The student (S_{LBD}) picks an initial vertex s, and asks for it's distance from the target. Then he searches for a vertex u, a neighbor of s, such that u is on the shortest path from s to t. u is found after at most $deg(s)$ queries (or d_{out} (s) queries in a directed graph). S_{LBD} continues the search in the same way from u, until he reaches t.
As S_{LBD} approaches the target along the shortest path from s to t, the learning complexity depends on the length of this path, and on the degree (or out-degree) of the vertices along this path. The expected time complexity and number of queries is $\mathcal{O}(P_G \cdot D_G)$, where P_G is the maximal finite distance in G.
Since the above strategy reminds us of the well known BFS search, we call it *the naive search* (NS) .

Theorem 9 *Let* $G = (V, E)$ *be a connected undirected graph (or strongly connected digraph). Then* NS *(see (figure 1) in the appendix) learns* V *(in* (V, d_G)*) in time* $\mathcal{O}(P_G \cdot D_G)$.

Remark NS may be applied to unconnected undirected graphs and to directed not strongly connected graphs as well. Let G be an unconnected graph with k-connected components. By at most k queries the student can find which connected component contains t and apply NS to this component.
For directed graphs, the student has to find a set of vertices U, such that there is a directed path from U to every vertex in V. k will be the minimal size of such a set.

The efficiency of NS depends on the graph topology. We present some graph families in which $P_G \cdot d_G = \mathcal{O}(\log^k \| V \|)$ (i.e. NS is efficient).

Example consider the case of a *full tree* $T = (V, E)$ of depth h , and degree d. Learning V (in (V, d_G)) requires $1 + d \cdot h$ queries and computation time $\mathcal{O}(h \cdot d)$. The number of nodes in T is $\| V \| = \frac{d^{h+1}-1}{d-1} \approx d^h$. Hence $h \cdot d = \mathcal{O}(\log \| V \|)$ and NS is efficient.

Definition 7 *Let Σ be a finite alphabet of σ letters. The* de Bruijn Graph *$G_{\sigma,n}$ is the following directed graph:*

1. *V is the set of all σ^{n-1} words of length $n-1$ over Σ.*
2. *E is $\{(b_1 \cdots b_{n-1}, \quad b_2 \cdots b_n) \ : \ b_1 \cdots b_n \in \Sigma \}$.*

Lemma 3 *Let $G_{\sigma,n}$ be a de Bruijn graph. Then $\forall u \, \forall v \ \in V \quad d_G(v, u) \leq n - 1$.*

Corollary 10 *For any de Bruijn graph $G_{\sigma,n}$, learning V (in (V, d_G)) by NS requires $1+\sigma \cdot n$ queries and $\mathcal{O}(\sigma \cdot n)$ computation time .*
(Notice that $\sigma \cdot n \ = \ \mathcal{O}(\log \| V \|)$, hence NS is efficient.)

Our definition of NS doesn't determine the order in which $v's$ neighbors are visited. There are cases in which a wise choice of this order, and the starting point, speeds up the learning process. We demonstrate this phenomena on the family of *Binomial Trees* . This family was suggested by Vuillemin ([Vui]) as an efficient data structure for implementing priority-queues. This data structure handles an arbitrary sequence of n primitives, each drawn from { Insert , Delete , Min , Update , Union } in time $\mathcal{O}(n \log n)$ and $\mathcal{O}(n)$ memory cells. For further discussion on the combinatorial properties of this family consider Brown ([Br]).

Definition 8 *For each $k \geq 0$ we define a class B_k of ordered trees as follows:*

1. *Any tree consisting of a single node is a B_0 tree.*
2. *Suppose that $Z_{k-1}, \cdots Z_0$ are disjoint trees such that Z_l is a B_l tree for $0 \leq l \leq k-1$. Let R be a node which is disjoint from each Z_l. Then the tree obtained by taking R as the root and making the roots of $Z_{k-1}, \cdots Z_0$ the offspring of R, left to right in this order, is a B_k tree.*

A Binomial tree *is a tree which is in class B_k for some k.*

The following lemma describes some properties of binomial trees. These properties appear in [Br] and [Vui], and are easily verified by induction on k.

Lemma 4 *Let Z be a B_k tree. Then*

1. *Z has 2^k nodes.*
2. *The depth of Z is k.*
3. *In Z there is exactly one node , the root , having k children; for $0 \leq p < k$ there are 2^{k-p-1} nodes having p children.*

Lemma 4 yields an $\mathcal{O}(k^2)$ upper bound on the learning complexity of B_k. Notice that property 4 states that only the root has k children. Therefore, it might be possible to achieve a lower upper bound.

Theorem 11 *For any binomial tree B_k there is a naive search in which learning V (in (V, d_G)) requiring $k+1$ queries and $\mathcal{O}(k)$ computation time .*

Proof Let NS start at the root and search the offsprings of each node in decreasing order, in terms of their size. We prove by induction on k that this algorithm needs at most $k+1$ queries to learn a B_k tree. *basis:* trivial, since B_0 is a single node. *induction step:* Assume the theorem is true for every $i < k$. If $t = R$ then the first query completes the learning process. Otherwise , there exists $1 \leq i \leq k$ such that t is in the B_{k-i} offspring. The algorithm makes i queries to reach the correct son of the root and then, by the induction, at most $k-i+1$ queries to search that subtree. □

NS is not necessarily the most efficient way to learn a vertex in a graph. There are cases in which other algorithms are faster. For example, NS needs $\mathcal{O}(n)$ queries to handle an n-long string, while we have already demonstrated a 2 queries algorithm for any string.
The following family of *Hamming graphs* is another such example.

Definition 9 Hamming Graph *H_n is the following undirected graph:*

1. *V is the set of all 2^n words of length n over $\{0,1\}$.*
2. *$(v,u) \in E$ iff the words u and v are identical in exactly $n-1$ bits.*

As the degree of any node in H_n is n, and any pair of nodes have a path of length $\leq n-1$ connecting them, theorem 9 yields an $\mathcal{O}(n^2)$ upper bound on the learning complexity of H_n through naive search. Other algorithms can do better:

Theorem 12 *Let $H_n = (V,E)$ be a Hamming graph. Then V is learnable (in (V, d_{H_n})) by $n+2$ queries and $\mathcal{O}(n)$ computation time.*

Proof Given $n \in \mathcal{N}$, S_{LBD} initializes an empty stack S', and queries 0^n . Then, for each $0 \leq i \leq n$, S_{LBD} queries e^i when e^i is the i'th unit vector in $\{0,1\}^n$. S_{LBD} stores i in S' if and only if $r(e^i) < r(0^n)$. The last hypothesis of S_{LBD} is the characteristic vector of S', i.e. $\forall i \;\; 1 \leq i \leq n \;\; v_i = 1 \iff i \in S'$, and the teacher's answer is $SUCCESS$. □

6 Learning By Relative Distances

In this section, we present a new model - *learning by relative distances (RD)* . Here at each step of the learning process the student compares 2 hypotheses , and asks which one is closer to the target.
This notion of learning reminds us the "hot-cold" game. In this game player 1 hides a

treasure and player 2 tries to find it. In each step , player 1 informs player 2 whether he is approaching the target ("hotter") or retreating it ("colder"). The game continues until the treasure is found.

The formal definition of the RD learning model is as follows:
(X, d) is a metric space and $C \subseteq X$ as before.
Given ϵ and a target $t \in C$, an *ϵ-learning sequence for t* is a sequence

$$(< (c_{11}, c_{12}), r(c_{11}, c_{12}) >< (c_{21}, c_{22}), r(c_{21}, c_{22}) > \cdots < (c_{k1}, c_{k2}), r(c_{k1}, c_{k2}) >)$$

where $c_{i1}, c_{i2} \in X$ and $r(c_{i1}, c_{i2})$ describes which of them is closer to t. I.e.

$$r(c_{i1}, c_{i2}) = \begin{cases} 1 & if \;\; d(c_{i1}, t) < d(c_{i2}, t) \\ 2 & if \;\; d(c_{i1}, t) > d(c_{i2}, t) \\ 0 & if \;\; d(c_{i1}, t) = d(c_{i2}, t) \end{cases}$$

In addition, if $d(c_{i2}, t) \leq \epsilon$ then $r(c_{i1}, c_{i2}) = SUCCESS$.
The notions of $\epsilon-$learnability and learnability are defined analogously to the respective LBD definitions.

7 RD Vs. LBD

Notice that in the RD model we assumed the accuracy of the teacher. It's also possible to define several variations of this model, in which the teacher is not always accurate. We postpone a discussion of such variations to the full version of the paper. We wish to compare the RD model to the LBD scenario. As we discuss the accurate case in the RD setting, we feel that the comparison should be naturally made with an accurate LBD version.
An LBD teacher is accurate if its precision parameter δ satisfies:

$$\delta \;<\; 1/2 \cdot \inf\{\; d = |d(x_1, c_1) - d(x_2, c_2)| \;:\; x_1, x_2 \in X, \;\; c_1, c_2 \in C, \;\; d > 0 \;\}$$

In the LBD model the student gets more information. Within two queries h_1, h_2 the student knows the distances from the hypotheses to the target, and can tell which hypothesis is better (i.e. closer to t). Hence, the RD learnability of a concept class C implies that C is also learnable by distances. Furthermore, let l_{RD} , l_{LBD} denote the number of queries required for learning C in both models respectively, then $l_{LBD} \leq 2 \cdot l_{RD}$.

The following theorem states that in bounded metric spaces the reverse implication is valid too.

Theorem 13 *Let C be concept class in a bounded metric space (X, d), then C is $RD-$learnable if and only if it is $LBD-$learnable.*

Proof One should just note that the [BIK] proof of their theorem 1 also implies that C is $RD-$learnable iff $\forall \epsilon > 0 \;\; DIM(C, \epsilon) \;< \infty$. □
Remark Notice that theorem 13 doesn't hold in an unbounded metric space. Consider $X = C = \mathcal{N}$ when the distance between every to integers $i, j \in \mathcal{N}$ is $|i - j|$. C is LBD-learnable, but it is not RD-learnable.
Some of the bounds developed above for the LBD model, translated immediately to the RD model.

1. **Upper Bounds**
   There are cases in which the learner uses the distances only to check whether he is moving towards the target or away of it. In such cases the same learning algorithm is applicable in the RD model as well. This, for example, is the case with

   - Exhaustive search. S_{RD} can search an $\epsilon-$approximation $\{a_1, \cdots a_l\}$ of C , by l queries of the form (a_i, a_i).
   - NS can be applied in the RD model as well (with the same learning complexity). Consequently, every upper bound achieved by NS is valid in both models.
   - The family of Hamming graphs. S_{LBD} identifies the target by comparing each of the unit vectors e^i to 0^n.
   - The class of symmetric functions over $\{0,1\}^n$. S_{LBD} learns this class by comparing $n+1$ hypotheses to $\emptyset$.
   - The class $MAJORITY_S$ is RD-learnable by n comparisons too. Notice that S_{LBD} stores in the stack the indices of the queries in which $r(c_i) < 1/2 - \delta$. The argument beyond his strategy is that

     $$\forall i \;\; 1 \leq i \leq n \;\; i \notin S_t \Rightarrow d(c_i, t) = 1/2 \;\; and \;\; i \in S_t \Rightarrow \;\; d(c_i, t) < 1/2 \;\; .$$

     S_{RD} can reach the same conclusions by n comparisons of the form $(\overline{c_i}, c_i)$ when $\overline{c_i} = \{0,1\}^n \setminus c_i$.

2. **Lower Bounds**

   - The fact that $l_{LBD} \leq 2 \cdot l_{RD}$ implies that if k is a lower bound on the number of queries in the LBD model, then $k/2$ is a lower bound on the number of queries in the RD model.
   - In theorems 2 ,3 we presented lower bounds on the number of queries needed for learning C in terms of the $\epsilon-$dimension of C. In these theorems q and S_G represent the number of possible different responses of the LBD teacher along the learning process. An RD teacher is limited to 4 possible answers $\{0, 1, 2, SUCCESS\}$. Hence, the lower bound is $\log_4 DIM(C, \epsilon)$.

Notice that the last lower bound implies gaps in the learning complexity of these models. Consider the case of an n-long string. $\log_4 n$ is a lower bound on the number of queries needed for learning V in the RD model, while we have already seen that two queries suffice to learn any string in the LBD model.

8 Acknowledgement

This paper is based on my M.Sc. thesis written at the Technion under the supervision of Shai Ben-David. I wish to thank him for his many significant contributions to this work.

References

[BIK] Ben-David S., Itai A., and Kushilevitz E. , "Learning by Distances", In *Proceeding of the 3rd Workshop on Computational Learning Theory,* August 1990, pp. 232-245.

[Br] Brown R.M. , "Implementation and Analysis of Binomial Queue Algorithms", *SIAM Journal of Computing* 7(3) pp. 298-319 , 1978.

[KT] Kolomogorov A.N., and V.M. Tihomirov, *AMS Translations,* series 2 , Vol. 17.

[Vui] Vuillemin J. "A Data Structure for manipulating priority queues" , *CACM* 21(4) pp. 309-315 1978.

9 Appendix

```
INPUT: G = (V, E)
begin
1. v := v_1
2. query v
3. if r(v) = SUCCESS then return v
4. b(v) := 1            (* marking a visited vertex, initially all vertices are unmarked . *)
5. repeat
6.    found:= FALSE
7.    while (not found) do
8.    begin
9.       { find u ∈ V such that (v, u) ∈ E  and b(u) = 0
         { query u
         { b(u) = 1
         { if (r(u) = SUCCESS or r(u) = d − 1) then found := TRUE
10.   end
11.   r(v) := r(u)
12.   v := u
13. until SUCCESS
14. return(v)
end
```

Figure 1: NS : Learning A Vertex in a General Graph