Property Testing: Monotonicity and regular languages, Lower bounds

Michel de Rougemont
IRIF-CNRS

Abstract

Introduction to Property Testing and sublinear algorithms: how to approximate decision problems with just local queries?

Contents

1 Introduction

Property testing was introduced in [7], building on earlier notions of Program testing [5]. The main definition is inspired by the IP and PCP complexity classes, as an attempt to have a probabilistic generalization of the class NP. It is a statistics based approximation technique to decide if either an input satisfies a given property, or is far from any input satisfying the property, using only few samples of the input and a specific distance between inputs. The idea of moving the approximation to the input was implicit in *Program Checking* [4, 10], in *Probabilistically Checkable Proofs* (PCP) [2], and explicitly studied for graph properties under the context of Property Testing [7].

The class of sublinear algorithms has similar goals: given a massive input, a sublinear algorithm can approximately decide a property by sampling a tiny fraction of the input. The design of sublinear algorithms is motivated by the recent considerable growth of the size of the data that algorithms are called upon to process in everyday real-time applications, for example in bioinformatics for genome decoding or in Web databases for the search of documents. Linear-time, even polynomial-time, algorithms were considered to be efficient for a long time, but this is no longer the case, as inputs are vastly too large to be read in their entirety.

Given a distance between inputs, an ε-tester for a property P accepts all inputs which satisfy the property and rejects with high probability all inputs which are ε-far from inputs that satisfy the property. Inputs which are ε-close to the property determine a gray area where no guarantees exists. These restrictions allow for sublinear algorithms and even $O(1)$ time algorithms, whose complexity only depends on ε.

Let $\mathbf{K}$ be a class of finite structures with a normalized distance dist between structures, i.e. dist lies in $[0, 1]$. For any $\varepsilon > 0$, we say that $U, U' \in \mathbf{K}$ are *ε-close* if their distance is at most ε. They are *ε-far* if they are not ε-close. In the classical setting, satisfiability is the decision problem whether $U \models P$ for a structure $U \in \mathbf{K}$ and a property $P \subset \mathbf{K}$. A structure $U \in \mathbf{K}$ *ε-satisfies* P, or U is ε-close to $\mathbf{K}$ or $U \models_\varepsilon P$ for short, if U is ε-close to some $U' \in \mathbf{K}$ such that $U' \models P$. We say that U is ε-far from $\mathbf{K}$ or $U \not\models_\varepsilon P$ for short, if U is not ε-close to $\mathbf{K}$. Recall the definition given in the first lecture.

Definition 1 (Property Tester [7]) *Let $\varepsilon > 0$. An* ε-tester *for a property $P \subseteq \mathbf{K}$ is a randomized algorithm A such that, for any structure $U \in \mathbf{K}$ as input:*
(1) If $U \models P$, then A accepts;
(2) If $U \not\models_\varepsilon P$, then A rejects with probability at least $2/3$.[1]

A *query* to an input structure U depends on the model for accessing the structure. For a word w, a query asks for the value of $w[i]$, for some i. For a tree T, a query asks for

[1] The constant $2/3$ can be replaced by any other constant $0 < \gamma < 1$ by iterating $O(\log(1/\gamma))$ the ε-tester and accepting iff all the executions accept

the value of the label of a node i, and potentially for the label of its parent and its j-th successor, for some j. For a graph a query asks if there exists an edge between nodes i and j. We also assume that the algorithm may query the input size. The *query complexity* is the number of queries made to the structure. The *time complexity* is the usual definition, where we assume that the following operations are performed in constant time: arithmetic operations, a uniform random choice of an integer from any finite range not larger than the input size, and a query to the input.

Definition 2 *A property $P \subseteq \mathbf{K}$ is* testable*, if there exists a randomized algorithm A such that, for every real $\varepsilon > 0$ as input, $A(\varepsilon)$ is an ε-tester of P whose query and time complexities depend only on ε (and not on the input size).*

We have seen the linearity test, on the first lecture. We now consider some other properties.

2 Monotonicity Test

Let $f : F_2^n \to \{0, 1, ...r\}$ be a discrete function. Such a function is monotone if:

$$f(0, x_{-i}) \leq f(1, x_{-i})$$

Monotonicity Test(f, t)
Repeat t times
1. Generate uniformly $i \in_r \{1, 2, ...n\}$ and $x_{-i} \in_r F_2^{n-1}$,
2. Reject if $f(0, x_{-i}) > f(1, x_{-i})$

Accept.

Clearly, if f is monotone, it satisfies the Monotonicity Test with probability 1. The main question is to bound t to insure that a function f which is ε-far from monotone is rejected with high probability.

Theorem 1 *For $t = \Omega(n/\varepsilon)$, the Monotonicity Test rejects every function f which is ε-far from monotone.*

Proof: Equivalently, let us show that the probability that the test rejects in one trial is at least ε/n when f is ε-far. The number of possible edges of the hypercube is $n.2^{n-1}$. The Tester selects an i, and an hyperplane A_i separating the nodes with $x_i = 0$ from the nodes $x_i = 1$. Let α_i be the number of edges which cut the hyperplanes with $f(0, x_{-i}) > f(1, x_{-i})$. Then the Tester rejects with probability:

$$\frac{\sum_i \alpha_i}{n.2^{n-1}}$$

Consider the following partial Corrector: for each edge e of the hypercube which cuts the hyperplane, if $f(0, x_{-i}) > f(1, x_{-i})$, then switch the values of f. A case by case analysis shows that the new function f_i has no more errors than f on each hyperplane A_j, i.e. for $j = 1, ...n$, $\alpha_j \geq \alpha'_j$. For every $A_1, ...A_n$ we make two modifications for every edge with an error: hence f can be made monotone after modifying at most $2.\sum_i \alpha_i$ values. If f is ε-far from monotone, then $2.\sum_i \alpha_i \geq \varepsilon.2^n$.

The probability to reject is $\frac{\sum_i \alpha_i}{n.2^{n-1}} \geq \frac{\varepsilon.2^{n-1}}{n.2^{n-1}} = \frac{\varepsilon}{n}$. □

3 Lower bounds

Recall from Communication Complexity $R(f)$ as $R_{1/3}$ is the Communication complexity with error 1/3, and $R^1(f)$ as $R^1_{1/2}$ as the one-sided Communication complexity with error 1/2 and $R^{\rightarrow}(f)$ as $R^{\rightarrow}_{1/3}$ the one-way Communication complexity with error 1/3. Given a property P of functions, let $Q(P)$ the minimum cost of an adaptative tester with two-sided error. Similarly $Q^1(P)$ is the 1-sided error version and $Q^{na}(P)$ is the non-adaptative version, when the random choices are made only at the beginning of the protocol. The general lower bound method, introduced in [3] is the following setting: given a property P of functions with boolean output, let $P_{\oplus}$ be following CC problem $C(h, P)$:

1. Alice gets a function f,
2. Bob gets a function g,
3. The output $P_{\oplus}(f, g) = 1$ iff the function h defined by $h(x) = f(x) \oplus g(x)$ has the property P.

Lemma 1 *For any function h and property P:*

1. $R(C(h, P) \leq 2.Q(P)$,
2. $R^1(C(h, P) \leq 2.Q^1(P)$,
3. $R^{\rightarrow}(C(h, P) \leq Q^{na}(P)$.

Proof: Given a Tester with t queries, Alice and Bob generate $x_1, ...x_t$. They exchange the values of $f(x_i)$ and $g(x_i)$. Each player can compute $h(x_i)$, use the tester to decide if $h \in P$. For the non adaptative tester, they first decide on the $x_1, ...x_t$. Then Alice transmits $f(x_1), f(x_2), ...f(x_t)$ to Bob which use the tester to decide if $h \in P$. □

3.1 Lower bound for k-linearity

A function f is k-linear if there is S such that $|S| = k$ and $f(x) = \sum_{i \in S} x_i = \bigoplus_{i \in S} x_i$. Let $k/2$-Disjointness be the Disjointness problem when the sets are of size $k/2$.

Lemma 2 *$R(k/2$-Disjointness$) \leq 2.Q(k$-linearity$)$,*

Proof: Let $\chi_S = \bigoplus_{i \in S} x_i$ and $\chi_T = \bigoplus_{i \in T} x_i$. If $S \cap T = \emptyset$, then h is k-linear. If $S \cap T \neq \emptyset$, then h is k'-linear for $k' < k$, and therefore 1/2-far from a k-linear function. $\square$

As $\Omega(k) = R(k\text{-Disjointness})$, we conclude that $\Omega(k) < Q(k\text{-linearity})$ when $k \leq n/2$.

3.2 Lower bound for Monotonicity

Definition 3 *Let, $A, B \subseteq U = \{1, 2, ...n\}$. Let $h_{A,B} : \ 2^U \to Z$ be defined by:*

$$h_{A,B}(S) = 2.|S| + (-1)^{S \cap A} + (-1)^{S \cap B}$$

Lemma 3 *The function h is such that:*
(i) If $A \cap B = \emptyset$, then $h_{A,B}$ is monotone,
(ii) If $|A \cap B| = 1$, then $h_{A,B}$ is 1/8-far from monotone.

Proof: If $A \cap B = \emptyset$, and $S_1 \subseteq S_2$, then the $2.|S|$ factor increases faster than the $(-1)^{|S \cap A|}$ or the $(-1)^{|S \cap B|}$ factor.

If $|A \cap B| = 1$, i.e. $A \cap B = \{i\}$, let $S \subseteq U = \{1, 2, ...n\} - \{i\}$. $Prob[|A \cap S| \ is \ even \ ] \geq 1/2$, as it is 1 if $A = \{i\}$ and 1/2 otherwise. Similarly $Prob[|B \cap S| \ is \ even \ ] \geq 1/2$. These two events are independent as $|A \cap B| = 1$. Hence $Prob[|B \cap S| \ is \ even \ \wedge |A \cap S| \ is \ even \ ] \geq 1/4$. In this case $h_{A,B}(S \cup \{i\}) - h_{A,B}(S) = -2$, i.e. contradicts the monotonicity. There are $1/4.2^{n-1}$ possible S, hence $2^n/8$ errors.

$\square$

Reduction from UNIQUE-DISJOINTNESS to MONOTOCITY

- 1-inputs (x, y) of UNIQUE-DISJOINTNESS map to functions $h_{(x,y)}$ which are MONOTONE
- 0-inputs (x, y) of UNIQUE-DISJOINTNESS map to functions $h_{(x,y)}$ which are ε-far from MONOTONE

Theorem 2 *For large enough r and $\varepsilon = 1/8$, then every MONOTOCITY Test requires $\Omega(n)$ queries.*

Proof: Assume, Alice knows A, Bob knows B and they try to compute $h_{AB}(S)$. They exchange $(-1)^{|S \cap A|}$ and $(-1)^{|S \cap B|}$, i.e. 2 bits. Knowing if Q is monotone with q queries, we would deduce if A and B are disjoint. As UNIQUE-DISJOINTNESS requires $\Omega(n)$ communications bits, every MONOTOCITY test requires $\Omega(n)$ queries.

$\square$

4 Testing graphs, words and trees

We detail some of the methods on Graphs, Words and Trees.

4.1 Graphs

In the context of undirected graphs [7], the distance is the (normalized) Edit Distance on edges: the distance between two graphs on n nodes is the minimal number of edge-insertions and edge-deletions needed to modify one graph into the other one. Let us consider the adjacency matrix model. Therefore, a graph $G = (V, E)$ is said to be ε-close to another graph G', if G is at distance at most εn^2 from G', that is if G differs from G' in at most εn^2 edges.

In several cases, the proof of testability of a graph property on the initial graph is based on a reduction to a graph property on constant size but random subgraphs. This was generalized for every testable graph properties by [8]. The notion of ε-reducibility highlights this idea. For every graph $G = (V, E)$ and integer $k \geq 1$, let Π denote the set of all subsets $\pi \subseteq V$ of size k. Denote by G_π the vertex-induced subgraph of G on π.

Definition 4 *Let $\varepsilon > 0$ be a real, $k \geq 1$ an integer, and ϕ, ψ two graph properties. Then ϕ is* (ε, k)-reducible to ψ *if and only if for every graph G,*

$$\begin{aligned} G \models \phi \quad &\Longrightarrow \quad \forall \pi \in \Pi,\ G_\pi \models \psi, \\ G \not\models_\varepsilon \phi \quad &\Longrightarrow \quad \Pr_{\pi \in \Pi}[G_\pi \not\models \psi] \geq 2/3. \end{aligned}$$

Note that the second implication means that if G is ε-far to all graphs satisfying the property ϕ, then with probability at least $2/3$ a random subgraph on k vertices does not satisfy ψ.

Therefore, in order to distinguish between a graph satisfying ϕ to another one that is far from all graphs satisfying ϕ, we only have to estimate the probability $\Pr_{\pi \in \Pi}[G_\pi \models \psi]$. In the first case, the probability is 1, and in the second it is at most $1/3$. This proves that the following generic test is an ε-tester:

Generic Test(ψ, ε, k)

1. Input: A graph $G = (V, E)$
2. Generate uniformly a random subset $\pi \subseteq V$ of size k
3. Accept if $G_\pi \models \psi$ and reject otherwise

Proposition 1 *If for every $\varepsilon > 0$, there exists k_ε such that ϕ is $(\varepsilon, k_\varepsilon)$-reducible to ψ, then the property ϕ is testable. Moreover, for every $\varepsilon > 0$,* ***Generic Test***$(\psi, \varepsilon, k_\varepsilon)$ *is an ε-tester for ϕ whose query and time complexities are in $(k_\varepsilon)^2$.*

In fact, there is a converse of that result, and for instance we can recast the testability of c-colorability [7, 1] in terms of ε-reducibility. Note that this result is quite surprising since c-colorability is an NP-complete problem for $c \geq 3$.

Theorem 3 ([1]) *For all $c \geq 2$, $\varepsilon > 0$, c-colorability is $(\varepsilon, O((c \ln c)/\varepsilon^2))$-reducible to c-colorability.*

If for every $\varepsilon > 0$, there exists k_ε such that ϕ is $(\varepsilon, k_\varepsilon)$-reducible to ψ, then the property ϕ is testable. Moreover, for every $\varepsilon > 0$, **Generic Test**$(\psi, \varepsilon, k_\varepsilon)$ is an ε-tester for ϕ whose query and time complexities are in $(k_\varepsilon)^2$.

In fact, there is a converse of that result, and for instance we can recast the testability of c-colorability [7, 1] in terms of ε-reducibility. Note that this result is quite surprising since c-colorability is an NP-complete problem for $c \geq 3$.

Theorem 4 ([1]) *For all $c \geq 2$, $\varepsilon > 0$, c-colorability is $(\varepsilon, O((c \ln c)/\varepsilon^2))$-reducible to c-colorability.*

4.2 Words and Trees

Consider the following operations on strings:

1. Modification of a letter
2. Insertion of a letter
3. Deletion of a letter
4. Move of a factor (substring), i.e. Cut/Paste

Given two strings w, w', the absolute distance is the minimum number of operations (among some of the 4 possibilities we select) necessary to transform w into w'. The distance $\mathsf{dist}(w, w')$ is the relative distance divided by the maximum length of w, w'. Notice that dist is symetric ans satisfies the triangular inequality.

If we only take the Modification operator, we get the Hamming distance dist_H. If we take Modification, Insertion, Deletion we get the Edit distance dist_E and if take all 4 operators we get the Edit distance with Moves dist_M. Notice that:

$$\mathsf{dist}_M(w, w') \leq \mathsf{dist}_E(w, w') \leq \mathsf{dist}_H(w, w')$$

Given an automaton $\mathcal{A}$, let $L(\mathcal{A})$ be the language accepted. A Tester for the Edit distance has the following structure. We separate the strongly connected components $C_1....C_p$ and the possible paths $\Pi = C_1....C_l$ from C_1 which contains the initial state to C_l which contains a final state, such that C_{i+1} is reachable from C_i. We first construct a Tester T_1 for one component C, then a Tester for $\Pi = C_1....C_l$ and finally a tester for $L(\mathcal{A})$.

Tester $T_1(w, C)$

1. Sample a factor u of length $k = 2.(m+1)/\varepsilon$ of w_n,
2. If u is C-compatible Accept else Reject.

4.2.1 Tester for a Component C

We say that w is C-compatible if there is run for w in C, i.e. there are two states q, q' such that we reach q' from q reading w. We want to show that if w is ε-far from C, there are many incompatible factors u of some length. We conceive the following Corrector for C.

Start w in some state q which maximizes the length of a run. At this point we have a *cut*: we remove the letter, and start again from another state q_1 which maximizes another run. We then introduce a *link*, a small word of length less than m to connect q' with q_1. We made at most $m+1$ Edit operations. Let h the number of cuts in w. We write

$$w = w_1 \ |_1 \ w_2 \ |_2 \ w_3 \ _3| \ ... \ |_h \ w_{h+1}$$

for a word with h cuts.

Lemma 4 *Let C have at most m states. If w has h cuts, then it is $h.(m+1)$ close to C.*

Proof: We just correct w with the sequence of modifications along the cuts, defined by the previous Corrector. □

By contraposition, if w is ε-far from C, we expect many cuts. Observe that if a sample u contains 2 cuts, it is necessarly incompatible, i.e. a witness that w is not accepted by C, for any initial and final state. We need to bound the size of a sample so it occurs with a constant probability.

Let $\alpha_i = |\{w_j : \ 2^{i-1} \leq |w_j| < 2^i\}|$ where $|w_j|$ is the length of w_j. By definition $h = \sum_i \alpha_i$ is the number of cuts. We need to find a bound i_l such that:

$$\sum_{0 \leq i \leq i_l} \alpha_i \geq \varepsilon.n$$

It will guarantee that samples of length $k = 2^{i_l}$ will contain many incompatible factors.

Lemma 5 *If w is ε-far from C, then the probability to find an incompatible factor u of length $2.k = 8.(m+1)/\varepsilon$ is greater than $c = 3.\varepsilon/8.(m+1)$.*

Proof: We need to estimate $\sum_{0\leq i\leq i_l}\alpha_i$ when we choose $k=2^{i_l}$. First let us estimate $\beta=\sum_{i\geq i_l}\alpha_i$, i.e. for large i. There are at most $n/2^{i_l}$ feasible w_j of length larger than 2^{i_l}, i.e. $\beta\leq n/2^{i_l}$. Hence

$$\sum_i \alpha_i = \sum_{0\leq i\leq i_l}\alpha_i+\beta\geq \varepsilon.n/(m+1)$$

$$\sum_{0\leq i\leq i_l}\alpha_i\geq \varepsilon.n/(m+1)-\beta\geq \varepsilon.n/(m+1)-n/2^{i_l}$$

Let $k=4.(m+1)/\varepsilon=2^{i_l}$, or $i_l=\log(4.(m+1)/\varepsilon)$. Then

$$\sum_{0\leq i\leq i_l}\alpha_i\geq 3.\varepsilon.n/4.(m+1)$$

Hence:

$$\beta=\sum_{i\geq i_l}\alpha_i\leq n/2^{i_l}\leq \varepsilon.n/4.(m+1)$$

Let us estimate the probability to have two consecutive small w_j, w_{j+1} in a sample u which starts in w_j. If this probability is large, it will guarantee that a sample of length $2.k$ starting in w_j is incompatible. We bound the probability that we hit a small w_j such that the following w_{j+1} is large.

$$Prob[|w_j|\leq k]\geq 3.\varepsilon/4.(m+1)$$

It is the weight of the small blocks: in the worst case they are of length 1 and their weight is $3.\varepsilon/4.(m+1)$. Consider now, the probability that the random u starts on a small w_j followed by a large w_{j+1}:

$$Prob[|w_{j+1}|>k \ \mid \ |w_j|\leq k]$$

For this event, a large block of size at least $4.(m+1)/\varepsilon$ follows a small block: there are only $\varepsilon.n/4.(m+1)$ small blocks which could be used. There remains $\varepsilon.n/2.(m+1)$ small blocks. Let us call A the set of all positions determined by the small blocks followed by the large blocks. The remaining $\varepsilon.n/2.(m+1)$ small blocks occupy at least $n-\mid A\mid$ positions. Hence:

$$Prob[|w_{j+1}|>k \ \mid \ |w_j|\leq k]\leq Prob[w_j\in A]/2\leq (1-\varepsilon/2.(m+1))/2=1/2-\varepsilon/4.(m+1)\leq 1/2$$

By contraposition:

$$Prob[|w_{j+1}|\leq k \ \mid \ |w_j|\leq k]\geq 1/2$$

We can then bound the probability that a sample of weight $2k$ is incompatible:

$$\begin{aligned} Prob[\text{a sample } u \text{ of weight } 2k \text{ is incompatible }] &\geq Prob[|w_j|\leq k].Prob[|w_{j+1}|\leq k \ \mid \ |w_j|\leq k] \\ &\geq (3.\varepsilon/4(m+1)).1/2\geq 3.\varepsilon/8.(m+1) \end{aligned}$$

Hence we take $c=3.\varepsilon/8.(m+1)$.

□

We can conclude:

Theorem 5 *The Tester T_1 accepts if $w \in L(\mathcal{A})$ and rejects with probability $3\varepsilon/4(m+1)$ if w is ε-far from $L(\mathcal{A})$*

4.2.2 Tester for a sequence Π of compatible connected components

Consider the following decomposition for $\Pi = C_1, C_2$, as in Figure 1, which we can generalize for an arbitrary Π: start in all possible states of C_1 wich are accessible from the initial state and take the longest compatible prefix w_1. It determines a cut of weight c. We continue in a similar way until we reach cuts of total weight at least $\varepsilon.n/2$ for C_1. We then switch to cuts for C_2. The position of the last cut for C_1 determines a border between C_1 and C_2, set by the intervals I_1 and I_2. If there are cuts for C_1 of weight less than $\varepsilon.n/2$, or cuts for C_1 of weight $\varepsilon.n/2$ and cuts for C_2 of weight less than $\varepsilon.n/2$, then the word w_n is ε-close to the regular expression associated with $\Pi = C_1, C_2$.

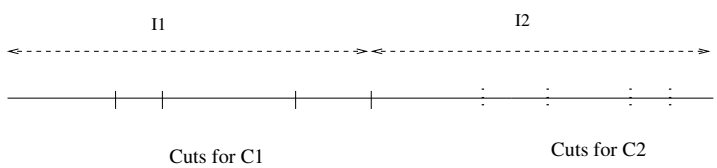

Figure 1: A possible decomposition of w_n into feasible components for $\Pi = C_1, C_2$.

We say that two independent samples $u_1 < u_2$ are $\Pi = C_1.C_2$ compatible if u_1 is compatible for C_1 or C_2 or $C_1.C_2$ and u_2 is compatible accordingly. We generalize to $\Pi = C_1....C_l$.

Tester $T_2(w, \Pi)$
$\Pi = C_1....C_l$

1. Sample l independent factors $u_1 < u_2 < ... < u_l$ of length $k = 2.(m+1)/\varepsilon$ of w_n,
2. If $u_1 < u_2 < ... < u_l$ is Π-compatible Accept else Reject.

If w_n is ε-far for Π, there exists such a decomposition, given as (I_1, I_2) where there are heavy cuts, i.e. of weight greater than $\varepsilon.n/2$, for C_1 and C_2. We then show that there are many samples u of a some finite weight which contain a w_i for the C_1 decomposition and a w_j for the C_2 decomposition. This decomposition generalizes to $\Pi = C_1....C_l$ by taking cuts for C_{i_1} of weight $\varepsilon.n/l$, cuts for C_{i_2} of weight $\varepsilon.n/l$ until possible cuts for C_l of weight $\varepsilon.n/l$.

If w_n is ε-far from $\Pi = C_1, C_2$, we know that there are many samples u's of weight k incompatible for C_1 in the interval I_1 and many samples u's of weight k incompatible for C_2 in the interval I_2. The density of the incompatible samples is greater than $3.\varepsilon.n/4.(m+1)$. We can then conclude that the Tester will reject with constant probability.

Lemma 6 *If w_n is ε-far from $\Pi = C_1, C_2$ then the Word Tester rejects with constant probability.*

Proof: Assume w_n is ε-far from $\Pi = C_1, C_2$. Consider two samples $u_1 < u_2$ taken independently. If u_1 is incompatible for C_1 and u_2 is incompatible for C_2, then the Tester rejects. Hence:

$$Prob[\text{Tester rejects}] \geq Prob[u_1 \text{ incompatible for } \mathrm{C}_1 \wedge u_2 \text{ incompatible for } \mathrm{C}_2] \geq$$

$$Prob[(u_1 \in I_1 \wedge u_1 \text{ incompatible for } \mathrm{C}_1) \wedge (u_2 \in I_2 \wedge u_2 \text{ incompatible for } \mathrm{C}_2] \geq$$

These two events are independent, hence we can rewrite the expression as:

$$Prob[(u_1 \in I_1 \wedge u_1 \text{ incompatible for } \mathrm{C}_1)].Prob[(u_2 \in I_2 \wedge u_2 \text{ incompatible for } \mathrm{C}_2]$$

But $Prob[(u_1 \in I_1 \wedge u_1 \text{ incompatible for } \mathrm{C}_1)] \geq (\varepsilon/2).(3.\varepsilon/4.(m+1))$ and similarly for u_2. Hence:

$$Prob[\text{Tester rejects}] \geq (3.\varepsilon^2/8(m+1))^2$$

□

This decomposition generalizes to $\Pi = C_1....C_l$ by taking cuts for C_{i_1} of weight $\varepsilon.n/l$, cuts for C_{i_2} of weight $\varepsilon.n/l$ until possible cuts for C_l of weight $\varepsilon.n/l$.

Notice that another Tester $T_2'(w, \Pi)$ could be used: given $u_1 < u_2 < ... < u_l$, use $T_1(u_i, C_i)$, i.e. if each u_i is compatible for C_i for $i = 1, ...l$. It would accept of one of test T_1 accepts and reject otherwise.

4.2.3 Tester for a $L(\mathcal{A})$

We can now use the previous Tester T_2 and have a Tester T_3 for $L(\mathcal{A})$. A sequence $\Pi = C_1....C_l$ is essential if the initial state is in C_1 and some initial state is in C_l.

Tester $T_3(w, L(\mathcal{A}))$

For all essential $\Pi = C_1....C_l$ of the automaton:

1. If $T_2(w, \Pi)$ Rejects, Reject,

Accept.

A much simpler tester is given in [6] for the Edit distance with Moves.

5 Testing vs. Learning

5.1 Learning a linear Classifier

A Classifier takes data labelled with the name of a class and outputs linear functions which best separate the classes. In the case of two classes, and two dimensional data, we have labelled data $(x_i, y_i, 0)$ if 0 is the label of the first class and $(x_i, y_i, 1)$ if 1 is the label of the second class.

A linear regression finds the linear function $a.x + b.y + c = 0$ which minimizes the global error, defined as the sum of the distances from the points to the line, the Ordinary least squares.

PAC-learning is the generalization to words, graphs and general data.

5.2 Learning a Community in a graph

In this case, we don't have labelled data as input. We want to find the dense subgraphs or clusters. It is often called unsupervised learning.

6 Homework

1. A function $f : F_2^n \to F_2$ is a *Dictator function* if $f = x_i$ for some $i \in \{1, 2, ...n\}$, i.e. a special linear function. Construct a Tester for this property.

2. Recall that a function f is k-linear if there is S such that $|S| = k$ and $f(x) = \sum_{i \in S} x_i = \bigoplus_{i \in S} x_i$. A function f is a k-junta if there is S such that $|S| = k$ and $f(x) = f(y)$ whenever $x_i = y_i$ for $i \in S$. Construct a tester for this class of functions and show a lower bound $\Omega(k)$ lower bound for the number of queries.

7 Open problems

1. Non deterministic Property Testing [9].

2. Dynamic reservoir sampling.

3. Uniform Spanning trees from a stream of edges.

References

[1] Noga Alon and Michael Krivelevich. Testing k-colorability. *SIAM J. Discrete Math.*, 15(2):211–227, 2002.

[2] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of np. *J. ACM*, 45(1):70–122, 1998.

[3] Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. *Computational complexity*, 2:311–358, 2012.

[4] M. Blum and S. Kannan. Designing programs that check their work. *Journal of the ACM*, 42(1):269–291, 1995.

[5] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *ACM Symposium on Theory of Computing*, pages 73–83, 1990.

[6] E. Fischer, F. Magniez, and M. de Rougemont. Approximate satisfiability and equivalence. *SIAM Journal of Computing*, 39(6):421–430, 2010.

[7] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998.

[8] Oded Goldreich and Luca Trevisan. Three theorems regarding testing graph properties. *Random Struct. Algorithms*, 23(1):23–57, 2003.

[9] L. Lovász and K. Vesztergombi. Nondeterministic graph property testing. *ArXiv e-prints*, 2012.

[10] R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):23–32, 1996.