Probably Approximately Correct Learning and Property Testing

Rémi Gilleron

Inria Lille - Nord Europe & LIFL & Univ Lille

juillet 2010

Introduction

Motivations

- The study of property testing algorithms for the membership problem for tree languages,
- connections between property testing and PAC learning.

Sources

- An introduction to computational learning theory, M.J. Kearns and U.V. Vazirani, the MIT press, 1994.
- Property testing and its connection to learning and approximation, O. Goldreich and S. Goldwasser and D. Ron, journal of the ACM, 653–750, 1998.

Plan

Preliminaries

Concepts

- Let $X = \{0,1\}^n$ be the instance space.
- A concept c is a subset of X or a function from X into $\{0,1\}$,
- a concept class $\mathcal{C}$ is a set of concepts.
- Examples of representation classes are CNF, DNF, Boolean circuits, ...

Measuring errors

- Let c be a target concept, let D be a probability distribution over X,
- $EX(c,D)$ is a procedure that, on each call, returns a labelled example $< \mathbf{a}, c(\mathbf{a}) >$ where $\mathbf{a}$ is drawn randomly and independently according to D.
- The error between c and an hypothesis concept h is defined

$$error(h) = Pr_{\mathbf{a} \sim D}[c(\mathbf{a}) \neq h(\mathbf{a})].$$

PAC model

Definition 1

$\mathcal{C}$ is efficiently properly PAC learnable if there exists an algorithm L such that :
for every distribution D, $0 < \epsilon < \frac{1}{2}$, $0 < \delta < \frac{1}{2}$,
if L is given access to $EX(c, D)$ and inputs ϵ, δ, then,
with probability at least $1 - \delta$, L outputs $h \in \mathcal{C}$ satisfying $error(h) \leq \epsilon$.
The running time is polynomial in $1/\epsilon, 1/\delta, n, size(c)$.

Comments

- error parameter ϵ,
- confidence parameter δ,
- L must perform well w.r.t. any distribution D,
- $error(h)$ is evaluated w.r.t. the same distribution D,
- n implicitly given by $EX(c, D)$, $size(c)$ can already be polynomial in n.

Conjunctions are efficiently properly PAC learnable (1)

Consider the representation class of all conjunctions of literals over $x_1, \ldots, x_n$. Note that $size(c) \leq 2n$.

Algorithm

input : ϵ, δ

init : $h = x_1 \wedge \overline{x_1} \wedge \ldots \wedge x_n \wedge \overline{x_n}$

for $j = 1$ to m

if $EX(c, D)$ returns a positive example $< \mathbf{a}, 1 >$ then

for $i = 1$ to n, if $a_i = 1$ delete $\overline{x_i}$ else delete x_i from h

output : h

Example

Let $n = 4$, $c = x_1 \wedge \overline{x_3}$. $h = x_1 \wedge \overline{x_1} \wedge \ldots \wedge x_4 \wedge \overline{x_4}$. No example satisfies h.

Receive $< 1100, 1 >$, $h = x_1 \wedge x_2 \wedge \overline{x_3} \wedge \overline{x_4}$.

Receive $< 0111, 0 >$, h unchanged.

Receive $< 1000, 1 >$, $h = x_1 \wedge \overline{x_3} \wedge \overline{x_4}$.

Conjunctions are efficiently properly PAC learnable (2)

Analysis

- Literals of h always include those of c, h is more specific than c.
- Let z be a literal that occurs in h but not in c. Let $p(z)$ be the total probability of instances that would have caused our algorithm to delete z from h : $p(z) = Pr_{\mathbf{a}\sim D}[c(\mathbf{a}) = 1 \wedge z \text{ is 0 in } \mathbf{a}]$.
- A literal is bad if $p(z) \geq \frac{\epsilon}{2n}$. If h contains no bad literal then $error(h) \leq \epsilon$. It remains to upper bound the probability that a bad litteral will appear in h.
- For any fixed bad literal z, the probability that z is not deleted from h after m calls to $EX(c, D)$ is at most $(1 - \frac{\epsilon}{2n})^m$. Independent events
- The probability that there is some bad literal that is not deleted from h after m calls to is at most $2n(1 - \frac{\epsilon}{2n})^m$. Union bound
- $2n(1 - \frac{\epsilon}{2n})^m \leq \delta$ yields $m \geq \frac{2n}{\epsilon}(\log(2n) + \log(\frac{1}{\delta}))$. If our algorithm takes at least this number of examples, the PAC conditions are verified and the running time is polynomial (each example is processed in linear time).

3-term DNF are not efficiently properly PAC learnable

unless $NP \not\subset BPP$

A language A is in BPP if there is a randomized algorithm taking as input any string u and confidence parameter δ, and that, with probability at least $1-\delta$ determines whether $u \in A$ or $u \notin A$ in time polynomial in $|u|$ and $\frac{1}{\delta}$.

The concept class of 3-term DNF formulae

is the set of all disjunctions $T_1 \vee T_2 \vee T_3$ of conjunctions of literals over $x_1, \ldots, x_n$. Note that $size(c) \leq 6n$.

Sketch of proof

- An NP-complete language (graph 3-coloring problem),
- Map any u to a sample S_u of polynomial size such that $u \in A$ if and only if S_u is consistent with some c (a 3-term DNF)
- Given a PAC learning algorithm L, one can run L on S_u (with D uniform on S_u and adequate ϵ) to determine with high probability if $u \in A$ or $u \notin A$. every D

3-term DNF are efficiently PAC learnable using 3-CNF

The concept class of 3-CNF formulae

is the set of all CNF formulae over $x_1, \ldots, x_n$ in which each clause contains at most 3 litterals. And $T_1 \vee T_2 \vee T_3 \equiv \bigwedge_{z_i \in T_i}(z_1 \vee z_2 \vee z_3)$.

Reduction to learning conjuctions

- transform each random example of a target 3-CNF in an example of a conjunction. Introduce y_{z_1,z_2,z_3} whose value is $y_{z_1,z_2,z_3} = z_1 \vee z_2 \vee z_3$,
- use the learning algorithm for conjunctions,
- convert the resulting conjunction in a 3-CNF formula.

PAC model

Definition 2

$\mathcal{C}$ is efficiently PAC learnable using $\mathcal{H}$ if there exists an algorithm L such that :
for every distribution D, $0 < \epsilon < \frac{1}{2}$, $0 < \epsilon < \frac{1}{2}$,
if L is given access to $EX(c, D)$ and inputs ϵ, δ, then,
with probability at least $1 - \delta$, L outputs $h \in \mathcal{H}$ satisfying $error(h) \leq \epsilon$.
The running time is polynomial in $1/\epsilon, 1/\delta, n, size(c)$.

Summarizing results

- 1-term DNF are efficiently properly learnable
- k-term DNF ($k \geq 2$) are not efficiently properly learnable
- k-term DNF are efficiently learnable using k-CNF

Overview of results

- Information theoretic results. Let $\mathcal{A}$: draw a random sample of $O(\frac{1}{\epsilon}\log(\frac{|\mathcal{H}|}{\delta}))$ examples and find a consistent hypothesis. Then $\mathcal{A}$ satisfies the PAC criteria (except time complexity). If $\mathcal{H}$ is infinite use the VC dimension and $O(\frac{1}{\epsilon}\log(\frac{1}{\delta}) + \frac{d}{\epsilon}\log(\frac{1}{\epsilon}))$ examples.
- Weak learning = strong learning leads to boosting algorithms.
- Untractability results.
  Are there classes of concepts that are hard to learn regardless the hypothesis class used ?
  Are there classes of concepts of polynomial VC dimension that are not PAC learnable using any polynomially evaluable hypothesis class $\mathcal{H}$?
  YES : the class of polynomial Boolean circuits, the class of polynomial size Boolean formulae, the class of deterministic finite automata are inherently unpredictable.
- Active Learning = learning with queries

Plan

Preliminaries

Concepts

- Let $X = \{0,1\}^n$ be the instance space.
- A concept c is a subset of X or a function from X into $\{0,1\}$, a function class $\mathcal{C}$ is a set of functions.

Measuring errors

- Let c be a function, let D be a probability distribution over X, $EX(c,D)$ is a procedure that, on each call, returns a labelled example $< \mathbf{a}, c(\mathbf{a}) >$ where $\mathbf{a}$ is drawn randomly and independently according to D. The error between c and an hypothesis function h is $error(h) = Pr_{\mathbf{a} \sim D}[c(\mathbf{a}) \neq h(\mathbf{a})]$.
-
  - c est ϵ-close from $\mathcal{C}$ if exists $h \in \mathcal{C}$ such that $error(h) \leq \epsilon$.
  - Otherwise c est ϵ-far from $\mathcal{C}$

Property Testing

Definition

$\mathcal{A}$ is a property testing algorithm for $\mathcal{C}$ if for every n, for every distribution D, $0 < \epsilon < \frac{1}{2}$, $0 < \delta < \frac{1}{2}$, for every $c \in \mathcal{C}$, if $\mathcal{A}$ is given access to $EX(c, D)$ and inputs n, ϵ, δ, then,

- if $c \in \mathcal{C}$, then with probability at least $1 - \delta$, $\mathcal{A}$ accepts c ;
- if c is ϵ-far from $\mathcal{C}$, then with probability at least $1 - \delta$, $\mathcal{A}$ rejects c ;

The sample complexity and time complexity depend on (n, ϵ, δ).

Remarks and variants

- δ can be given a fixed value, i.e. $\delta = \frac{1}{3}$;
- when c is $\frac{\epsilon}{2}$-close from $\mathcal{C}$, $\mathcal{A}$ may accept or reject.
- Property testing w.r.t. a specific D, a class of distributions.
- Property testing with queries.
- One-sided error : if $c \in \mathcal{C}$, $\mathcal{A}$ accepts c.

Plan

The differences

Proper PAC learning

if there exists an algorithm L such that : for every distribution D, ϵ, δ, if L is given access to $EX(c, D)$ and inputs ϵ, δ, then, with probability at least $1 - \delta$, L outputs $h \in \mathcal{C}$ satisfying $error(h) \leq \epsilon$.

Property testing

if there exists an algorithm $\mathcal{A}$ such that : for every distribution D, ϵ, δ, if $\mathcal{A}$ is given access to $EX(c, D)$ and inputs n, ϵ, δ, then, if $c \in \mathcal{C}$, then, with probability at least $1 - \delta$, $\mathcal{A}$ accepts c ; if c is ϵ-far from $\mathcal{C}$, then with probability at least $1 - \delta$, $\mathcal{A}$ rejects c.

Differences

- Testing harder : the target function is known to be in $\mathcal{C}$ for PAC ;
- Learning harder : an hypothesis function h is required in PAC.

Testing is not harder than proper learning

Theorem

If a function class $\mathcal{C}$ has a polynomial time proper learning algorithm L, then $\mathcal{C}$ has a polynomial time property testing algorithm $\mathcal{A}$ with sample complexity $m_L(n, \frac{\epsilon}{2}, \frac{\delta}{2}) + O(\frac{\log(1/\delta)}{\epsilon})$.

Algorithm

1 run $L(\frac{\epsilon}{2}, \frac{\delta}{2})$
2 If L does not output a hypothesis h then reject
3 Otherwise make $O(\frac{\log(1/\delta)}{\epsilon})$ calls to $EX(c, D)$ and compute $\hat{err}or(h)$
4 If $\hat{err}or(h) < \frac{3\epsilon}{4}$ then accept else reject

Proof - hints

If $c \in \mathcal{C}$, with probability at least $1 - \frac{\delta}{2}$, L's output is $\frac{\epsilon}{2}$-close from c, and our test set will not reject it with probability at least $1 - \frac{\delta}{2}$;
If c is ϵ-far from $\mathcal{C}$, any hypothesis $h \in \mathcal{C}$ is at least ϵ-far from c, and our test set will reject it with probability at least $1 - \frac{\delta}{2}$; Chernoff, VC.

Testing may be harder than proper learning

Theorem

If $NP \not\subset BPP$, then there exist function classes that are not poly($\frac{n}{\epsilon}$)-time testable but are poly($\frac{n}{\epsilon}$)-time (non properly) PAC-learnable.

Proof

Indeed, the proof that 3-term DNF are not efficiently properly PAC learnable is already a proof that 3-term DNF are not efficiently testable. Thus, 3-term DNF are not efficiently properly PAC learnable can be deduced from the previous theorem. every D.

Testing may be easier than learning

Theorem

There exist function classes $\mathcal{C}$ such that :

- $\mathcal{C}$ has a property testing algorithm whose sample and time complexity are $O(\frac{1}{\epsilon}\log(\frac{1}{\delta}))$ (independent of n) ;
- Any learning algorithm for $\mathcal{C}$ must have sample complexity exponential in n.

Proof

Let $\mathcal{C}_n = \{f \mid \text{if the first bit of } \mathbf{x} \text{ is 1 then } f(\mathbf{x}) = 1\}$. The algorithm $\mathcal{A}$ is :
1 make $O(\frac{1}{\epsilon}\log(\frac{1}{\delta}))$ calls to $EX(c, D)$
2 if for all examples with first bit 1 $c(\mathbf{x}) = 1$ then accept else reject
If $c \in \mathcal{C}_n$, $\mathcal{A}$ always accept it
If c is ϵ-far from $\mathcal{C}_n$, the probability that $\mathcal{A}$ does not observe an example of the form $(1 \ldots, 0)$ is bounded by $(1-\epsilon)^n < \delta$.
The VC-dimension of $\mathcal{C}_n$ is 2^{n-1}, then the sample complexity is exponential.

Plan

Testing membership in regular languages

let L be a language over alphabet A. A word u of length n is ϵ-far from L if no word in L differs from u in no more than ϵn positions.

A property testing algorithm for membership in L

is a randomized algorithm, which given n and given access to the symbol in any position of an input word, distinguishes with probability at least $\frac{2}{3}$ between the case $u \in L$ and the case u is ϵ-far from L.

What about our previous definition ?

- A word u of length n is a function from $[n]$ into A, f,
- A language L is $\cup_n L_n$ and each L_n is a set of functions, $\mathcal{C}$, $\mathcal{C}_n$,
- $|v| = n$, $error(v) = \#\{i \mid u(i) \neq v(i)\} = Pr_{i \sim U}[u(i) \neq v(i)] \times n$,
- $\frac{2}{3} = \delta$.

$L = \cup_n L_n$, U uniform on $[n]$, same length for comparison.

Monotonicity Tester

Problem

Determine whether a list $\mathbf{x} = (x_1, \ldots, x_n)$ of integers is monotone.
We suppose we can access x_i in one computation step for any i.
There is no sub-linear time exact algorithm for the problem

A sub-linear time testing algorithm

1 let $l = \frac{c}{\epsilon}$ with adequate c ; choose $i_1, \ldots, i_n$ uniformly from $[n]$;
2 for each i_j, do a binary search to determine whether x_{i_j} is present in $\mathbf{x}$;
3 output reject if one of the binary search fails to find any x_{i_j} in location i_j or finds a pair of out-of-order elements along the search.
The algorithm runs in $O(\frac{1}{\epsilon} \log n)$ and outputs accept if $\mathbf{x}$ is monotone and outputs reject if $\mathbf{x}$ is ϵ-far from a monotone list.
$\mathbf{x}$ is ϵ-close from a monotone list if by changing at most ϵn values of the x_i's one can transform $\mathbf{x}$ in a monotone list.

Connectivity Tester

Problem

Let G be an undirected graph on n nodes and degree bounded by d.
$g(u, i)$ is the i-th neighbor of vertex u, if it exists, and 0 otherwise.
$error(H)$ is the fraction of places where G and H disagree.
G is (ϵ, d)-far from being connected if its distance from every connected graph of bounded degree d is at most ϵ.

A sub-linear time testing algorithm

1 pick $\frac{16}{\epsilon d}$ vertices at random in G ;
2 for each vertex s, perform a Breadth First Search starting from s ;
3 output reject if one of BFS search encountered less than $\frac{8}{\epsilon d}$ vertices otherwise output accept.
The algorithm runs in $O(\frac{1}{(\epsilon^2 d)})$ time and outputs accept if G is connected and outputs reject if G is (ϵ, d)-far from being connected.

Plan

Plan

- Works on PAC and Property Testing are rather old.
- The general property testing model is not used in practice
- Is the property testing model for membership the right one ?