PAC learning

PAC learning

Invented by L.Valiant in 1984

L.G.Valiant *A theory of the learnable*, Communications of the ACM, 1984, vol 27, 11, pp. 1134-1142.

PAC learning

Basic notions

X - instance space (in general, an arbitrary set)
$c \subseteq X$ - concept (each subset of X is a concept)
$C \subseteq \{c | c \subseteq X\}$ - class of concepts to be learned
$T \in C$ - the unknown concept to be learned

Examples

X	$\mathbf{R}^2$	$\{0,1\}^n$
C	set of rectangles	CNF with n variables
T	a given rectangle	a given CNF

It may be more convenient for the learning algorithm to represent T in other way, and not simply as a subset of X.

PAC learning

Learning algorithm - inputs

P - fixed probability distribution defined on X

Learning algorithm receives a sequence of examples

$(x_0,v_0),(x_1,v_1),(x_2,v_2),(x_3,v_3),...$

where $x_i \in X$ and v_i = "+", if $x_i \in T$, v_i = "–", if $x_i \notin T$, and the probability that x_i appears as a current element in the sequence is in accordance with P.

ε - accuracy parameter

δ - confidence parameter

Learning algorithm - inputs

A model where all v_i = "+" can be considered, in which case we say that an algorithm is learning from positive examples.

In general case we can say that algorithm is learning from positive and negative examples.

A learning from only negative examples can also be considered.

Learning algorithm - outputs

After the processing a finite sequence of inputs a learning algorithm outputs a concept $S \in C$.

$S \oplus T$ - the symmetric difference of S and T

$P(S \oplus T)$ - the error rate of concept S, i.e. the probability that T and S classify a random example differently.

Learnability - definition

A concept S is *approximately correct*, if $P(S \oplus T) \leq \varepsilon$

The learning algorithm is *probably approximately correct*, if the probability that the output S is approximately correct is at least $1-\delta$

In this case we say that a learning algorithm *pac-learns* the concept class C, and that the concept class C is *pac-learnable*

Polynomial learnability 1

A learning algorithm L is a *polynomial PAC-learning algorithm* for C, and C is *polynomially PAC-learnable*, if L PAC-learns C with time complexity (and sample complexity) which are polynomial in $1/\varepsilon$ and $1/\delta$.

It is useful to consider similar classes of concepts with different sizes n, and require polynomial complexity also in n, but then we must focus on specific instance spaces dependent from parameter n (eg. $X= \{0,1\}^n$).

Polynomial learnability 2

We consider $\mathbf{C} = \{(X_n, C_n) | n>0\}$

A learning algorithm L is a *polynomial PAC-learning algorithm* for **C**, and **C** is *polynomially PAC-learnable*, if L PAC-learns **C** with time complexity (and sample complexity) which are polynomial in n, $1/\varepsilon$ and $1/\delta$.

Learnability - simple observation

The (polynomial) PAC-learnability from positive (or negative) examples implies (polynomial) PAC-learnability from positive andnegative examples.

The converse may not be necessarily true.

Valiant's results - k-CNF

$X = \{0,1\}^n$ (n boolean variables)

C = set of k-CNF formulas (ie, CNF formulas with at most k literals in each clause)

Theorem

For any positive integer k the class of k-CNF formulas is polynomially PAC-learnable from positive examples.

(Even when partial examples (with some variables missing) are allowed.)

PAC learning

Valiant's results - k-CNF - function L(r,m)

Definition

For any real number r >1 and for any positive integer m the value L(r,m) is the smallest integer y, such that in y Bernoulli trials with probability of success at least 1/r, the probability of having fewer than m successes is less than 1/r.

Proposition

$L(r,m) \leq 2r(m+\ln r)$

Valiant's results - k-CNF - algorithm

- start with formula containing all possible clauses with k literals (there are less than $(2n)^{k+1}$ such clauses)

- when receiving current example x_i, delete all clauses not consistent with x_i

Lemma

The algorithm above PAC-learns the class of k-CNF formulas from $L(h,(2n)^{k+1})$ examples, where $h = \max\{1/\varepsilon, 1/\delta\}$.

Valiant's results - k-CNF - algorithm

g - the initial k-CNF
f(i) - the k-CNF produced by algorithm after i examples

Consider $P(g \oplus f(i))$

At each step $P(g \oplus f(i))$ can only decrease, the probability that this will happen will be exactly $P(g \oplus f(i))$

After $l=L(h,(2n)^{k+1})$ examples there may be two situations:

- $P(g \oplus f(l)) \leq 1/h$ OK, f(l) is h approximation
- $P(g \oplus f(l)) > 1/h$

Valiant's results - k-CNF - algorithm

After $l=L(h,(2n)^{k+1})$ examples there may be two situations:

- $P(g \oplus f(l)) \leq 1/h$ OK, f(l) is h approximation
- $P(g \oplus f(l)) > 1/h$

The probability of the second situation is at most 1/h
(we have $L(h,(2n)^{k+1})$ Bernoulli experiments with probability at least 1/h of success for each)

PAC learning

Valiant's results - k-term-DNF

$X = \{0,1\}^n$ (n boolean variables)

C = set of k-term-DNF formulas (ie, DNF formulas with at most k monomials)

Theorem

For any positive integer k the class of k-DNF formulas is polynomially PAC-learnable.

(Only, when total examples (with all variables present) are allowed.)

PAC learning

VC (Vapnik - Chervonenkis) dimension

C - nonempty concept class

$s \subseteq X$ - set

$\Pi_C(s) = \{s \cap c \mid c \in C\}$ - all subsets of s, that can be obtaining by intersecting s with a concept from c

We say that s is *shattered* by C, if $\Pi_C(s) = 2^s$.

The VC (Vapnik-Chervonenkis) dimension of C is the cardinality of the largest finite set of points $s \subseteq X$ that is shattered by C (if arbitrary large finite sets are shattered, we say that VC dimension of C is infinite).

VC dimension - examples

Example 1

$X = \mathbf{R}$
C = set of all (open or closed) intervals

If $s = \{x_1, x_2\}$, then there exist $c_1, c_2, c_3, c_4 \in C$, such that $c_1 \cap s = \{x_1\}$, $c_2 \cap s = \{x_2\}$, $c_3 \cap s = \emptyset$, and $c_{14} \cap s = s$.

If $s = \{x_1, x_2, x_3\}$, $x_1 \leq x_2 \leq x_3$, then there is no concept $c \in C$, such that $x_1 \in c$, $x_3 \in c$ and $x_2 \notin c$. Thus the VC dimension of C is 2.

VC dimension - examples

Example 2

C = any finite concept class

It requires 2^d distinct concepts to shatter a set of d points, therefore no set of cardinality larger that log|C| can be shattered. Hence the VC dimension of C is at most log |C|.

VC dimension - examples

Example 3

$X = \mathbf{R}$

C = set of all finite unions of (open or closed) intervals

Any finite $s \in X$ can be shattered, thus the VC dimension of C is infinite.

VC dimension - relation to PAC

Theorem [A.Blumer, A.Ehrenfeucht, D.Haussler, M.Warmuth]

C is PAC-learnable if and only if the VC dimension of C is finite.

Theorem also gives an upper and lower bounds of number of examples needed for learning.

Learnability and the Vapnik-Chervonenkis dimension, Journal of the ACM, vol. 36, 4, 1989, pp. 929-965.

VC dimension - relation to PAC

Let d be the VC dimension of C, then:

- no algorithm can PAC-learn class C with less than $\max((1-\varepsilon)/\varepsilon \ln 1/\delta, d(1 - 2(\varepsilon(1-\delta)+\delta)))$ examples.

- any consistent algorithm can PAC-learn class C with $\max(4/\varepsilon \log 2/\delta, 8d/\varepsilon \log 13/\varepsilon)$ examples.

PAC learning

VC dimension - relation to PAC - example

$X = R^2$

C = set of all triangles in X

VC dimension of C is 4, thus C is PAC-learnable.

PAC learning

VC dimension - relation to polynomial PAC

We consider **C** = $\{(X_n, C_n) | n>0\}$

A *randomised polynomial hypothesis finder* (*RPHF*) for **C** is a randomised polynomial time algorithm that takes as input a sample of a concept in **C** and for some $\gamma > 0$, with probability at least γ produces a hypothesis in **C** that is consistent with this sample.

γ is called the *success rate* of RPHF.

PAC learning

VC dimension - relation to polynomial PAC

Theorem [A.Blumer, A.Ehrenfeucht, D.Haussler, M.Warmuth]

For any concept class **C** = $\{(X_n, C_n) | n>0\}$, **C** is polynomially PAC-learnable in hypothesis space **C**, if and only if there is an RPHF for **C** and the VC dimension of C_n grows polynomially in n.

PAC learning

VC dimension - relation to polynomial PAC

Example

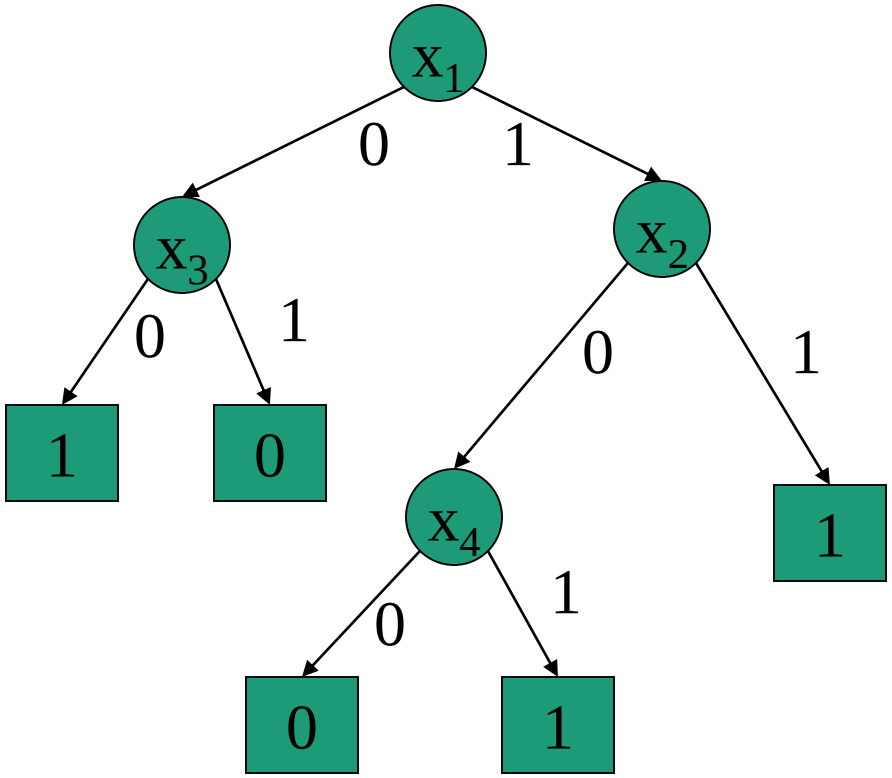
C_n = class of all axis-parallel rectangles in $\mathbf{R}^n$.

VC of C_n does not exceed 2n.

$2n/\varepsilon \ln(2n/\delta)$ examples are sufficient (or use the upper bound from the previous theorem).

PAC learning

Learning of decision lists - decision trees



Decision tree for formula $\neg x_1 \neg x_3 + x_1 \neg x_2 x_4 + x_1 x_2$

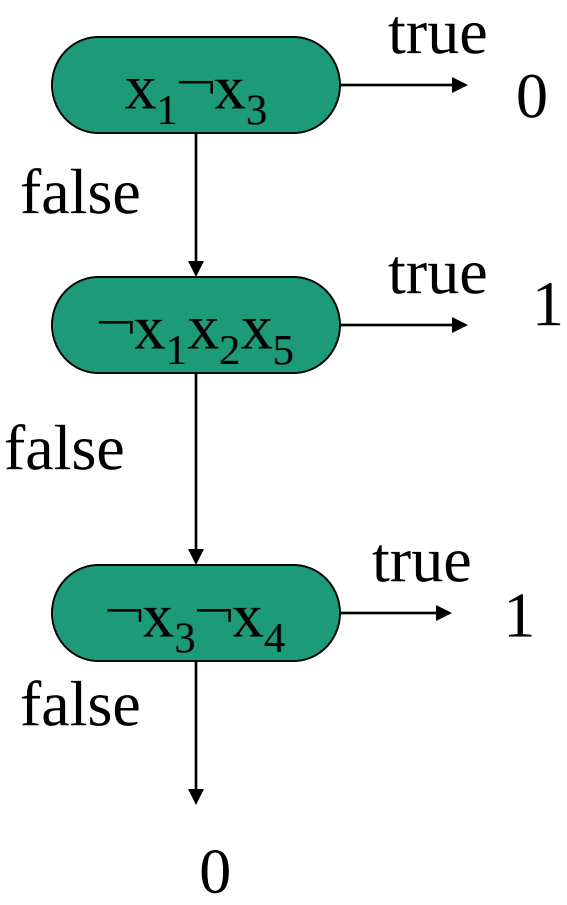
PAC learning

Learning of decision lists - decision trees

Let k-DT be the set of all boolean functions defined by decision trees of depth at most k

PAC learning

Learning of decision lists - decision lists



PAC learning

Learning of decision lists - decision lists

Let k-DL be the set of all boolean functions defined by decision lists containing clauses contatining at most k literals.

Theorem [R.Rivest]

For any positive integer k the class of k-DL formulas is polynomially PAC-learnable.

Some proper subsets of class k-DL

k-CNF
k-DNF
k-CNF∪k-DNF
k-clause-CNF
k-term-DNF
k-DT

k-clause-CNF and k-term-DNF are not learnable in their "own" hypotheses space!

Some other facts about k-DL

Whilst k-DL are polynomially PAC-learnable, some subclasses may be learnable more efficiently in their "own" hypotheses space (eg. k-DNF).

It is likely that unbounded DT are not polynomially PAC-learnable (the converse will imply some NP related result that is unlikely to be true).