MACHINE LEARNING - CS671 - Part 1

Prof. Dan A. Simovici

UMB

Mitchell's Definition

Machine Learning (ML) was defined in [3] as the discipline that

> *aims to construct computer programs that learn from experience with respect to some class of tasks and performance measure*,

if its performance improves with experience.

- An essential feature of ML algorithms is their capability for *generalization*, defined as the *ability of an algorithm to perform accurately on new, unseen examples after having trained on a learning data set*.
- Typically, training examples come from some generally unknown probability distribution and the ML program has to extract from them something more general that allows it to produce useful predictions in new cases.

ML and DM

ML is a foundational discipline for Data Mining:

- the focus of ML is on prediction, based on knowledge learned from the training data;
- Data Mining focuses on efficient algorithms for discovery of unknown properties on the data.

A sequence of length n on S is a function

$$s : \{0, \ldots, n-1\} \longrightarrow S.$$

We denote such a sequence by $(s(0), s(1), \ldots, s(n-1))$.

- For $n \geqslant 0$, the set of sequences of length n of elements of S is denoted by $\mathbf{Seq}_n(S)$.
- The set of all sequences of S is denoted by

$$\mathbf{Seq}(S) = \bigcup_{n \geqslant 0} \mathbf{Seq}_n(S).$$

- Occasionally, when $n \geqslant 1$, we shall denote a sequence of length n as $(s(1), \ldots, s(n))$.

Example

For $S = \{0, 1\}$, we have

$$\begin{aligned} \mathbf{Seq}_3(S) &= \{(0,0,0),(0,0,1),(0,1,0),(0,1,1), \\ &\quad (1,0,0),(1,0,1),(1,1,0),(1,1,1)\}. \end{aligned}$$

If $\mathbb{R}$ is the set of real numbers, $(0.4, -7.1, 88, -9) \in \mathbf{Seq}_4(\mathbb{R})$.

If S is a finite set and $|S| = m$, then there exist n^m sequences of length n. In particular, there exists exactly one sequence of length 0 and this is denoted by () or by λ.

Functions of the form $f : \{0,1\}^n \longrightarrow \{0,1\}$ are referred to as Boolean functions.

Propositional formulas on a set of variables $V = \{x_1, \ldots, x_n\}$ are defined as follows:

- for each $x_i \in V$, x_i is a formula;
- if ϕ, ψ are formulas, then $\overline{\phi}$, $(\phi \wedge \psi)$, and $(\phi \vee \psi)$ are formulas.

A formula of the form x_i or $\overline{x_i}$ is referred to as a *literal*.

The Boolean function defined by a formula ϕ is denoted by $\langle\phi\rangle$, and we have $\langle\phi\rangle : \{0,1\}^n \longrightarrow \{0,1\}$. There are 2^{2^n} such functions defined inductively as follows:

- for every variable x_i, $1 \leqslant i \leqslant n$ we have

$$\langle x_i\rangle(x_1,\ldots,x_n) = \begin{cases} 1 & \text{if } x_i = 1, \\ 0 & \text{otherwise;} \end{cases}$$

- $$\langle(\phi \wedge \psi)\rangle(x_1,\ldots,x_n) = \begin{cases} 1 & \text{if } \langle\phi\rangle(x_1,\ldots,x_n) = 1 \\ & \text{and } \langle\psi\rangle(x_1,\ldots,x_n) = 1, \\ 0 & \text{otherwise;} \end{cases}$$

- $$\langle(\phi \vee \psi)\rangle(x_1,\ldots,x_n) = \begin{cases} 1 & \text{if } \langle\phi\rangle(x_1,\ldots,x_n) = 1 \\ & \text{or } \langle\psi\rangle(x_1,\ldots,x_n) = 1, \\ 0 & \text{otherwise;} \end{cases}$$

- $$\langle\overline{\phi}\rangle(x_1,\ldots,x_n) = \begin{cases} 1 & \text{if } \langle\phi\rangle(x_1,\ldots,x_n) = 0, \\ 0 & \text{if } \langle\phi\rangle(x_1,\ldots,x_n) = 1 \end{cases}$$

Notational Conventions

- the top priority among the logical connective is assigned to the negation, followed by the conjunction, and having $\vee$ with the lowest priority;
- We may omit the symbol "$\wedge$" when writing a formula $(\phi \wedge \psi)$ and write just $\phi\psi$, if there is no risk of confusion;
- both $\vee$ and $\wedge$ are associated to the left; this means that instead of writing $(((\phi_1 \vee \phi_2) \vee \phi_3) \vee \phi_4)$ we shall write $\phi_1 \vee \phi_2 \vee \phi_3 \vee \phi_4$.

Specification of Boolean Functions

Example

The function $f : \{0,1\}^3 \longrightarrow \{0,1\}$ that is 1 if and only if an odd of arguments is 1 is defined by the table:

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

- The *conjunction* of the formulas $\phi_1, \ldots, \phi_k$ is the formula $\phi_1 \phi_2 \ldots \phi_k$.
- The *disjunction* of the same is the formula $\phi_1 \vee \phi_2 \vee \cdots \vee \phi_k$.
- A *monomial* is a conjunction of literals $\ell_1, \ldots, \ell_k$, that is $\ell_1 \cdots \ell_k$.
- A *clause* is a disjunction of literals $\ell_1, \ldots, \ell_k$, that is $\ell_1 \vee \cdots \vee \ell_k$.
- The set of monomials over m variables is denoted by MON_m; the set of monomials over m variables that contain at most p literals is denoted by $\text{MON}_{m,p}$. The set of disjunctions of monomials in $\text{MON}_{m,p}$ is denoted by $D_{m,p}$.

A *disjunctive normal form* is a formula

$$\mu_1 \vee \mu_2 \vee \cdots \vee \mu_k,$$

where $\mu_1, \ldots, \mu_k$ are monomials. Similarly, a *conjunctive normal form* is a formula

$$\kappa_1 \wedge \kappa_2 \wedge \cdots \wedge \kappa_k,$$

where $\kappa_1, \ldots, \kappa_k$ are clauses.
The tabular definition of a Boolean function yields a formula in disjunctive normal form. For example, starting from the table given above we obtain the disjunctive normal form:

$$\overline{x_1 x_2} x_3 \vee \overline{x_1} x_2 \overline{x_3} \vee x_1 \overline{x_2 x_3} \vee x_1 x_2 x_3.$$

Two formulas ϕ, ψ are said to be equivalent if $\langle \phi \rangle = \langle \psi \rangle$. This is denoted by $\phi \equiv \psi$.

Example

For any formulas ϕ, ψ we have

$$\overline{\phi \vee \psi} \equiv \overline{\phi} \wedge \overline{\psi} \text{ and } \overline{\phi \wedge \psi} \equiv \overline{\phi} \vee \overline{\psi}.$$

These equivalences are known as the *De Morgan laws*.

Disjunctive Normal Forms

Starting from the table of a Boolean function f, where $f = \langle \phi \rangle$ we can compute the table of the function g represented by the formula $\overline{\phi}$ by reversing the values of the column labeled by y. For instance, starting from the table given before we obtain the table

x_1	x_2	x_3	$g(x_1, x_2, x_3)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

The conjunctive normal form that corresponds to this function is

$$\overline{x_1 x_2 x_3} \wedge \overline{x_1} x_2 x_3 \wedge x_1 \overline{x_2} x_3 \wedge x_1 x_2 \overline{x_3}.$$

Negating this formula yields a disjunctive normal form for f:

$$(x_1 \vee x_2 \vee x_3)(x_1 \vee \overline{x_2} \vee \overline{x_3})(\overline{x_1} \vee x_2 \vee \overline{x_3})(\overline{x_1} \vee \overline{x_2} \vee x_3).$$

Let X be a set of sequences in $\mathbf{Seq}(S)$. The set X is the *example space*.

- A *concept* is a partial function $c : X \rightsquigarrow \{0, 1\}$.
- A *positive example* is an example x such that $c(x) = 1$; if $c(x) = 0$, then we say that x is a *negative example*. The set of positive examples of c in the example space X is denoted by $POS_X(c)$; the set of negative examples is $NEG_X(c)$.
  $\text{Dom}(c) = POS_X(c) \cup NEG_X(c) \subseteq X$.
- Concepts correspond to pairs of subsets of X of the form (P, N), where P is the set of positive examples and N is the set of negative examples.

Example

Let $p_n : \mathbf{Seq}(\mathbb{R}) \rightsquigarrow \{0,1\}$ be the concept defined by

$$POS_{\mathbf{Seq}(\mathbb{R})}(p_n) = \{(x_1, \ldots, x_n, r) \in \mathbb{R}^{n+1} \mid x_1^2 + \cdots + x_n^2 = r^2\},$$

and $NEG_{\mathbf{Seq}(\mathbb{R})}(p_n) = \mathbf{Seq}(\mathbb{R}) - POS_{\mathbf{Seq}(\mathbb{R})}(p)$. This concept represents the set of n-dimensional spheres.

Example

Let X be the set of examples that consists of triplets of natural numbers and let $p_t : \mathbf{Seq}_3(\mathbb{N}) \rightsquigarrow \{0, 1\}$ be the concept defined by

$$POS_{\mathbf{Seq}_3(\mathbb{N})}(p_t) = \{(\ell, m, n) \in \mathbf{Seq}_3(\mathbb{N}) \mid \ell^2 + m^2 = n^2\},$$

and

$$NEG_{\mathbf{Seq}_3(\mathbb{N})}(p_t) = \{(\ell, m, n) \in \mathbf{Seq}_3(\mathbb{N}) \mid |\ell - m| \leqslant n \leqslant \ell + m, \ell^2 + m^2 \neq n^2\}.$$

$\ell^2 + m^2 = n^2$ implies that $|\ell - m| \leqslant n \leqslant \ell + m$.
The domain of p_t consists of all triplets of natural numbers that can be the lengths of the sides of a triangle. The positive examples that define the concept p_t are the triplets of numbers that can be the lengths of the sides of a right triangle.

Example

Let S be a finite and non-empty set, referred to in this context as an *alphabet*. Define $p : \mathbf{Seq}_k(S) \rightsquigarrow \{0, 1\}$ by

$$p((s_1, \ldots, s_k)) = \begin{cases} 1 & \text{if } s_i = s_{k-i} \text{ for } 1 \leq i \leq \lfloor \frac{k}{2} \rfloor \\ 0 & \text{otherwise.} \end{cases}$$

$p(w) = 1$ iff $w = (s_1, \ldots, s_k)$ and the symbols s_i and s_{k-i} that occupy symmetric positions in w are equal. Such a sequence is known as a *palindrome*.

$POS_{\mathbf{Seq}_k(S)}(p)$ consists of all palidromes that can be written on the alphabet S and $NEG_{\mathbf{Seq}_k(S)}(p) = \mathbf{Seq}_k(S) - POS_{\mathbf{Seq}_k(S)}(p)$.

For instance, if $S = \{s_1, s_2, s_3\}$ and $k = 5$, the sequence

$$(s_2, s_1, s_3, s_1, s_2)$$

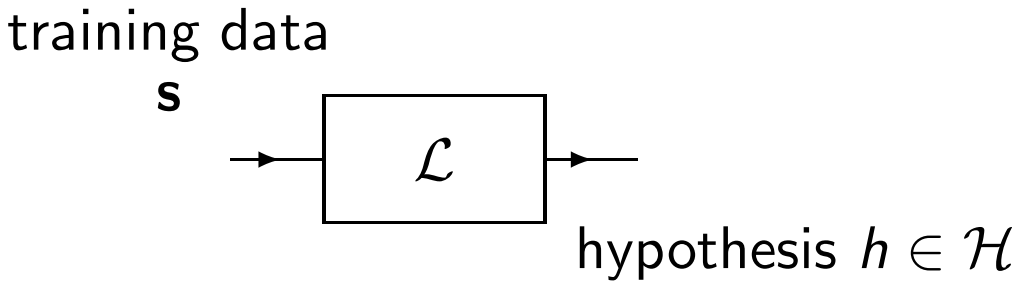
is a positive example of the palindrome concept.

Task of Learning Algorithms

The task of a *learning algorithm* is to produce a concept in a *concept space* $\mathcal{C}$ starting from a sequence of examples of the concept referred to as a *sample*.
We need to limit the class of concepts that we are searching for to a subspace of the concept space known as the *hypothesis* space $\mathcal{H}$ that consists of concepts that can be computed in a reasonable time.
Thus, the central problem of ML can be stated as follows: for a concept $c \in \mathcal{C}$ find a hypothesis $h \in \mathcal{H}$ which is an approximation of c.

Learning Algorithms

A hypothesis h for a concept c is formed by feeding a sequence of examples (positive and negative) $\mathbf{s}$ of c to a learning algorithm $\mathcal{L}$.



Samples

Definition

A *sample of length m*, where $m \geqslant 1$, is a sequence

$$\mathbf{s} = ((\mathbf{x}_1, b_1), \ldots, (\mathbf{x}_n, b_n))$$

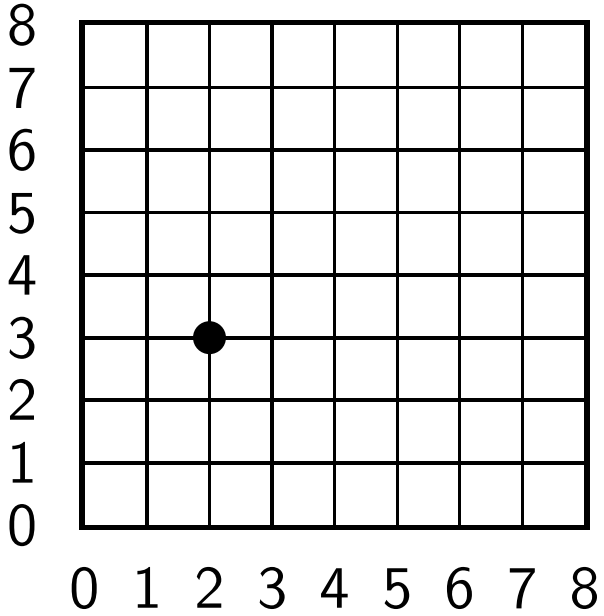
that satisfies the coherence condition: $\mathbf{x}_i = \mathbf{x}_j$ implies $b_i = b_j$ for $1 \leqslant i, j \leqslant m$.

The sample $\mathbf{s} = ((\mathbf{x}_1, b_1), \ldots, (\mathbf{x}_n, b_n))$ is a training example for a target concept c if $b_i = c(x_i)$ for $1 \leqslant i \leqslant m$.

A hypothesis h is *consistent* with $\mathbf{s}$ if $h(\mathbf{x}_i) = b_i$ for $1 \leqslant i \leqslant m$.

Learning Rectangles on a Grid

Consider a 9×9 grid. containing 81 points. There are $2^{81} \approx 10^{27}$ possible shapes that we can draw with these points and $2^{2^{81}} \approx 10^{3 \cdot 10^{26}}$ families of shapes. For comparison, the number of atoms in the observable universe is about 10^{80}!



The hypothesis space

is the set of rectangles on the grid, that is as the set having the form

$$\{p, \ldots, q\} \times \{u, \ldots, v\},$$

where $p \leqslant q$, $u \leqslant v$, and $p, q, u, v \in \{0, \ldots, 9\}$. We denote such a rectangle by $[p, u; q, v]$, where (p, u) are the coordinates of the southwestern corner of the rectangle and (q, v) are the coordinates of the northeastern corner. For a rectangle $R = [p, u; q, v]$ and a point (r, s) define the rectangle $R' = R \star (r, s)$ as

$$R' = [\min\{p, r\}, \min\{u, s\}; \max\{q, s\}, \max\{v, s\}].$$

Clearly, if $p \leqslant r \leqslant q$ and $u \leqslant s \leqslant v$, then $R' = R$.

Algorithm 5.1: Learning Algorithm for Rectangles

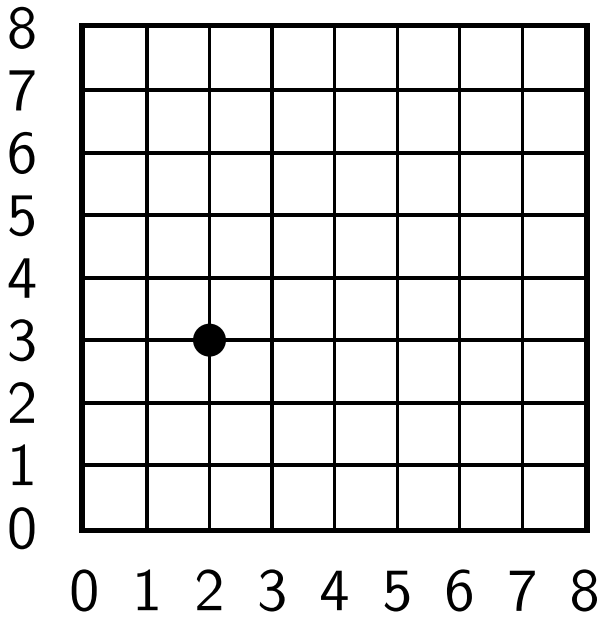
Data: a list of examples of the form $((x_i, y_i), b_i)$ for $1 \leqslant i \leqslant m$

Result: A rectangle $[p, u; q, v]$

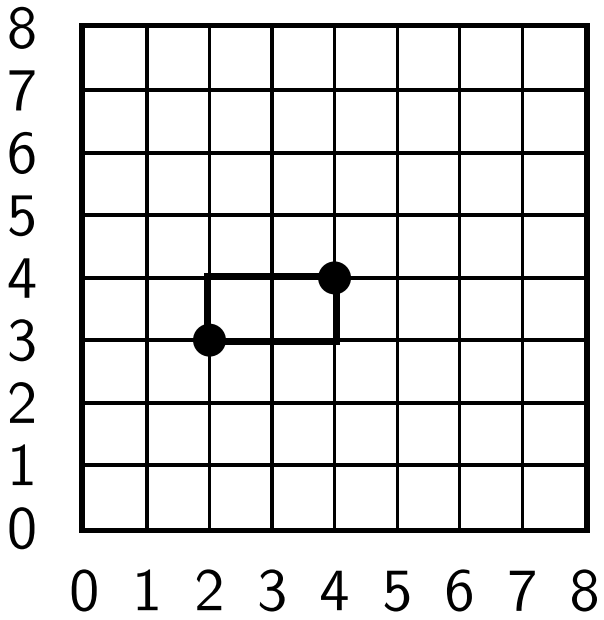
$R = [p_1, u_1; q_1, v_1]$;
$i = 1$;
for $i := 1$ *to* m **do**
    if $b_i = 1$ **then**
        $R = R \star [p_i, u_i; q_i, v_i]$
    end
end
return R;

As before, only positive examples modify the current concept.

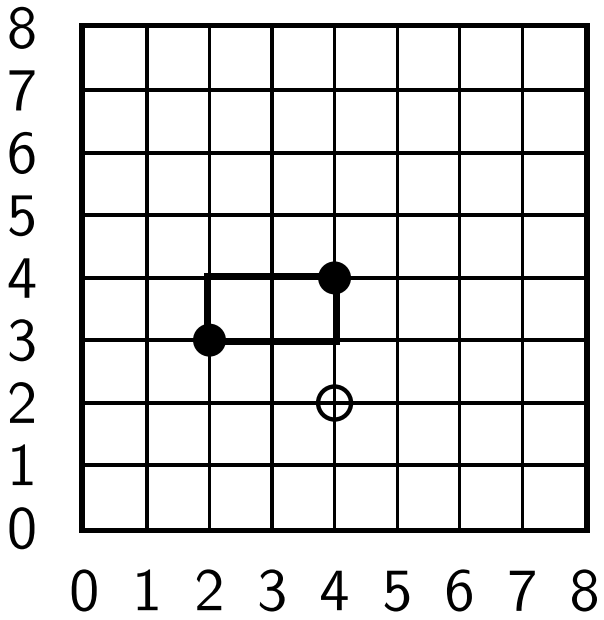
$\mathbf{s} = ((2,3),1),((4,4),1),((4,2),0),((2,6),1),((7,3),1),((6,5),1))$.



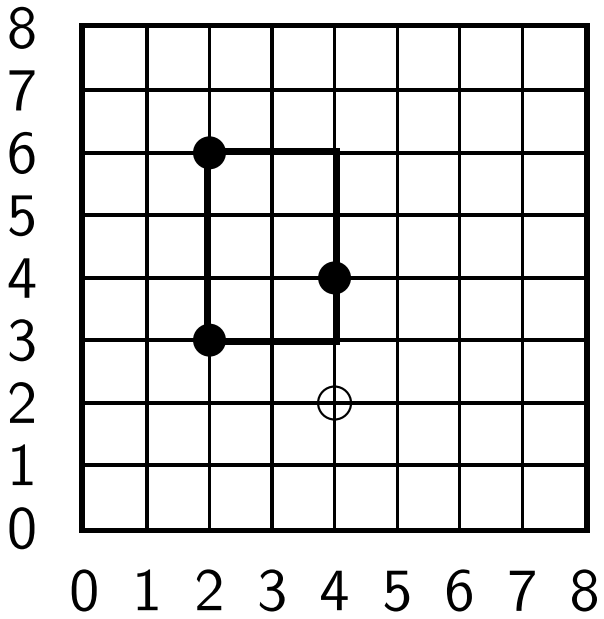
$$\mathbf{s} = ((2,3),1),((4,4),1),((4,2),0),((2,6),1),((7,3),1),((6,5),1))$$



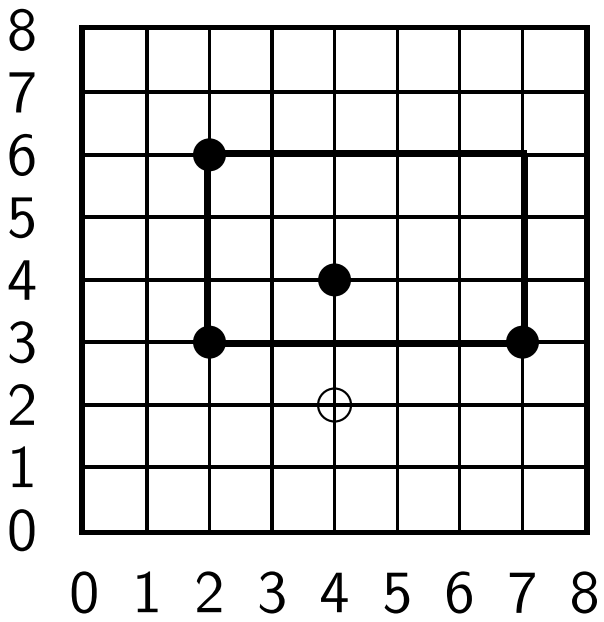
$$\mathbf{s} = ((2,3),1),((4,4),1),((4,2),0),((2,6),1),((7,3),1),((6,5),1))$$



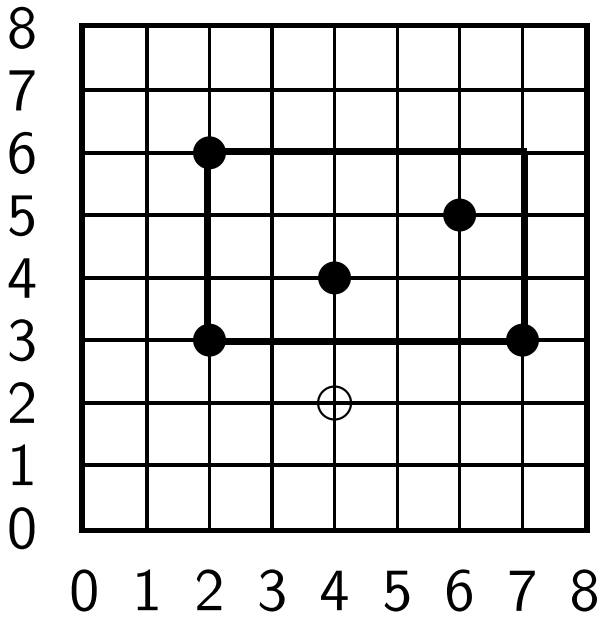
$\mathbf{s} = ((2,3),1),((4,4),1),((4,2),0),((2,6),1),((7,3),1),((6,5),1))$



$$\mathbf{s} = ((2,3),1),((4,4),1),((4,2),0),((2,6),1),((7,3),1),((6,5),1))$$



$\mathbf{s} = ((2,3),1),((4,4),1),((4,2),0),((2,6),1),((7,3),1),((6,5),1))$



- The hypotesis produced by the algorithm classifies correctly all examples of the sequence **s**; this property of the algorithm will be referred to as *consistency*.
- The algorithm is *memoryless*: no example is needs to be seen again to the algorithm.

To acquire a dog I define what I see as a likable dog. I decide that the features I care about are:

hair	yes
large size	no
bad temper	no
has a black nose	yes
color is tan	irrelevant
big ears	yes
good temperament	yes

A friend who helps me get a dog, is aware of my criteria for choosing a dog, but ignores my choices. Nevertheless, he tries to understand what I like in a dog. So, we proceed to sample various dogs by visiting breeders and he makes a note of my preferences. These notes contain a list of examples of the form $(\mathbf{x}_i, b_i) = ((\mathbf{x}_i)_1, \ldots, (\mathbf{x}_i)_7, b_i)$, where

$$(\mathbf{x}_i)_j = \begin{cases} 1 & \text{if feature number } j \text{ is present} \\ 0 & \text{otherwise,} \end{cases} \quad \text{and } b_i = \begin{cases} 1 & \text{if I like the dog,} \\ 0 & \text{otherwise} \end{cases},$$

for $1 \leqslant i \leqslant m$.

The learning begins with a sequence of examples of the form $((b_1 \cdots b_7), c)$, where $b_i \in \{0, 1\}$ is the value of the variable x_i and c is the class of the example:

$$((1,0,0,1,1,1,1),1),((1,1,0,1,1,1,1),0),((1,0,0,0,1,0,1),0),$$
$$((1,0,0,1,0,1,1),1),$$

and a set of variables and their negations, $U = \{x_1, \overline{x_1}, x_2, \overline{x_2}, \ldots, x_7, \overline{x_7}\}$.

Algorithm 6.1: Learning Algorithm for a Conjunction

Data: a list of examples of the form $(\mathbf{x}_i, b_i)$

Result: A conjunction of Variables in a set U

```
1  /* U is the initial set of variables */
2  set U = {u_1, ū_1, ..., u_n, ū_n};
3  for i := 1 to m do
4      if b_i = 1 then
5          for j := 1 to n do
6              if (x_i)_j = 1 then
7                  delete ū_j if present in U
8              else
9                  delete u_j if present in U
10             end
11         end
12     end
13 end
14 return U;
```

Only positive example trigger changes in the set U.

$\{u_1, \overline{u_1}, u_2, \overline{u_2}, u_3, \overline{u_3}, u_4, \overline{u_4}, u_5, \overline{u_5}, u_6, \overline{u_6}, u_7, \overline{u_7}\}$	((1,0,0,1,1,1,1),1)
$\{u_1, \overline{u_2}, \overline{u_3}, u_4, u_5, u_6, u_7\}$	((1,1,0,1,1,1,1),0)
$\{u_1, \overline{u_2}, \overline{u_3}, u_4, u_5, u_6, u_7\}$	((1,0,0,0,1,0,1),0)
$\{u_1, \overline{u_2}, \overline{u_3}, u_4, u_5, u_6, u_7\}$	((1,0,0,1,0,1,1),1)
$\{u_1, \overline{u_2}, \overline{u_3}, u_4, u_6, u_7\}$	

The conjuction returned by the algorithm is

$$x_1 \wedge \overline{x_2} \wedge \overline{x_3} \wedge x_4 \wedge x_6 \wedge x_7$$

like_dog: $\{0,1\}^7 \longrightarrow \{0,1\}$, where

$$\text{like\_dog}(x_1,\ldots,x_7) = \begin{cases} 1 & \text{if } x_1 = 1, x_2 = 0, x_3 = 0, \\ & x_4 = 1, x_6 = 1, x_7 = 1 \\ 0 & \text{otherwise.} \end{cases}$$

A set of literals $U = \{\ell_1,\ldots,\ell_n\}$ can be represented as a conjunction $\ell_1 \wedge \ell_2 \wedge \cdots \wedge \ell_n$. Starting from the set U we can represent U as a conjunction.
For example, the initial set U is represented by the conjunction

$$u_1 \wedge \overline{u_1} \wedge \cdots \wedge u_n \wedge \overline{u_n}$$

and the corresponding Boolean function is constant function 0.
The conjunction that corresponds to the set returned by the algorithm is

$$u_1 \wedge \overline{u_2} \wedge \overline{u_3} \wedge u_4 \wedge u_6 \wedge u_7.$$

It is easy to see that the Algorithm is consistent.

- The number of concepts that can be defined on n variables is 2^{2^n}.
- The number of concepts that can be represented by conjunctions is 3^n, a small fraction of 2^{2^n}, and this is both our concept space and our hypothesis space.

The hypothesis space is the set of Boolean functions that correspond to formulas in $D_{m,p}$. As before, we start with a sample

$$\mathbf{s} = ((\mathbf{x}_1, b_1), \ldots, (\mathbf{x}_n, b_n)).$$

- The algorithm is initalized with the set of all monomials in $D_{m,p}$.
- Only negative example trigger deletions of monomials.

Algorithm 7.1: Learning Algorithm for a Disjunction of Conjunctions

Data: a list of examples **s**

Result: A disjunction of monomials

set $H = D_{m,p}$;
for $i := 1$ *to* n **do**
    if $b_i = 0$ *and* $\langle H \rangle(\mathbf{x}_i) = 1$ **then**
        delete monomials μ in H such that $\langle \mu \rangle(\mathbf{x}_i) = 1$
    end
end
return H;

$m = 3$, $p = 2$ and

$$\begin{aligned} H &= \{x_1, x_2, x_3, \overline{x_1}, \overline{x_2}, \overline{x_3}, x_1x_2, x_1x_3, x_2x_3, \\ &\quad \overline{x_1}x_2, \overline{x_1}x_3, \overline{x_2}x_3, x_1\overline{x_2}, x_1\overline{x_3}, x_2\overline{x_3}, \\ &\quad \overline{x_1x_2}, \overline{x_1x_3}, \overline{x_2x_3}\} \end{aligned}$$

If the set of negative examples consists of $(0,0,0), (0,0,1), (0,1,0), (0,1,1)$ then the set H is modified as follows:

Negative example	*Removed monomials*
$(0,0,0)$	$\overline{x_1}, \overline{x_2}, \overline{x_3}, \overline{x_1x_2}, \overline{x_1x_3}, \overline{x_2x_3}$
$(0,0,1)$	$x_3, \overline{x_1}x_3, \overline{x_2}x_3$
$(0,1,0)$	$x_2, x_2\overline{x_3}$
$(0,1,1)$	x_2x_3

The remaining monomials are

$$x_1, x_2, x_3, x_1x_2, x_1x_3, \overline{x_1}x_2, x_1\overline{x_2}, x_1\overline{x_3},$$

which results in the disjunctive formula

$$\phi = x_1 \vee x_2 \vee x_3 \vee x_1x_2 \vee x_1x_3 \vee \overline{x_1}x_2 \vee x_1\overline{x_2} \vee x_1\overline{x_3}.$$

This formula can be simplified resulting in an equivalent formula

$$\psi = x_1 \vee x_2 \vee \overline{x_3},$$

which is the simplest representation of $\langle\phi\rangle$.

Recommended Bibliography

We recommend that you consult the following sources:

- [4]
- [1]
- [2]
- [3]


M. Anthony and N. Biggs.
Computational Learning Theory.
Cambridge University, Cambridge, 1997.


M. J. Kearns and U. V. Vazirani.
An Introduction to Computational Leaning Theory.
MIT Press, Cambridge, MA, 1997.


T. M. Mitchell.
Machine Learning.
McGraw-Hill, Boston, 1997.


L. Valiant.
Deductive learning.
Phylosophical Transactions of the Royal Society of London (A), 312:441–446, 1984.