# Concept learning

## 1. Definizione

Il problema del concept learning può essere definito come il problema di inferire attraverso uno spazio predefinito di potenziali ipotesi l'ipotesi che meglio fitta gli esempi di training.

## 2. Notazione

- X = insieme o **spazio delle istanze**, ovvero l'insieme degli elementi su cui è definito il concetto (o ipotesi) da inferire. Dati dunque $n$ attributi $A_1, A_2, ..., A_n$, ognuno contenente un numero arbitrario ma finito di valori, $X = A_1 \times A_2 \times ... \times A_n$ e **un 'istanza** è un elemento $x \in X$
- c = **concetto** o funzione da inferire (target concept). In generale c può essere una qualsiasi funzione booleana definita sullo spazio delle istanze, ovvero $c : X \to \{0, 1\}$
- H = insieme o **spazio delle ipotesi**, che contiene tutte le possibili ipotesi che l'algoritmo d'apprendimento può prendere in considerazione per inferire c. In generale si ha che $\forall h \in H,\ h : X \to \{0, 1\}$. Ogni ipotesi può anche essere considerata come una congiunzione di attributi: avevndo $n$ attributi, un'ipotesi può essere considerata come un vettore di $n$ elementi $h = \langle a_1, a_2, ..., a_n \rangle = \langle A_1 = a_1 \wedge A_2 = a_2 \wedge ... \wedge A_n = a_n \rangle$ dove $a_i$ rappresenta il valore assunto dall'attributo i-esimo. Inoltre vale che $a_i \in A_i \cup \{?, \emptyset\}$, dove $a_i = ?$ significa che l'attributo $a_i$ può assumere qualsiasi valore in $A_i$, mentre $a_i = \emptyset$ significa che non può assumere nessun valore
- $D = \{\langle x_1, c(x_1)\rangle, \langle x_2, c(x_2\rangle,\ ... \ , \langle x_n, c(x_n)\rangle\}$, insieme degli esempi o **training set**, dove $x_i \in X$ e $c(x_i)$ è il target concept associato a quell'istanza o esempio. Tutte le istanze per cui $c(x_i) = 1$ sono chiamate **esempi postitvi**, mentre tutte le istanze per cui $c(x_i) = 0$ sono chiamate **esempi negativi**

Nel concept learning dunque, dato un training set $D$, bisogna determinare un'ipotesi $h \in H \mid \forall x \in X(h(x) = c(x))$

### 2.1. Esempio

Attributi $A_i \in \{1, 2, 3\}$, con $i \in \{1, 2, 3\}$

- Quante istanze ci sono in $X$?
  $|X| = |A_1 \times A_2 \times A_3| = 3^3 = 27$
- Quante ipotesi sintatticamente distinte esistono in $H$?
  $(3 + 2)^3 = |A_1 \times A_2 \times A_3|$, con $A_i \in \{1, 2, 3, ?, \emptyset\}$

- Quante ipotesi <u>semanticamente</u> distinte esistono in $H$?
  $1 + (3 + 1)^3 = |\{\langle \emptyset, \emptyset, \emptyset \rangle\} \cup (A_1 \times A_2 \times A_3)|$, con $A_i \in \{1, 2, 3, ?\}$

## 3. Performance evaluation

Matrice di confusione:

| | **Positive** | **Negative** |
|---|---|---|
| **Positive** | $TP$ | $FP$ |
| **Negative** | $FN$ | $TN$ |

Dove:

- $TP$ = Numero di True Positive
- $FP$ = Numero di False Positive
- $TN$ = Numero di True Negative
- $FN$ = Numero di False Negative

Indici di performance:

- Sensitivity = $\frac{TP}{TP+FN}$, ovvero $p(y = 1|x = 1)$
- Specificity = $\frac{TN}{FP+TN}$, ovvero $p(y = 0|x = 0)$
- Positive predictive value (PPV) = $\frac{TP}{TP+FP}$, ovvero $p(x = 1|y = 1)$
- Negative predictive value (NPV) = $\frac{TN}{TN+FN}$, ovvero $p(x = 0|y = 0)$
- Accuracy = $\frac{TP+TN}{TP+FP+TN+FN}$, ovvero $p(y = x)$

Dove $x$ rappresenta la vera classe d'appartenenza, mentre $y$ rappresenta la classe predetta

# Concept learning come ricerca

Il concept learning può essere interpretato come la ricerca dell'ipotesi che meglio fitta gli esempi di training.
Molti algoritmi per il concept learning organizzano tale ricerca basandosi su un **ordine parziale** per lo sapzio delle ipotesi $H$:

> $\forall x \in X,\ \forall h \in H$, si dice che $x$ **soddisfa** $h$ sse $h(x) = 1$

> Date $h_j, h_k \in H$, $h_j$ è **più generale o uguale di** $h_k$ ($h_j \geq_g h_k$) sse $\forall x \in X, (h_k(x) = 1) \rightarrow (h_j(x) = 1)$.
> Mentre $h_j$ **è più generale di** $h_k$ ( $h_j > h_k$) sse $(h_j \geq_g h_k) \wedge (h_k \not\geq_g h_j)$

> Insiemisticamente parlando un'ipotesi $h_j$ **è più generale o uguale di** un'altra ipotesi $h_k$ sse $h_k \subseteq h_j$.
> Mentre $h_j$ **è più generale di** un'altra ipotesi $h_k$ sse $h_k \subset h_j$

> Logicamente parlando un'ipotesi $h_j$ **è più generale di** un'altra ipotesi $h_k$ sse $h_j$ impone meno vincoli rispetto a $h_k$

Ad esempio suppponiamo di avere uno spazio delle ipotesi $H$ definito sugli attributi $A_i \in \{1, 2\}$, con $i \in \{1, 2\}$.
Allora l'ipotesi $h_1 = \langle 1, ? \rangle \geq_g h_2 = \langle 1, 1 \rangle$, infatti $h_2 = \{\langle 1, 1 \rangle\} \subseteq h_1 = \{\langle 1, 1 \rangle, \langle 1, 2 \rangle\}$, inoltre $h_1$ vincola soltanto $A_1 = 1$, mentre $h_2$ vincola $A_1 = 1 \wedge A_2 = 1$.

Si definisce inoltre la **consistenza** di un'ipotesi come:

> Un'ipotesi $h \in H$ è **consistente** con un training set $D$ sse $\forall \langle x, c(x) \rangle \in D,\ h(x) = c(x)$

# 1. Find-S

L'algoritmo Find-S parte dall'ipotesi più specifica (quella con tutti $\emptyset$) e cerca di generalizzarla man mano. Ad ogni passo l'algoritmo trova l'ipotesi più specifica consistente con i dati di training.
L'algoritmo è il seguente:

> 1. Sia $h \in H$ l'ipotesi più specifica, ovvero quella per cui ogni $a_i = \emptyset$
> 2. $\forall \langle x, c(x) \rangle \in D \mid c(x) = 1$
>    * $\forall a_i \in h$
>    Se $a_i$ **non è soddisfatta** da $x$, allora sostituisci ad $a_i$ il vincolo più generale che è soddisfatto da $x$, dove $\emptyset \leq_g K \leq_g ?$, dove $K \in A_i$

## 1.1. Esempio

Supponiamo $D$ sia così composto:

| | $A_1$ | $A_2$ | $A_3$ | $A_4$ | $A_5$ | $c(x)$ |
|---|---|---|---|---|---|---|
| $x_1$ | 0 | 1 | 1 | 1 | 1 | 1 |
| $x_2$ | 1 | 0 | 1 | 1 | 1 | 0 |
| $x_3$ | 0 | 1 | 1 | 1 | 0 | 1 |
| $x_4$ | 0 | 1 | 0 | 1 | 0 | 1 |
| $x_5$ | 1 | 0 | 1 | 0 | 1 | 0 |

allora l'algoritmo Find-S si comporterà così:

1. $h = \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$
2. $c(x_1) = 1$, dunque $h = \langle 0, 1, 1, 1, 1 \rangle$, in quanto $a_i \neq x_{1,i}$ e valendo la relazione $x_{1,j} \geq_g a_i$
3. $c(x_2) = 0$, esempio scartato da Find-S
4. $c(x_3) = 1$, dunque $h = \langle 0, 1, 1, 1, ? \rangle$, in quanto $(a_5 = 1) \neq (x_{3,5} = 0)$ ed essendo $? \geq_g (a_5 = 1)$
5. $c(x_4) = 1$, dunque $h = \langle 0, 1, ?, 1, ? \rangle$, in quanto $(a_3 = 1) \neq (x_{4,3})$, ed essendo $? \geq_g (a_3 = 1)$
6. $c(x_5) = 0$, esempio scartato da Find-S

L'ipotesi prodotta da Find-S è dunque $h = \langle 0, 1, ?, 1, ? \rangle$ che estesa vale $h = \{\langle 0, 1, 0, 1, 0 \rangle, \langle 0, 1, 1, 1, 0 \rangle, \langle 0, 1, 0, 1, 1 \rangle, \langle 0, 1, 1, 1, 1 \rangle\}$

## 1.2. Pro

Supponendo che:

- Il training set $D$ sia esente da errori
- Lo spazio delle ipotesi sia composto dalla congiunzione di attributi
- $c \in H$

allora è garantito che l'ipotesi trovata da Find-S sia la più specifica in H consistente con D (positivi e negativi)

Ad esempio (from Zoppis), supponiamo:

- D sia così composto:

| | $A_1$ | $A_2$ | $A_3$ | $c(x)$ |
|---|---|---|---|---|
| $x_1$ | 1 | 1 | 2 | 1 |
| $x_2$ | 1 | 3 | 2 | 0 |
| $x_3$ | 1 | 2 | 2 | 1 |

- $c = \langle A_1 = 1 \wedge (A_2 = 1 \vee A_2 = 2) \wedge A_3 = 2 \rangle = \{\langle 1, 1, 2 \rangle, \langle 1, 2, 2, \rangle\}$

Se applichiamo Find-S troveremo che l'ipotesi più specifica "consistente" con D è $h = \langle 1, ?, 2 \rangle \neq c$

## 1.3. Contro

- Impossibile determinare se l'ipotesi trovata è l'unica ipotesi consistente con D o se ne esistono altre
- Ignorando gli esempi negativi, Find-S non può capire se D contiene dati incosinstenti (contenti errori)

- Preferisce solo l'ipotesi più specifica consistente con D, perchè non la più generale? O un qualcosa in between?

# 2. Candidate elimination

L'algoritmo Candidate elimination cerca di ovviare ad alcuni dei problemi di Find-S andando a ricercare le ipotesi consistenti con il training set D, senza però enumerarle tutte, ovvero ritorna un sottoinsieme $VS_{H,D} \subseteq H$ denominato **version space** così definito:

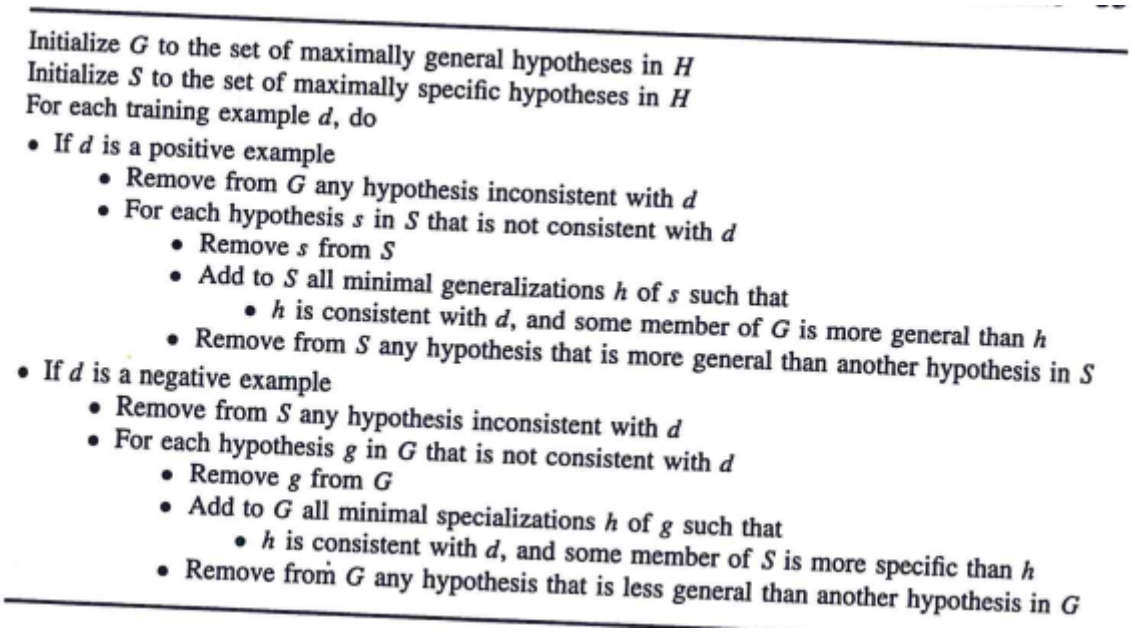
> $VS_{H,D} :=\{h \in H \mid h \; consistente \; con \; D\} \equiv \{h \in H \mid (\exists s \in S)(\exists g \in G)(g \geq_g h \geq_g s)\}$

dove

> $S = \{s \in H \mid s \; consistente \; con \; D \wedge (\neg \exists s' \in H)[(s >_g s') \wedge s' \; consistente \; con \; D]\}$ è l'insieme delle ipotesi **più specifiche**

> $G = \{g \in H \mid g \; consistente \; con \; D \wedge (\neg \exists g' \in H)[(g' >_g g) \wedge g' \; consistente \; con \; D]\}$ è l'insieme delle ipotesi **più generali**

In altre parole $S$ e $G$ fungono da lower e upper bound rispettivamente per le possibili ipotesi candidate.
L'algoritmo è il seguente:


Initialize $G$ to the set of maximally general hypotheses in $H$
Initialize $S$ to the set of maximally specific hypotheses in $H$
For each training example $d$, do
- If $d$ is a positive example
  - Remove from $G$ any hypothesis inconsistent with $d$
  - For each hypothesis $s$ in $S$ that is not consistent with $d$
    - Remove $s$ from $S$
    - Add to $S$ all minimal generalizations $h$ of $s$ such that
      - $h$ is consistent with $d$, and some member of $G$ is more general than $h$
    - Remove from $S$ any hypothesis that is more general than another hypothesis in $S$
- If $d$ is a negative example
  - Remove from $S$ any hypothesis inconsistent with $d$
  - For each hypothesis $g$ in $G$ that is not consistent with $d$
    - Remove $g$ from $G$
    - Add to $G$ all minimal specializations $h$ of $g$ such that
      - $h$ is consistent with $d$, and some member of $S$ is more specific than $h$
    - Remove from $G$ any hypothesis that is less general than another hypothesis in $G$

## 2.1. Esempio

Supponiamo $D$ sia così composto:

| | $A_1$ | $A_2$ | $A_3$ | $A_4$ | $A_5$ | $c(x)$ |
|---|---|---|---|---|---|---|

| | $A_1$ | $A_2$ | $A_3$ | $A_4$ | $A_5$ | $c(x)$ |
|---|---|---|---|---|---|---|
| $x_1$ | 0 | 1 | 1 | 1 | 1 | 1 |
| $x_2$ | 1 | 0 | 1 | 1 | 1 | 0 |
| $x_3$ | 0 | 1 | 1 | 1 | 0 | 1 |
| $x_4$ | 0 | 1 | 0 | 1 | 0 | 1 |
| $x_5$ | 1 | 0 | 1 | 0 | 1 | 0 |

allora l'algortimo Candidate elimination si comporterà così:

1. $S = \{\langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle\}$
2. $G = \{\langle ?, ?, ?, ?, ? \rangle\}$
3. $c(x_1) = 1$, ergo si generalizza $S$ mantenendo la consistenza con $G$. $G$ è consistente con $x_1$, dunque si avrà:
   - $S_1 = \{\langle 0, 1, 1, 1, 1 \rangle\}$
   - $G_1 = G$
4. $c(x_2) = 0$, ergo si specializza $G$ mantenendo la consistenza con $S_1$. $S_1$ è cosistente con $x_2$, infatti $s_1(x_2) = c(x_2) = 0$, specializzo dunque $G$ con tutte quelle ipotesi meno generali che siano consistenti con $x_2$, ottenendo dunque:
   - $S_2 = S_1$
   - $G_2 = \{\langle 0, ?, ?, ?, ? \rangle \langle ?, 1, ?, ?, ? \rangle \langle ?, ?, 0, ?, ? \rangle \langle ?, ?, ?, 0, ? \rangle \langle ?, ?, ?, ?, 0 \rangle\}$, da cui vengono eliminate le ipotesi $\langle ?, ?, 0, ?, ? \rangle \langle ?, ?, ?, 0, ? \rangle \langle ?, ?, ?, ?, 0 \rangle$ poichè non risultano essere più generali dell'unica ipotesi presente in $S_1$, ergo:
     $G_2 = \{\langle 0, ?, ?, ?, ? \rangle \langle ?, 1, ?, ?, ? \rangle\}$
5. $c(x_3) = 1$, ergo si generalizza $S_2$ mantenendo la consistenza con $G_2$. Tutte le ipotesi in $G_2$ sono consistenti con $x_3$, dunque si ottiene:
   - $S_3 = \{\langle 0, 1, 1, 1, ? \rangle\}$
   - $G_3 = G_2$
6. $c(x_4) = 1$, solito discorso. Si ottiene:
   - $S_4 = \{\langle 0, 1, ?, 1, ? \rangle\}$
   - $G_4 = G_3$
7. $c(x_5) = 0$, si specializza $G_4$ mantenendo la consistenza con $S_4$. Si ottiene:
   - $S_5 = S_4 = \{\langle 0, 1, ?, 1, ? \rangle\}$
   - $G_5 = G_4 = \{\langle 0, ?, ?, ?, ? \rangle, \langle ?, 1, ?, ?, ? \rangle\}$

Il version space è dunque composto da:
$VS_{H,D} = S \cup G \cup \{\langle 0, 1, ?, ?, ? \rangle, \langle 0, ?, ?, 1, ? \rangle, \langle ?, 1, ?, 1, ? \rangle\}$

## 2.2. Pro

Supponendo che:

- D non contenga errori
- $c \in H$

allora candidate elimination converge all'ipotesi corretta

## 2.3. Contro

- Se D contiene errori l'ipotesi corretta viene eliminata dal version space
- Possono essere appresi concetti composti solo dalla congiunzione di attributi
- Se $c \notin H$ allora l'output dell'algoritmo sarà l'ipotesi vuota