Definizione

Concept Learning

Inferire una funzione booleana da esempy di training positivi e negativi

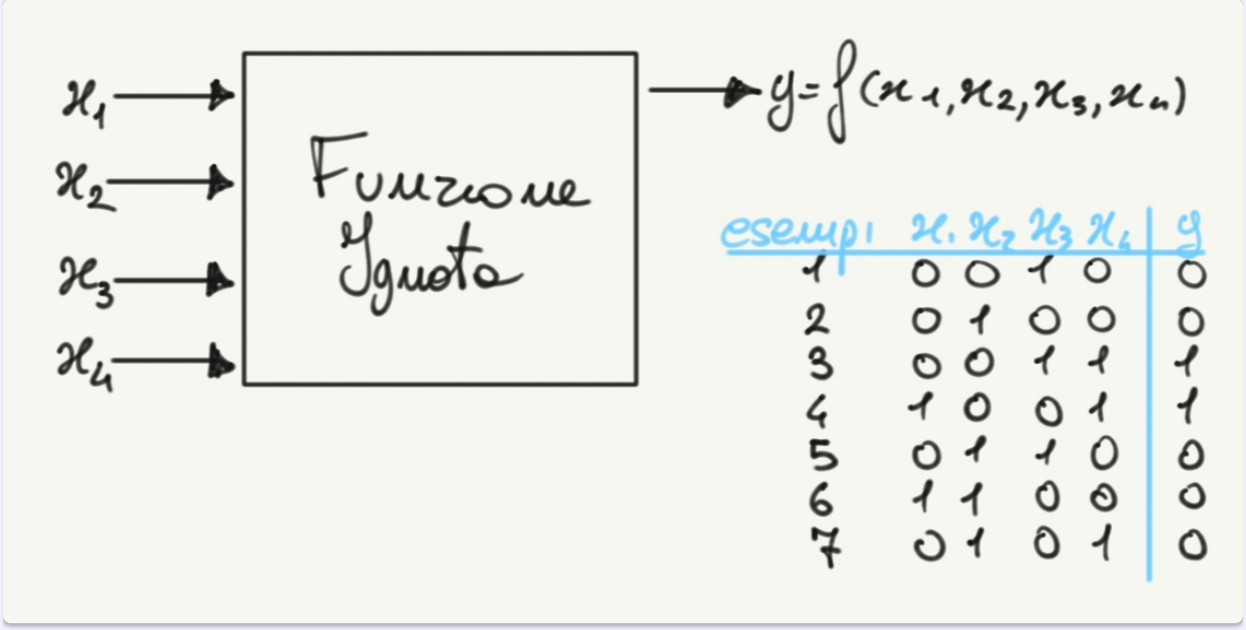
$$c : X \to \{t, f\} \vee \{yes, no\} \vee \{+, -\} \vee \{0, 1\}$$

dove X è lo spazio delle ipotesi.

- Un *esempio* di training ha la forma $< x, c(x) > \in D$
- $h : X \to \{0, 1\}$ soddisfa x se $h(x) = 1$
- Un'ipotesi h è *consistente* con
  - un esempio se $h(x) = c(x) \in D$
  - D se $h(x) = c(x)$ per ogni esempio di training $\in D$

Apprendimento delle funzioni booleane

Example: Learning Boolean functions



É un problema mal posto: potremmo violare la soluzione o la sua esistenza, unicità o stabilità.
Abbiamo $2^{2^4} = 2^{16} = 65536$ possibili funzioni su 4 valori in input. Non possiamo sapere quale sia quella corretta finché non le vediamo tutte

In generale ci sono $|H| = 2^{\#istanze} = 2^{2^n}$, dove $n = \text{dimensione dell'input}$. Cerchiamo di lavore con uno *spazio delle ipotesi ristretto*: cominciamo scegliendo un H considerabilmente minore dello spazio di tutte le funzioni possibili.

Regole delle congiunzioni

Literali possibili (in AND): $|H| = 2^n$, se consideriamo anche l'operatore NOT: $3^n + 1$

Rappresentare le ipotesi

Nella rappresentazione delle ipotesi consideriamo h una congiunzione di costrutti sugli attributi, ognino di questi può essere un valore

Specifico	Trascurabile	Non Accettato
Water = warm	Water = ?	Water = $\emptyset$

Example

$$< \begin{matrix} sky \\ sunny \end{matrix} \quad \begin{matrix} temp \\ ? \end{matrix} \quad \begin{matrix} Humid \\ ? \end{matrix} \quad \begin{matrix} Wind \\ strong \end{matrix} \quad \begin{matrix} Water \\ ? \end{matrix} \quad \begin{matrix} Farecast \\ same \end{matrix} > \Rightarrow$$
$$\Rightarrow sky = \text{sunny} \ \wedge \ \text{wind} = \text{strong} \ \wedge \ \text{forecast} = \text{same}$$

INPUT	OUTPUT
Istanza X : Possibili giorni descritti dagli attributi $sky, temp, humidity, \dots$	Un'ipotesi $h \in H$ t.c. $h(x) = x(x) \forall x \in X$
Target function: $c : EnjoySport\ X \to \{0, 1\}$	
Ipotesi H: congiunzione di un insieme finito di letterali $< Sunny, ?, ?, Strong, ?, Same >$	
Esempi di training: D: esempi positivi e negativi della funzione target $< x_1, c(x_1) >, \dots, < x_n, c(x_n) >$	

Sky: 3 valori, $Temp, Humidity, wind, water, forecast$: 2 valori

- $\underline{\#\text{istanze distinte}} = 3 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 96$
- $\underline{\#\text{concetti distinti}} = 2^{\#\text{istanze}} = 2^{96}$
- $\underline{\#\text{ipotesi sintatticamente distinte(congiunzioni)}} = 5 \cdot 4 \cdot 4 \cdot 4 \cdot 4 \cdot 4 = 5120$
- $\underline{\#\text{ipotesi semanticamente distinte}} = 1 + 4 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 = 973$

In generale, *strutturare lo spazio di ricerca* può sempre essere utile queando abbiamo grandi spazi di ricerca (anche infiniti).

Ordinamento da generale a specifico

Se consideriamo ad esempio 2 ipotesi:

$$h_1 =< Sunny, ?, ?, Strong, ?, ? > \to h_2 =< Sunny, ?, ?, ?, ?, ? >$$

e un'insieme di istanze coperte dalle ipotesi

h_2 impone meno vincoli di h_1, quindi classifica più istanze come positive

Generalize

Dati h_1 e h_2 funzioni a calori booleani definite su X, h_j è più generale o uguale a $h_k (h_j \geq h_k)$ sse

$$\forall x \in X : (h_k(x) = 1) \implies (h_j(x) = 1)$$

La relazione $\geq$ impone un ordine parziale (*p.o.*) sullo spazio delle ipotesi H. Possiamo trovare vantaggio *organizzando la ricerca in* H.

Algoritmo Find-S

Definizione

1. Inizializza h all'ipotesi più specifica di H
2. Per ogni istanza di training x

```
FOR EACH x in h
    IF the a_i in h is satisfied by x
```

```
            THEN do nothing
            ELSE replace a_i in the next more general constraing that is staisfied by x
```

3. Output dell'ipotesi h

Proprietà	Limitazioni
Lo *spazio delle ipotesi* è descritto da congiunzioni di attributi	Non dice se chi apprende *converge al concetto target*, nel senso che non può determinare se ha trovato l'*l'unica ipotesi consistente* con gli esempi di training
`Find-S` restituirà l'*ipotesi più specifica* che mantenga H consistente con gli esempi di training positivi	Non dice quando i *dati di training* sono *inconsistenti* e ignora gli esempi negativi
L'ipotesi h restituita sarà *consistente* con gli esempi negativi (dato che $c \geq h$)	no noise tolleration

Version Space

> Version space

Il **Version Space**, $VS_{H,D}$, rispettando lo spazio delle ipotesi H e il training set D, è il *sottoinsieme delle ipotesi H consistenti con tutti gli esempi di training*

$$VS_{H,D} = \{h \in H | consistent(h, D)\}$$

L'idea è quella che si restituisca una *descrizione* dell'insieme di tutti gli h consistenti con D, così possiamo **enumerarli**.

Algoritmo List-Then Eliminate

> Quote

1. **Version space** $\leftarrow$ una lista contenente tutte le ipotesi in H
2. `for each` esempio di training $< x, c(x) >$ rimuovi dallo spazio del version space tutte le ipotesi inconsistenti con l'esempio di training

$$h(x) \neq c(x)$$

3. Restituisci la lista delle ipotesi nel version space

Rappresentare il Version Space

> Quote

- Il **General Boundary** (*confine generale*) (G) del version space è l'insieme dei massimi membri generali di H consistente con D
- Il **Specific Boundary** (*confine specifico*) (S) del Version Space è l'insieme dei massimi membri specifici di H consistente con D
  TEOREMA: Tutti i membri del Version Space sono compresi fra questi due boundaries

$$VS_{H,D} = \{h \in H | (\exists s \in S)(\exists g \in G)(g \geq h \geq s)\}$$

Possiamo fare una *ricerca completa* di ipotesi consistenti usando in due boundaries S e G per il VS, quindi non ristretto a S come per find-S.

Candidate Elimination Algorithm

Esempio Positivo	Esempio Negativo
Si rimuovuono da G tutte le ipotesi inconsistenti con d	Rimuovi da S tutte le ipotesi inconsistenti con d
`For each` ipotesi $s \in S$ non consistente con d: Generalizza $s \longrightarrow$ Aggiungi s alla minima generalizzazione h t.c. $(h \text{ consistente con } d) \vee (\text{ dei membri di } S \text{sono più generali di} h$	`For each` ipotesi $g \in G$ non consistente con d: Specializza G Rimuovi g da $G \longrightarrow$ Aggiungi a G tutte le minime specializzazioni h t.c. $(h \text{ consistente con } d) \vee (\text{Alcuni membri di } S \text{ sono più generali d}$
Rimuovi da G tutte le ipotesi più generali	Rimuovi da G tutte le ipotesi meno generali

Il nostro spazio delle ipotesi non può rappresentare un concetto target disgiunto semplice (ad es. $Sky = Sunny \vee Sky = Cloudy$)

Bias

Assumiamo che lo spazio delle ipotesi H contenga il concetto target $c \implies c$ può essere descritto da una *congiunzione* (con un AND) *di letterali*.

Unbiased Learner

Si sceglie un H che esprima tutti i concetti apprendibili, quindi H è l'insieme dei *possibili sottoinsiemi di* X, il power set $P(X)$.

> **Example**
>
> $$|X| = 96, \quad |P(X)| = 2^{96} \approx 10^{28} \quad \text{concetti distinti}$$

Assumiamo gli esempi positivi (x_1, x_2, x_3) e quelli negativi (x_4, x_5)

$$S : \{(x_1 \vee x_2 \vee x_3\} \qquad G : \{\neg(x_4 \vee x_5)\}$$

Gli unici esempi *non ambigui* sono gli esempi di training stessi (rappresentati da H), quindi per apprendere il target concept.

> **Theorem**
>
> Un unbiased learner non può generalizzare
> DIM:
> Ogni istanza non osservata viene classificata come positiva precisamente dalla metà delle ipotesi nel VS e negativa dall'altra metà (*rejection*)
>
> $$\forall\ h \text{ consistente con } x_i\ (test), \exists\ h \text{ identico a } h \text{ eccetto } h'(x_i) <> h(x_i)$$
>
> $$h \in VS \implies h' \in VS$$
> $$(\text{sono identici in } D)$$

Futilità di un Free-Biased Learner

Un learner che non fa assunzioni sull'identità del concetto target, *non* ha basi razionali per *classificare le istanze non viste*. Bias non fa assunzioni per efficienza, ma ne ha *bisogno per generalizzarla*, ma non ci dice qual'è la migliore soluzione per generalizzare. Il *problema* è caratterizzare il bias per approcci di apprendimento diversi.

> **Example**
>
> (TR =Training Set, TS =Test Set)

	X	$c(x)$	$H = \{x, \neg x, 0, 1\}$
TR	0	0	$VS = \{0, 1\}$
TS	1	?	$\Longrightarrow$ Può essere 1 o 0 finché non si usa X come TR

Inductive Bias

Inductive Bias

I **bias induttivi** di L sono tutti i minimi insiemi di asserzioni B tali che per ogni concetto target c e corrispondente dato di training D_c

$$(\forall x_i \in X)[B \vee D_c \vee x_i] \vdash L(x_i, D_c)$$

> $A \vdash B$ significa che A comporta logicamente B

Consideriamo:

- L'algoritmo di concept learning L
- L'istanza X e il concetto target c
- Gli esempi di training $D_c = \{< x, c(x) >\}$
  $L(x_i, D_c)$ denota la classificazione assegnata all'istanza x_i da L dopo il training su D_c

3 Learner con biased diversi

Rote Learner (look up table)	Version Space Candidate Elimination Algorithm	Find-S
Memorizza gli esempi, classifica x sse è associato a un esempio osservato precedentemente $osservato \; noinductive \; bias \Longrightarrow no \; generalization$	Bias: lo spazio delle ipotesi contiene il target concept (congiunzioni di attributi)	Bias: lo spazio delle ipotesi contiene il concetto target e tutte le istanze negative $\Longrightarrow$ Abbiamo un linguaggio bias fatto di AND sui letterali più il *search bias* sulle preferenze dell'ipotesi più specifica

Per superare le restrizioni congiuntive sui *modelli flessibili* usiamo *diversi approcci di ML:*

- Decision Tree
- *Algoritmi Genetici* (TO LINK)
  Si codifica ogni insieme di regole come una stringa di bit e si usa la ricerca genetica per esplorare questo H
- *Programmazione logica induttiva* (TO LINK)
  Le regole della logica del prim'ordine contengono le variabili e i programmi sono inferiti automaticamente dagli esempi