Spazio delle versioni

Tecniche di apprendimento

- Apprendimento attributo valore:
  - ipotesi congiuntive
  - alberi di decisione
  - regole di produzione
  - metodi basati sulle istanze
  - reti bayesiane
- Apprendimento del primo ordine:
  - Programmazione Logica Induttiva
  - Apprendimento Statistico Relazionale

Apprendimento di ipotesi congiuntive

- [Mit97] Esempio: Apprendere il concetto target “giornate in cui ad Aldo piace fare sport”
- Esempi: giornate delle quali si conosce se ad Aldo e’ piaciuto o meno fare sport

Sky	Air Temp	Humid	Wind	Water	Forecast	Enjoy Sport
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

Rappresentazione delle ipotesi

- Una ipotesi h e' una congiunzione di vincoli sugli attributi dell'istanza (ipotesi congiuntive).
- Ogni vincolo puo' essere:
  - un valore specifico (ad es. "Water=Warm")
  - un valore qualunque (ad es. "Water=?")
  - nessun valore (ad es. "Water=∅")

Sky	Air Temp	Humid	Wind	Water	Forecast
?	Cold	High	?	?	?

- Puo' essere scritta come

h=<?,cold,high,?,?,?> oppure

if Air Temp=Cold and Humid=High then EnjoySport=Yes

Una formalizzazione

- Dati:
  - un insieme di possibili istanze X:
    - insieme dei possibili giorni, ciascun descritto dagli attributi Sky, Air Temp, Humid, Wind, Water, Forecast
  - un concetto target c: $c: X \rightarrow \{0,1\}$
    - c(x)=1 if EnjoySport=yes
    - c(x)=0 if EnjoySport=no
  - un insieme di esempi D: $<x_i, c(x_i)>$ dove ciascun $x_i \in X$
  - Uno spazio delle ipotesi H:
    - espresse come una congiunzioni di vincoli sugli attributi

Una formalizzazione

- Determinare:
  - una ipotesi h in H tale che
    - h(x)=c(x) per tutti gli x in X
  - il problema di apprendimento consiste nel determinare una ipotesi h che dà lo stesso risultato del concetto target c sull'intero insieme di istanze X.

Quali informazioni sono disponibili?

D

Sky	Air Temp	Humid	Wind	Water	Forecast	Enjoy Sport
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

- Se Sky ha 3 possibili valori e Air Temp, Humid, Wind, Water and Forecast ognuno ne hanno 2, allora X contiene in totale:
- 3*2*2*2*2*2=96 possibili istanze
- mentre |D|=4

L’ipotesi di apprendimento induttivo

- Si noti che, nonostante si voglia trovare una ipotesi h identica a c su tutto X, abbiamo a disposizione solo gli esempi di training D.
- Si cerca una ipotesi h tale che $h(x)=c(x)$ per ogni $x \in D$
- Per questo si assume che:
- Ogni ipotesi trovata che approssima la funzione target correttamente su un insieme di esempi sufficientemente grande approssimera’ bene la funzione target anche sugli esempi non osservati

Terminologia

- Se una istanza x soddisfa una ipotesi h si dice che h **copre** l'esempio x e si scrive h(x)=1. Se l'istanza non soddisfa l'ipotesi allora h(x)=0
  - $h(x)=1 \Leftrightarrow$ copre(h,x)
- Una ipotesi h e' **coerente** con un esempio <x,c(x)> se h(x)=c(x)
- Una ipotesi h e' **coerente con un training set D** (Coerente(h,D)) se e' coerente con ciascun esempio $x \in D$
  - Equivale ad una ipotesi completa e consistente secondo le definizioni date in precedenza

Apprendimento di concetti come ricerca

- L’apprendimento di concetti puo’ essere visto come il compito di effettuare una ricerca attraverso un grande spazio di ipotesi H
- Lo scopo della ricerca e’ di trovare l’ipotesi che meglio copre gli esempi del training set
- Lo spazio delle ipotesi e’ implicitamente definito dalla rappresentazione delle ipotesi

Un esempio

- Una ipotesi h e' una congiunzione di vincoli sugli attributi dell'istanza
- Ogni vincolo puo' essere:
  - un valore specifico
  - un valore qualunque ?
  - Nessun valore, ∅
- H ha 5*4*4*4*4*4=5120 ipotesi sintatticamente distinte
- In realta' pero' ogni ipotesi che contiene uno o piu' ∅ rappresenta l'insieme vuoto di istanze quindi in realta' le ipotesi semanticamente distinte sono 1+4*3*3*3*3*3=973

Ricerca efficiente: come?

- Metodo di ricerca ingenuo: generate-and-test di tutte le ipotesi in H
- Impossibile per spazi di ricerca grandi
- La ricerca puo' basarsi su una struttura definita da una relazione d'ordine da generale a specifico

h_1

Sky	Air Temp	Humid	Wind	Water	Forecast
Sunny	?	?	Strong	?	?

h_2

Sky	Air Temp	Humid	Wind	Water	Forecast
Sunny	?	?	?	?	?

- h_2 e' piu' generale di h_1

Ordinamento da generale a specifico

- Date due ipotesi h_g e h_s

  h_g e' piu' generale o uguale a h_s

$$h_g \geq_g h_s$$

se e solo se ogni istanza che soddisfa h_s soddisfa anche h_g

h_g e' strettamente piu' generale di h_s

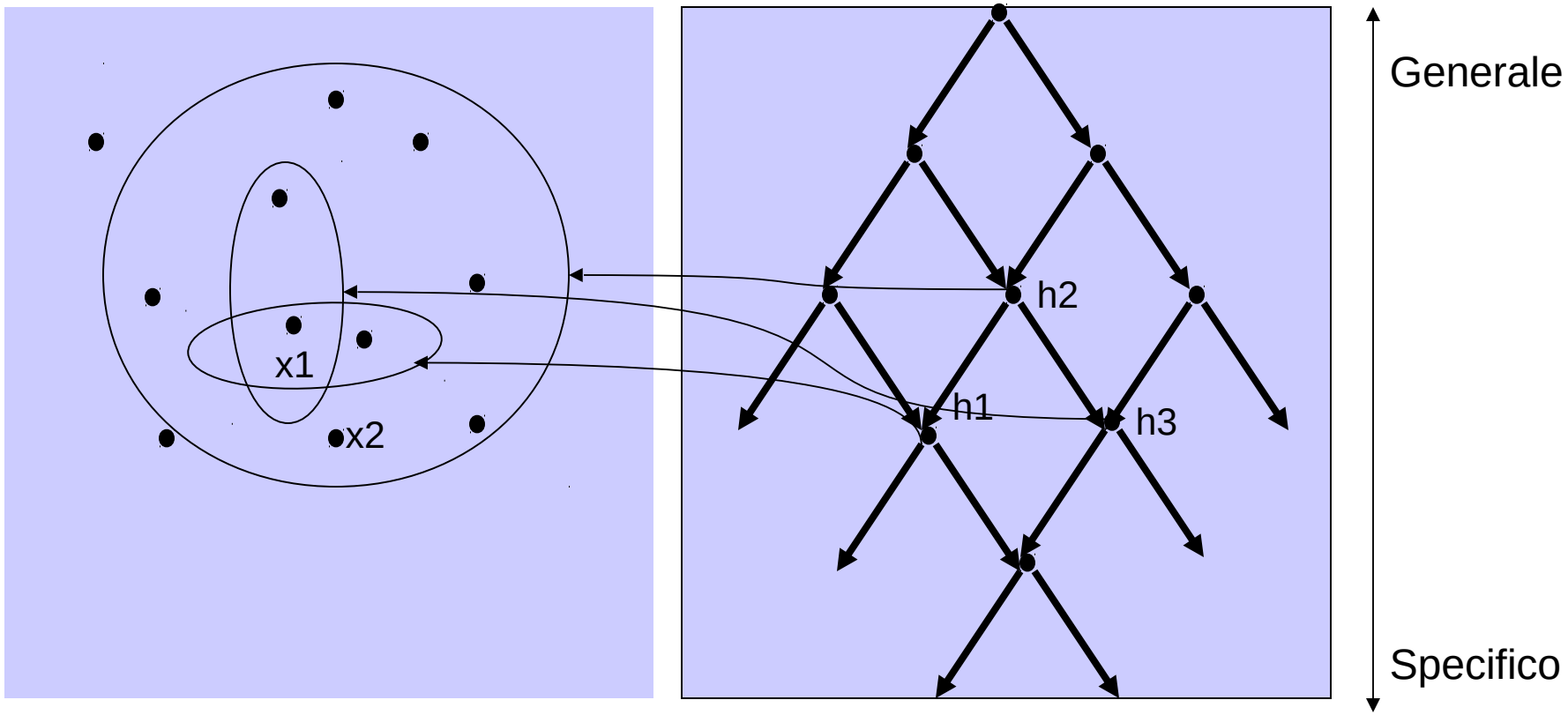
$$h_g >_g h_s$$

se e solo se $h_g \geq_g h_s$ e $h_s \not\geq_g h_g$

La relazione inversa piu' specifico di puo' essere definita in maniera simile

Istanze X

Ipotesi H



x1=<Sunny,Warm,High,Strong,Cool,Same>

x2=<Sunny,Warm,High,Light,Warm,Same>

h1=<Sunny,?,?,Strong,?,?>

h2=<Sunny,?,?,?,?,?>

h3=<Sunny,?,?,?,Cool,?>

Impiego dell'ordinamento da generale a specifico

- Un modo consiste nel cominciare con l'ipotesi piu' specifica in H e poi generalizzare questa ipotesi il meno possibile ogni volta che non copre un esempio positivo del training set (ricerca bottom-up)
- In questo modo si trova una ipotesi massimamente specifica

Algoritmo Find-S

- Inizializza h alla piu' specifica ipotesi in H
- Per ciascun esempio positivo x
  - per ciascun vincolo V_i su un attributo a_i in h
    - se il vincolo V_i e' soddisfatto da x allora non fare niente
    - altrimenti sostituisci V_i in h con il prossimo vincolo piu' generale che e' soddisfatto da x
- Restituisci l'ipotesi h

- Find-S trova sempre una sola soluzione

Esempio

- h←< ∅, ∅, ∅, ∅, ∅, ∅>
- Considera il primo esempio in D:
- <sunny,warm,normal,strong,warm,same>,+
- L'ipotesi piu' specifica che lo copre e'
- h ← <sunny,warm,normal,strong,warm,same>
- Considera il secondo esempio in D:
- <sunny,warm,high,strong,warm,same>,+
- L'ipotesi piu' specifica che copre questo e il precedente e'
- h ← <sunny,warm,?,strong,warm,same>
- Il terzo esempio e' ignorato perché è negativo
- <rainy,cold,high,strong,warm,change>,-
- Considera il quarto esempio in D:
- <sunny,warm,high,strong,cool,change>,+
- L'ipotesi piu' specifica che copre tutti i positivi e'
- h ← <sunny,warm,?,strong,?,?>

Nessuna revisione in caso di esempi negativi: perchè?

- Assunzione di base:
  - il concetto target c e' in H
  - nessun errore nei dati di training
- h e' l'unica ipotesi piu' specifica in H che copre tutti gli esempi positivi
- quindi $c \geq_g h$
- ma c non sara' mai soddisfatto da un esempio negativo
- quindi nemmeno h lo sara'

Limitazioni di Find-S

- Non si puo' dire se il learner ha trovato il concetto target corretto
  - ha trovato l'unica ipotesi in H coerente con i dati oppure ci sono altre ipotesi coerenti?
- Trova una ipotesi massimamente specifica:
  - perche' dovremmo preferire questa ipotesi rispetto, ad esempio, alla piu' generale?

Limitazioni di Find-S (cont.)

- Non puo' determinare se i dati di training sono inconsistenti
  - l'inconsistenza negli esempi di training (e^+ presentato come e^- o viceversa) puo' sviare Find-S, perchè ignora gli esempi negativi. Come si puo' determinare tale inconsistenza?
- Cosa succede se ci possono essere diverse ipotesi coerenti massimamente specifiche?
  - Find-S dovrebbe fare backtracking sulle sue scelte per esplorare diversi rami dell'ordinamento parziale

Spazio delle versioni (Version Space)

- Restituisci uno spazio delle versioni invece di una ipotesi singola
- Lo spazio delle versioni $VS_{H,D}$, rispetto allo spazio delle ipotesi H e al training set D, e' il sottoinsieme delle ipotesi in H coerenti con gli esempi in D
- $VS_{H,D}=\{h \in H | \text{Coerente}(h,D)\}$

Algoritmo List-then-eliminate

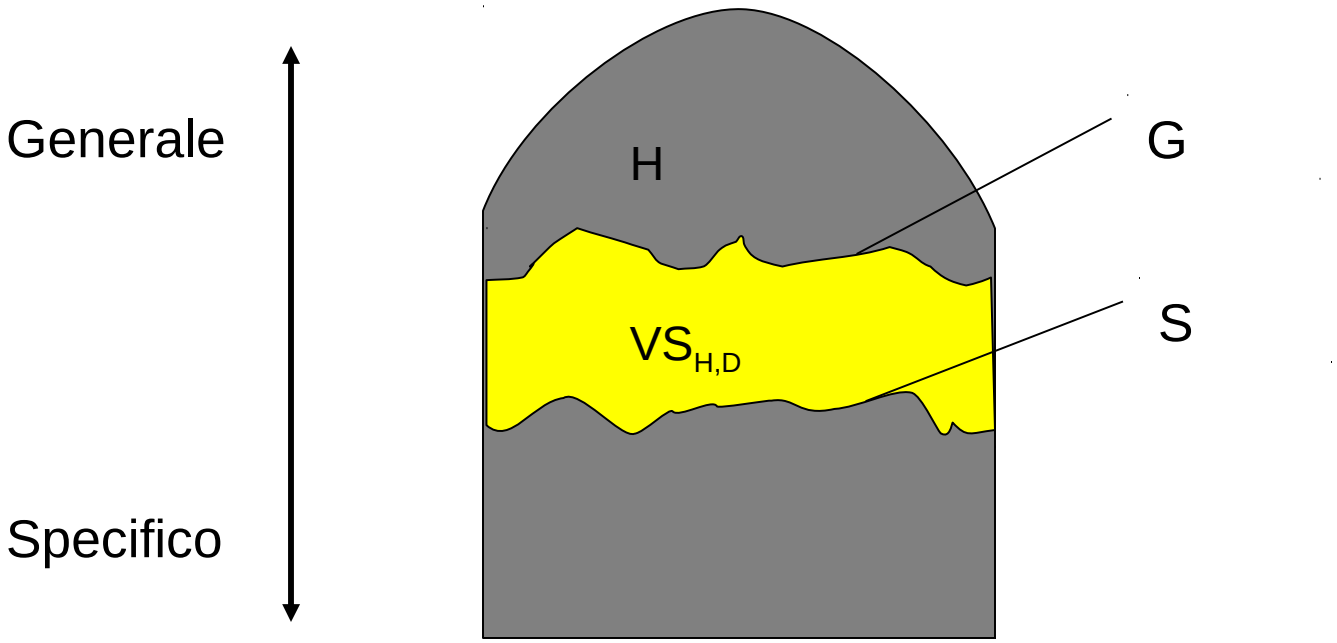
- $VS_{H,D} \leftarrow$ una lista contenente ogni ipotesi in H
- per ciascun esempio <x,c(x)>, rimuovi da $VS_{H,D}$ ogni ipotesi h per la quale $h(x) \neq c(x)$
- restituisci la lista delle ipotesi in $VS_{H,D}$

Vantaggi e svantaggi

- E’ garantito che trovi tutte le ipotesi coerenti con i dati di training
- Può trovare inconsistenze nei dati di training
- Enumerazione esaustiva di tutte le ipotesi:
  - possibile solo per spazi finiti H
  - non realistica per spazi H grandi

Spazio delle versioni: una rappresentazione compatta

- Lo spazio delle versioni puo' essere rappresentato dai suoi membri piu' generali e piu' specifici (teorema di rappresentazione dello spazio delle versioni)



Confine generale

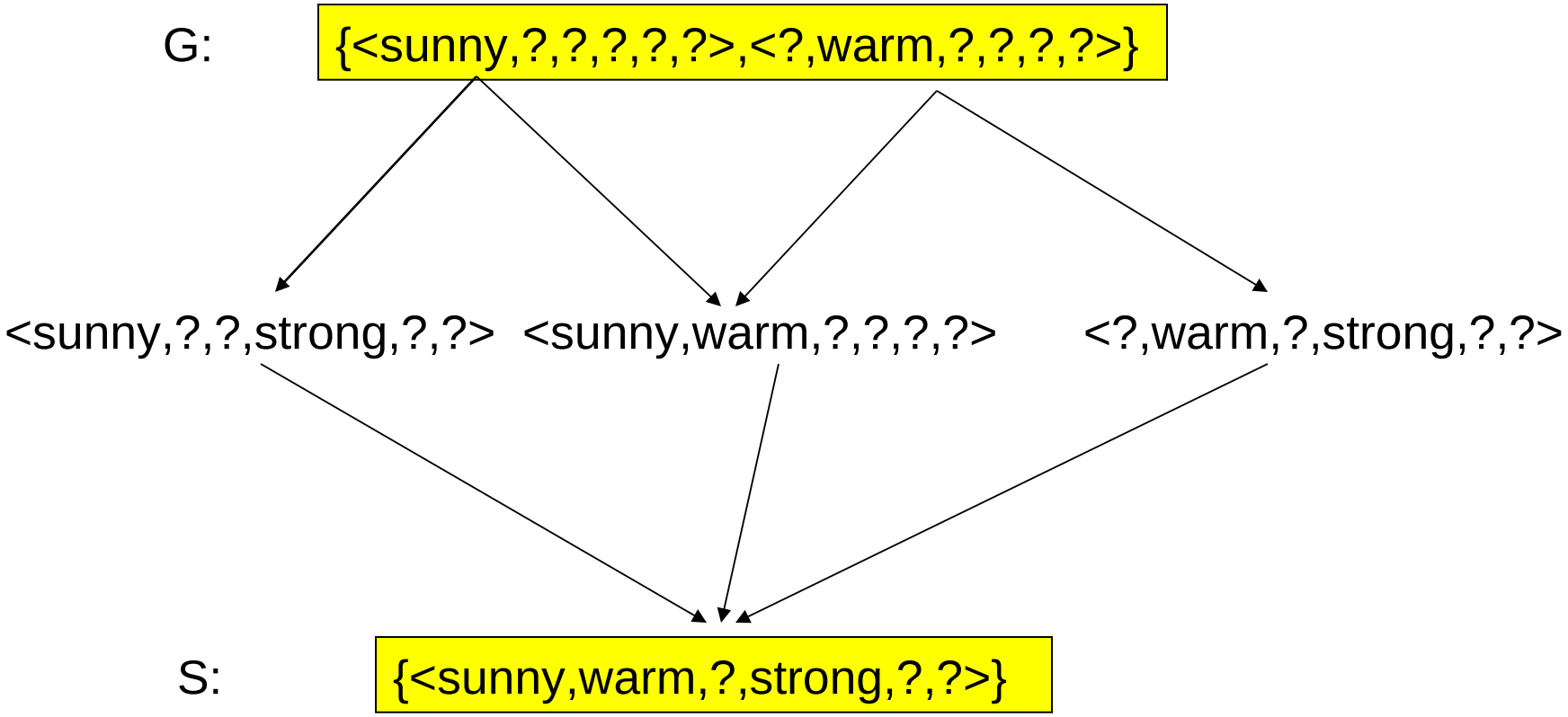
- Il confine generale G, rispetto allo spazio delle ipotesi H e ai dati di training D, e' l'insieme dei membri di H massimamente generali che sono coerenti con D
- $G=\{g \in H | \text{Coerente}(g,D) \text{ and } (\neg \exists g' \in H)[(g' >_g g) \text{ and Coerente}(g',D)]\}$

Confine specifico

- Il confine specifico S, rispetto allo spazio delle ipotesi H e ai dati di training D, e' l'insieme dei membri di H minimamente generali (ovvero massimamente specifici) che sono coerenti con D
- $S=\{s \in H | \text{Coerente}(s,D) \text{ and } (\neg \exists s' \in H)[(s >_g s') \text{ and Coerente}(s',D)]\}$

Uno spazio delle versioni

- Spazio delle versioni per il training set D visto in precedenza



Teorema di rappresentazione dello spazio delle versioni

- Sia X un insieme di istanze e H un insieme di ipotesi booleane definite su X. Sia $c:X \rightarrow \{0,1\}$ un concetto target definito su X e sia D un insieme di esempi di training $\{<x,c(x)>\}$. Per ogni X, H, c e D tali che S e G siano ben definiti:

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

S e G ben definiti

- S e' ben definito se $\forall D,h \in H$, Coerente(h,D)=> $\exists\, s \in S$ tale che $h >_g s$
- G e' ben definito se $\forall D,h \in H$, Coerente(h,D)=> $\exists\, g \in G$ tale che $g >_g h$
- Esempio di S non ben definito:
  - Esempi: numeri reali x
  - Spazio H: ogni ipotesi e' della forma a<x<b dove a e b sono numeri reali e x e' l'istanza, ad esempio 4,1<x<6,5 classifica tutti gli esempi tra 4,1 e 6,5 come positivi
  - Se D={<√2,+>} non esistono ipotesi massimamente specifiche, i.e. S=∅
  - Se modifichiamo H in a≤x≤b allora S e' ben definito, nel caso precedente S={√2 ≤x≤ √2}

Algoritmo Candidate-Elimination

- G←insieme di ipotesi massimamente generali in H
- S←insieme di ipotesi massimamente specifiche in H
- Per ciascun esempio di training d fai:
  - se d e' un esempio positivo
    - togli da G ogni ipotesi incoerente con d
    - UPDATE-S routine: per ciascuna ipotesi s in S che non e' coerente con d
      - togli s da S
      - aggiungi ad S tutte le minime generalizzazioni h di s tali che
        - » h e' coerente con d e
        - » alcuni membri di G sono piu' generali di h
      - Rimuovi da S ogni ipotesi che e' piu' generale di un'altra ipotesi in S

Algoritmo Candidate-Elimination (cont.)

- se d e' un esempio negativo
  - togli da S ogni ipotesi incoerente con d
  - UPDATE-G routine: per ciascuna ipotesi g in G che non e' coerente con d
    - togli g da G
    - aggiungi a G tutte le minime specializzazioni h di g tali che
      - » h e' coerente con d e
      - » alcuni membri di S sono piu' specifici di h
    - Rimuovi da G ogni ipotesi che e' meno generale di un'altra ipotesi in G

UPDATE-S

- Si svolge in due passi:
  1. Prima si trovano le minime generalizzazioni h' di h che sono coerenti con d
  2. Poi si verifica se esiste un membro di G che e' piu' generale di h'
     - se non esiste si elimina h'
- Passo 1:
  - Si considera ogni vincolo di h del tipo attributo=costante e si verifica se e' rispettato da d, se non lo e', si sostituisce in h' il vincolo con attributo=?

Esempio di UPDATE-S

- h=<sunny, warm, normal, strong, warm, same>
- d=<sunny,warm,high,strong,warm,same>,EnjoySport =yes
- h'= <sunny, warm, ?, strong, warm, same>

UPDATE-G

- Si svolge in due passi:
  1. Prima si trovano le minime specializzazioni h' di h che sono coerenti con d
  2. Poi, per ogni h', si verifica se esiste un membro di S che e' piu' specifico di h'
     - se non esiste si elimina h'
- Passo 1:
  - Si considera ogni vincolo di h del tipo attributo=? e si genera un h' in cui il vincolo e' sostituito con un vincolo attributo=costante con costante diversa da quella presente in d

Esempio di UPDATE-G

- h=<?,?,?,?,?,?>
- d=<rainy,cold,high,strong,warm,change>,EnjoySport =no
- h1=<sunny,?,?,?,?,?>
- h2=<cloudy,?,?,?,?,?>
- h3=<?,warm,?,?,?,?>
- h4=<?,?,normal,?,?,?>
- h5=<?,?,?,weak,?,?>
- h6=<?,?,?,?,cool,?>
- h7=<?,?,?,?,?,same>

Esempio di esecuzione

G0: {<?,?,?,?,?,?>}

S0: {<∅, ∅, ∅, ∅, ∅, ∅ >}

Esempio:

1 <sunny,warm,normal,strong,warm,same>,EnjoySport=yes

Esempio di esecuzione

Esempio:

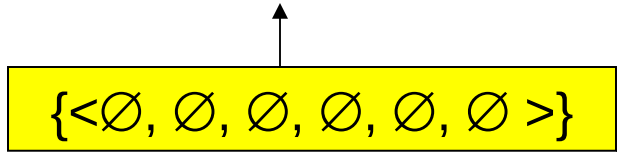
1 <sunny,warm,normal,strong,warm,same>,EnjoySport=yes

- UPDATE-S:
  - Passo 1:
    - h= <∅, ∅, ∅, ∅, ∅, ∅ >
    - Minima generalizzazione h'=<sunny,warm,normal,strong,warm,same>
  - Passo 2:
    - <?,?,?,?,?,?> è più generale di h' quindi la si tiene

Esempio di esecuzione

G0, G1: {<?,?,?,?,?,?>}

S1: {<sunny, warm, normal, strong, warm, same >}



Esempio:

2 <sunny,warm,high,strong,warm,same>,EnjoySport=yes

Esempio di esecuzione

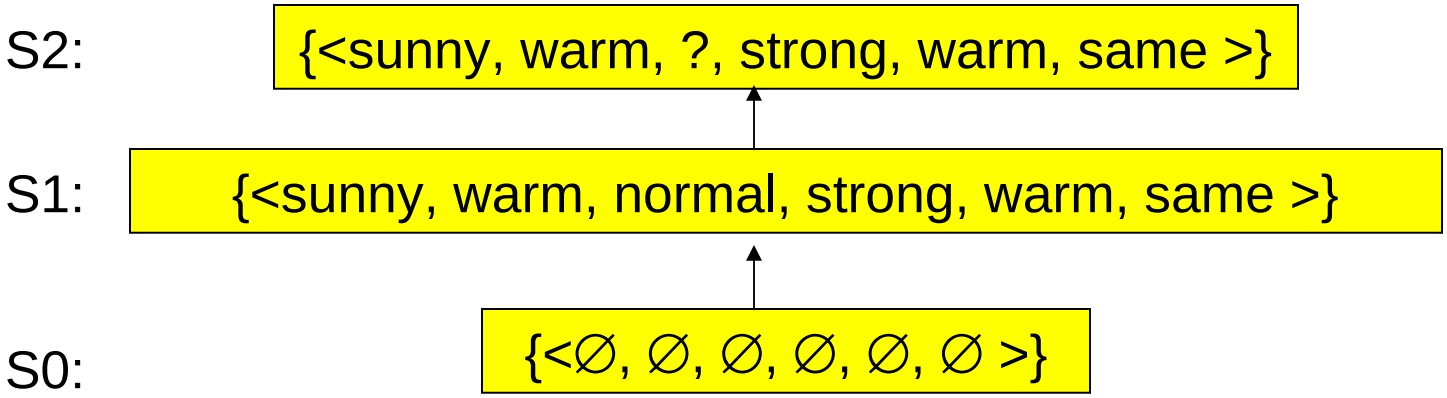
Esempio:

2 <sunny,warm,high,strong,warm,same>,EnjoySport=yes

- UPDATE-S:
  - Passo 1:
    - h=<sunny,warm,normal,strong,warm,same>
    - Minima generalizzazione h'=<sunny,warm,?,strong,warm,same>
  - Passo 2:
    - <?,?,?,?,?,?> è più generale di h' quindi la si tiene

Esempio di esecuzione

G0, G1, G2: {<?,?,?,?,?,?>}



Esempi:

1 <sunny,warm,normal,strong,warm,same>,EnjoySport=yes

2 <sunny,warm,high,strong,warm,same>,EnjoySport=yes

Esempio di esecuzione

G2: {<?,?,?,?,?,?>}

S2: {<sunny, warm, ?, strong, warm, same >}

Esempio:
3 <rainy,cold,high,strong,warm,change>,EnjoySport=no

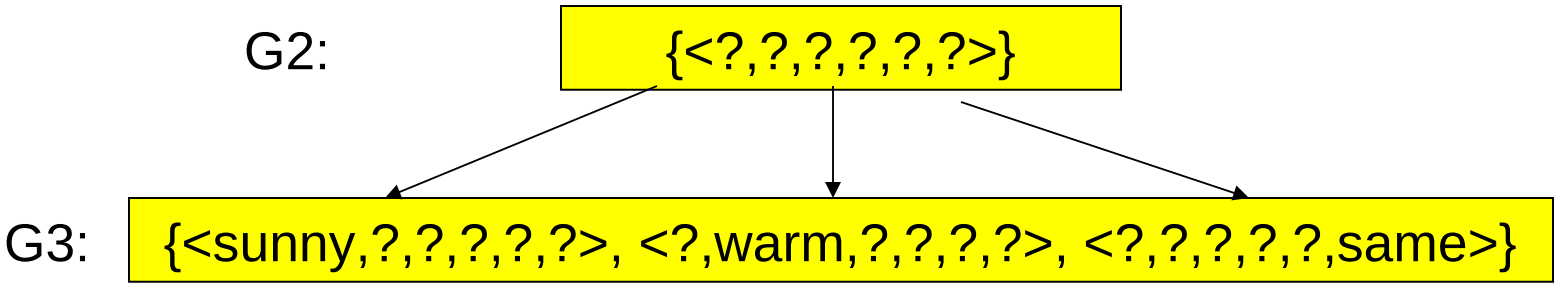
Esempio di esecuzione

Esempio:

3 <rainy,cold,high,strong,warm,change>,EnjoySport=no

- UPDATE-G:
  - Passo 1:
    - h= <?,?,?,?,?,?>
    - Minime specializzazioni:
    - h1=<sunny,?,?,?,?,?> h2=<cloudy,?,?,?,?,?>
    - h3=<?,warm,?,?,?,?> h4=<?,?,normal,?,?,?>
    - h5=<?,?,?,weak,?,?> h6=<?,?,?,?,cool,?>
    - h7=<?,?,?,?,?,same>
  - Passo 2:
    - <sunny, warm, ?, strong, warm, same > è più specifica solo di h1, h3, h7 quindi si tengono solo quelle

Esempio di esecuzione



S2, S3: {<sunny, warm, ?, strong, warm, same >}

Esempio:

3 <rainy,cold,high,strong,warm,change>,EnjoySport=no

Esempio di esecuzione

G3: {<sunny,?,?,?,?,?>, <?,warm,?,?,?,?>, <?,?,?,?,?,same>}

S3: {<sunny, warm, ?, strong, warm, same >}

Esempio:

4 <sunny,warm,high,strong,cool,change>,EnjoySport=yes

Esempio di esecuzione

Esempio:

4 <sunny,warm,high,strong,cool,change>,EnjoySport=yes

- Si toglie da G3 <?,?,?,?,?,same> perchè è incoerente con l'esempio
- UPDATE-S:
  - Passo 1:
    - h= <sunny, warm, ?, strong, warm, same >
    - Minima generalizzazione h'=<sunny,warm,?,strong,?,?>
  - Passo 2:
    - <sunny,?,?,?,?,?> è più generale di h' quindi la si tiene

Esempio di esecuzione

G3: {<sunny,?,?,?,?,?>, <?,warm,?,?,?,?>, <?,?,?,?,?,same>}

G4: {<sunny,?,?,?,?,?>, <?,warm,?,?,?,?>}

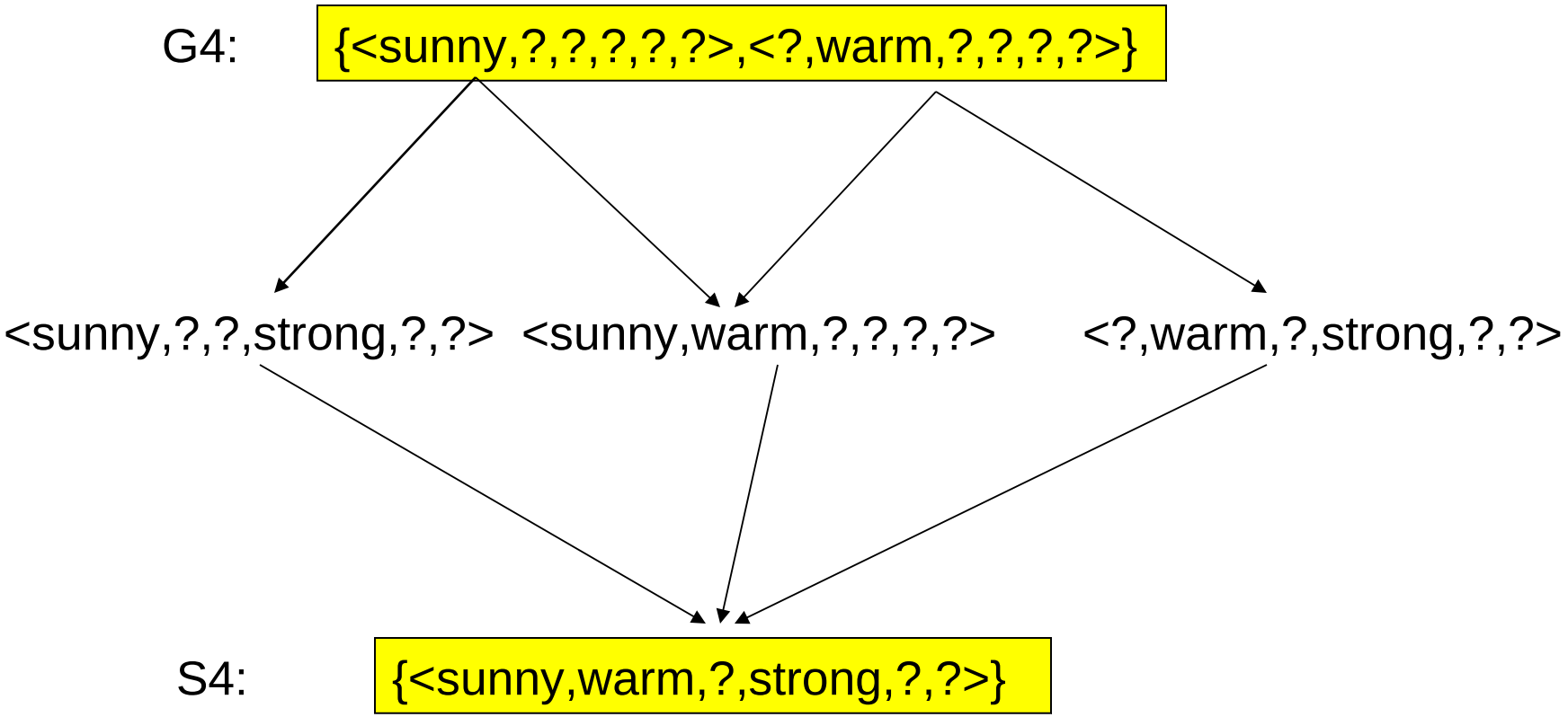
S4: {<sunny, warm, ?, strong, ?, ? >}

S3: {<sunny, warm, ?, strong, warm, same >}

Esempio:

4 <sunny,warm,high,strong,cool,change>,EnjoySport=yes

Esempio di esecuzione: risultato



A che cosa converge l'algoritmo Candidate-Elimination?

- Il concetto target e' appreso quando S e G convergono ad una singola, identica, ipotesi.

- L'algoritmo convergera' verso il concetto target

  purche'

  - non ci siano errori negli esempi di training
  - ci sia qualche ipotesi in H che descriva correttamente il concetto target
  - ci siano abbastanza esempi

Osservazioni

- Cosa succede se i dati di training contengono degli errori?
  - Supponiamo che un esempio abbia una errata classificazione. In questo caso, l'algoritmo rimuove il concetto target corretto dallo spazio delle versioni
  - Dato un sufficiente numero di esempi, l'algoritmo scoprira' l'inconsistenza producendo uno spazio delle versioni vuoto
- L'algoritmo produce uno spazio delle versioni vuoto anche quando il concetto target non puo' essere descritto nel linguaggio di rappresentazione delle ipotesi

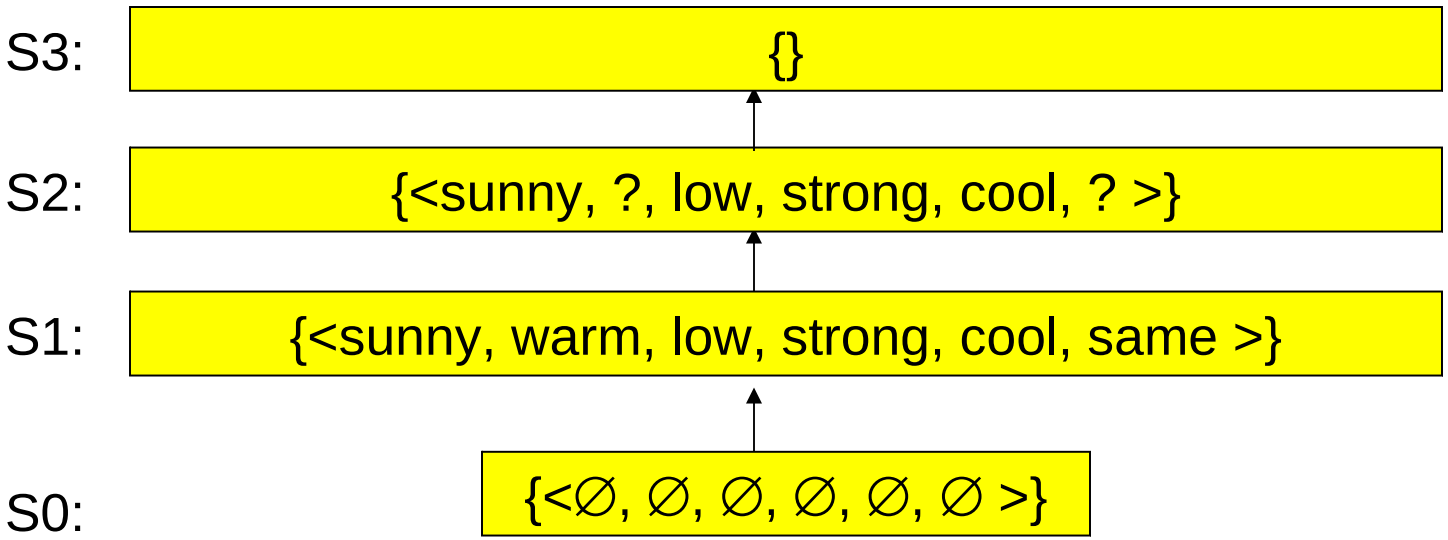
Esempio

1<sunny,warm,low,strong,cool,same>EnjoySport=yes

2<sunny,cold,low,strong,cool,change>EnjoySport=yes

3<sunny,cold,low,strong,cool,same> EnjoySport=no

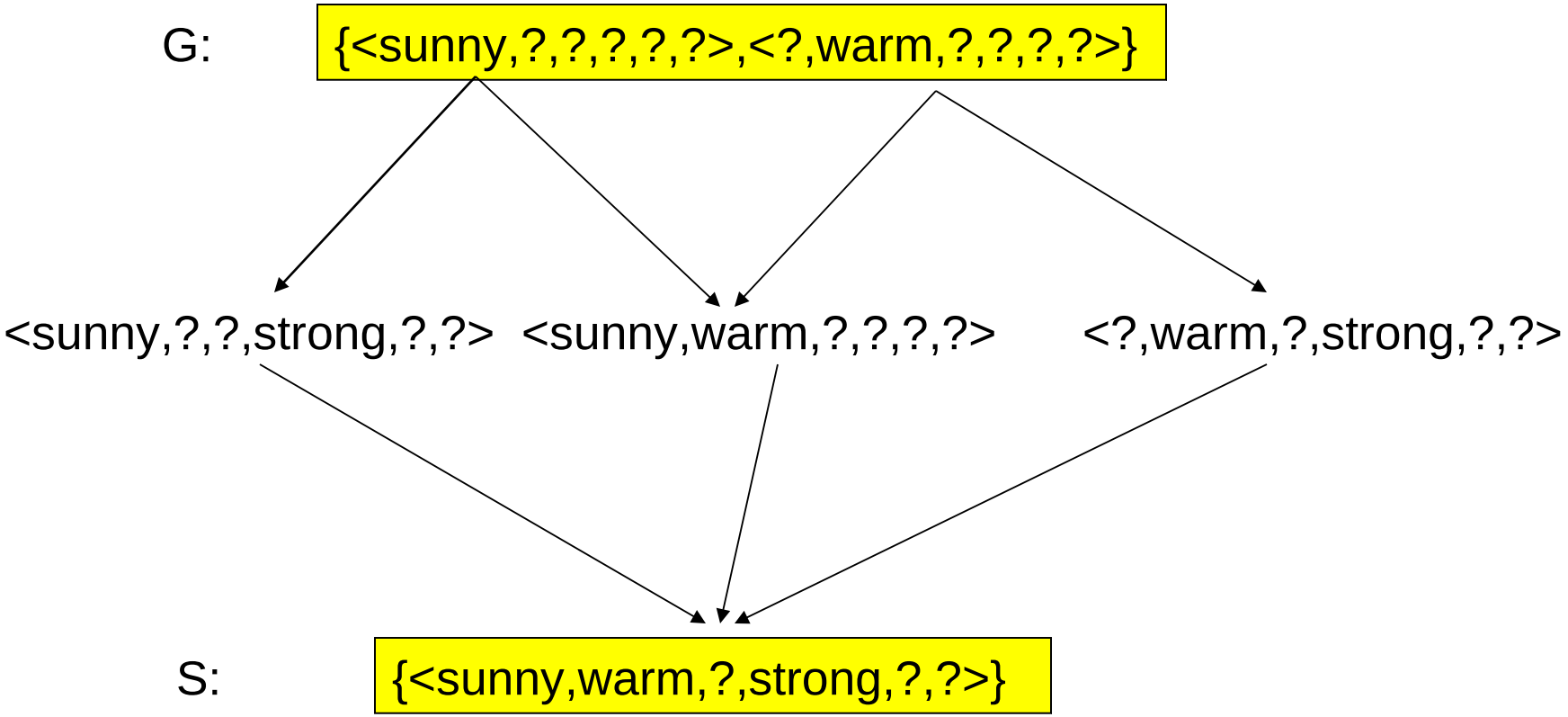
G0, G1, G2: {<?,?,?,?,?,?>}



Quali esempi richiedere?

- Si supponga che il learner possa condurre esperimenti in cui sceglie la prossima istanza e ne ottiene la classificazione, quali istanze dovrebbe scegliere?
- Dovrebbe scegliere istanze che soddisfino meta' delle ipotesi nello spazio delle versioni. In questo modo lo spazio delle versioni e' ridotto di un mezzo ad ogni nuovo esempio e per apprendere correttamente il concetto target sono necessari solo $\lceil \log_2 |VS| \rceil$ esempi.
- Generalmente, e' impossibile adottare questa strategia di ricerca ottimale

Come usare concetti parziali

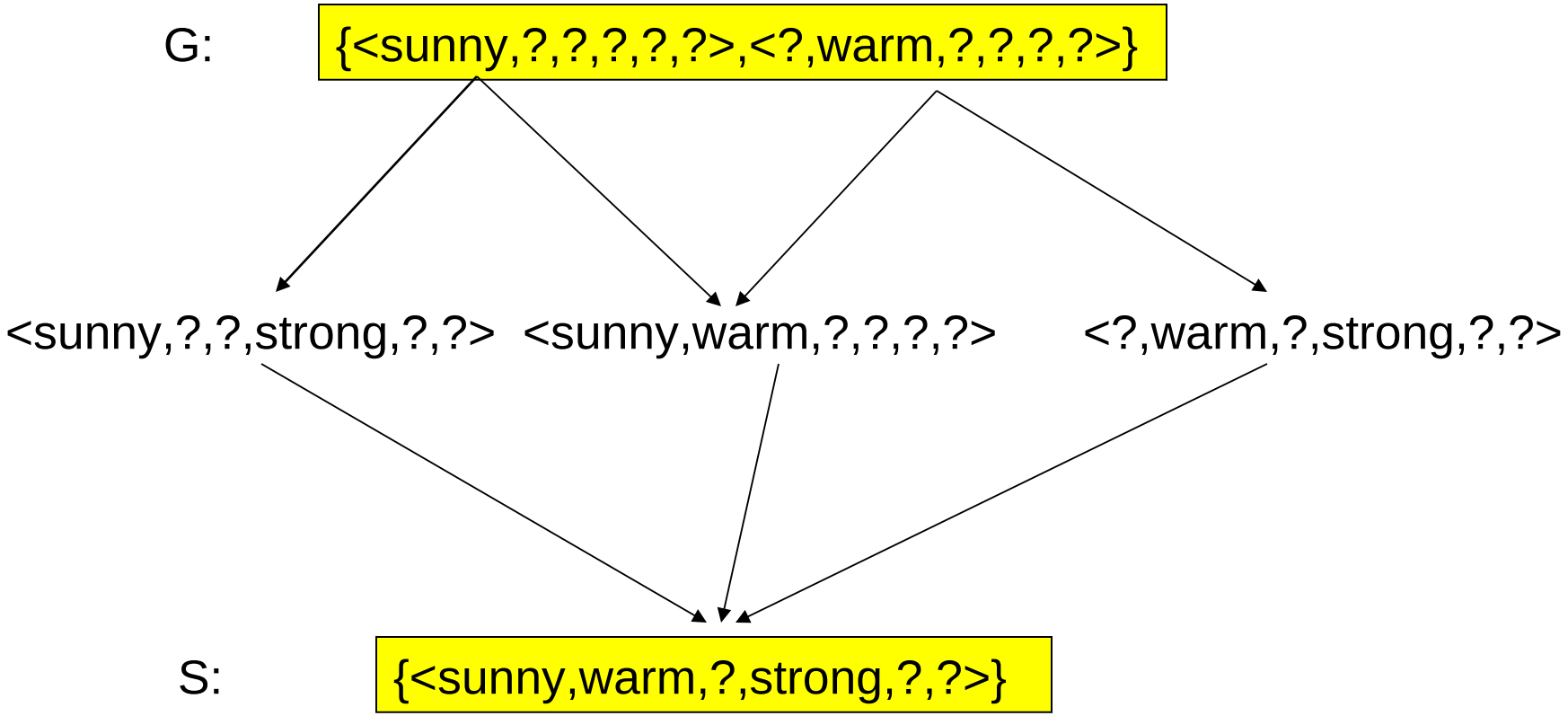


<sunny,warm,normal,strong,cool,change>

Positivo a giudizio unanime->positivo

(basta verificare che soddisfi tutte le ipotesi in S)

Come usare concetti parziali

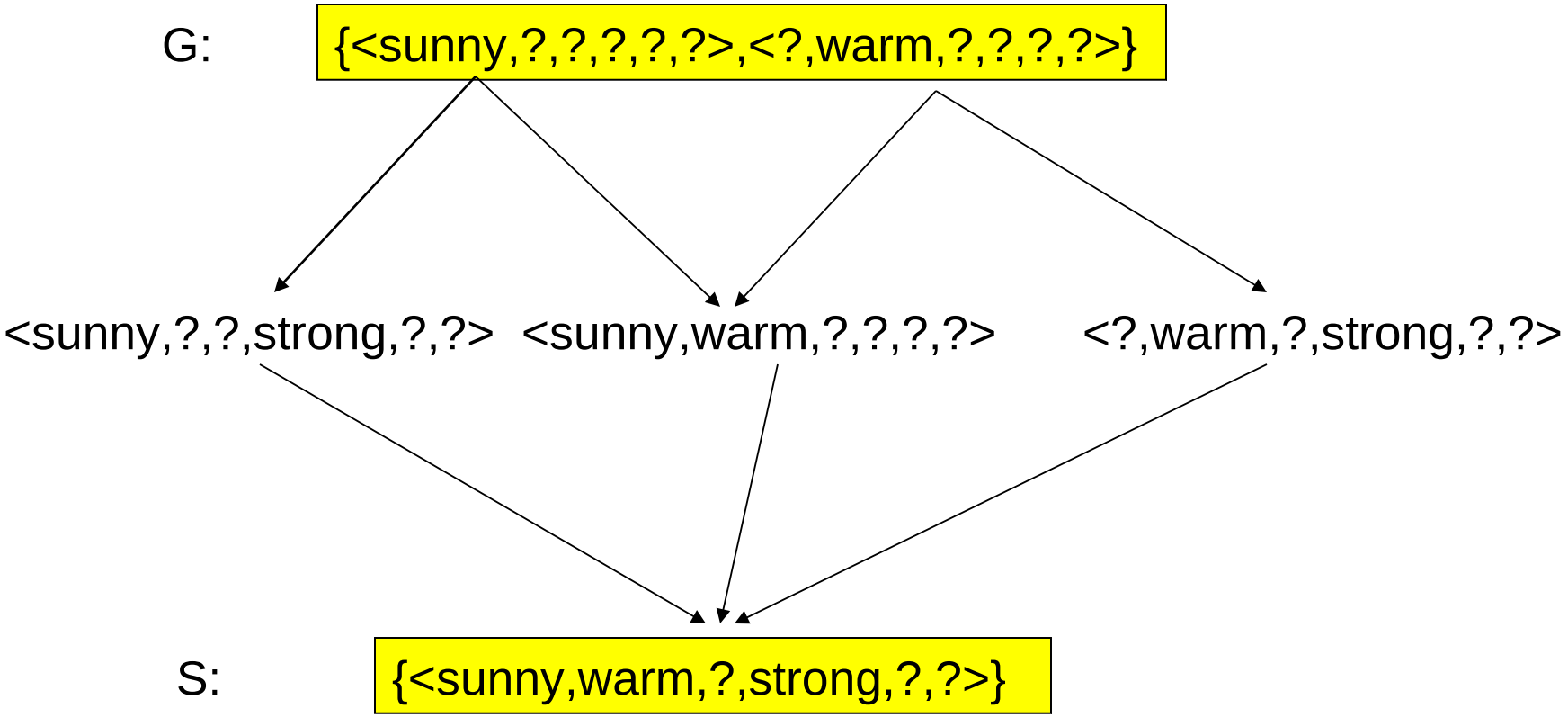


<rainy,cold,normal,light,warm,same>

Negativo a giudizio unanime->negativo

(basta verificare che non soddisfi nessuna ipotesi in G)

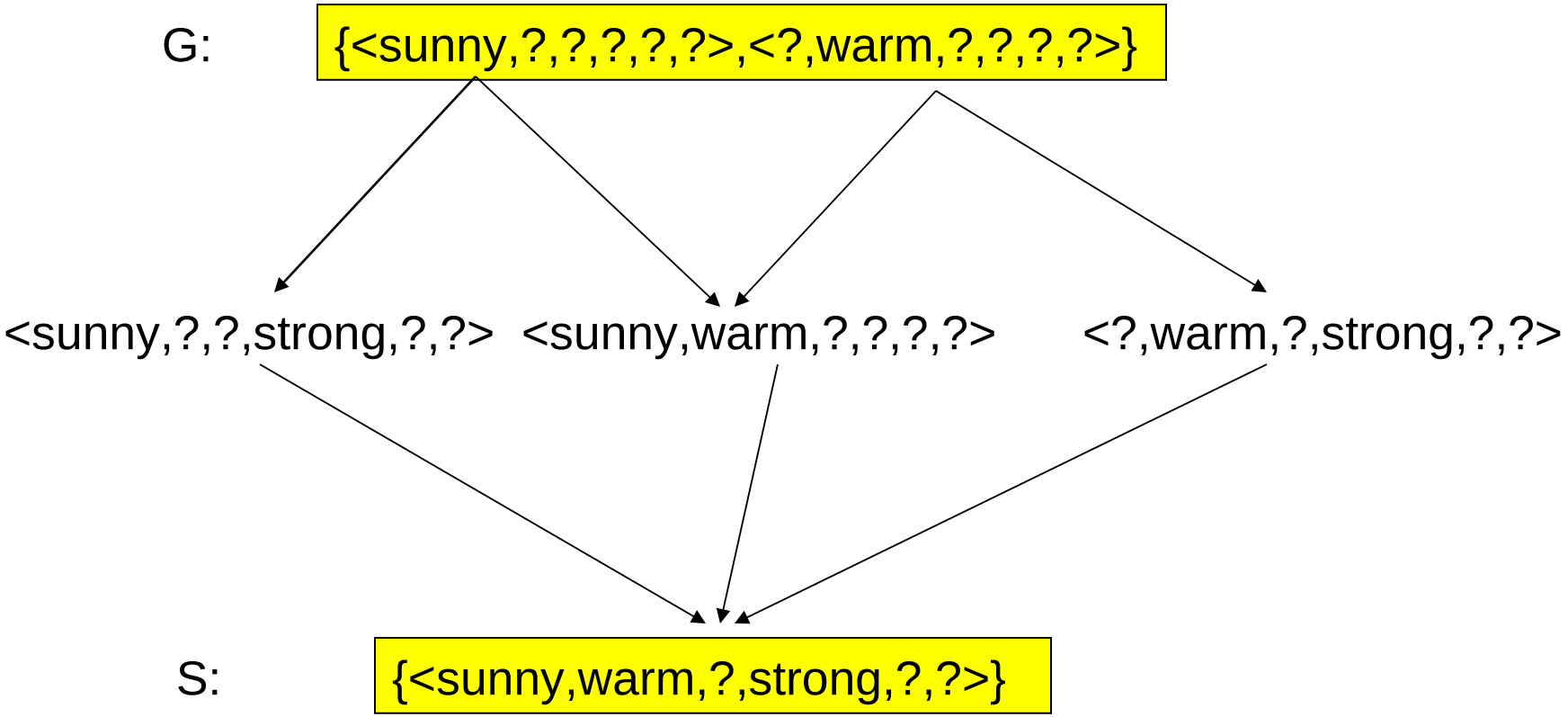
Come usare concetti parziali



<sunny,warm,normal,light,warm,same>

Meta' positivi, meta' negativi->non puo' essere classificato

Come usare concetti parziali



<sunny,cold,normal,strong,warm,same>

Due positivi, quattro negativi-> si puo' classificare come negativo (voto di maggioranza). La proporzione di positivi e negativi puo' essere usata come misura della confidenza nella classificazione

Cosa succede se il concetto non è contenuto nello spazio delle ipotesi

- Non si possono rappresentare concetti disgiuntivi come “sky=sunny or sky=cloudy”

Sky	Air Temp	Humid	Wind	Water	Forecast	Enjoy Sport
Sunny	Warm	Normal	Strong	Cool	Change	Yes
Cloudy	Warm	Normal	Strong	Cool	Change	Yes
Rainy	Warm	Normal	Strong	Cool	Change	No

- E’ richiesto uno spazio delle ipotesi piu’ espressivo

Uno spazio delle ipotesi che include ogni possibile ipotesi?

- Scegli H' in modo che esprima ogni concetto insegnabile (cioè, H' è il power set di X)
- H' puo' essere ottenuto in questo modo: H'=disgiunzioni, congiunzioni, negazioni di h $\in$ H
- <sunny,?,?,?,?,?,?>$\vee$<cloudy,?,?,?,?,?,?>
- Che cosa sono S e G in questo caso?

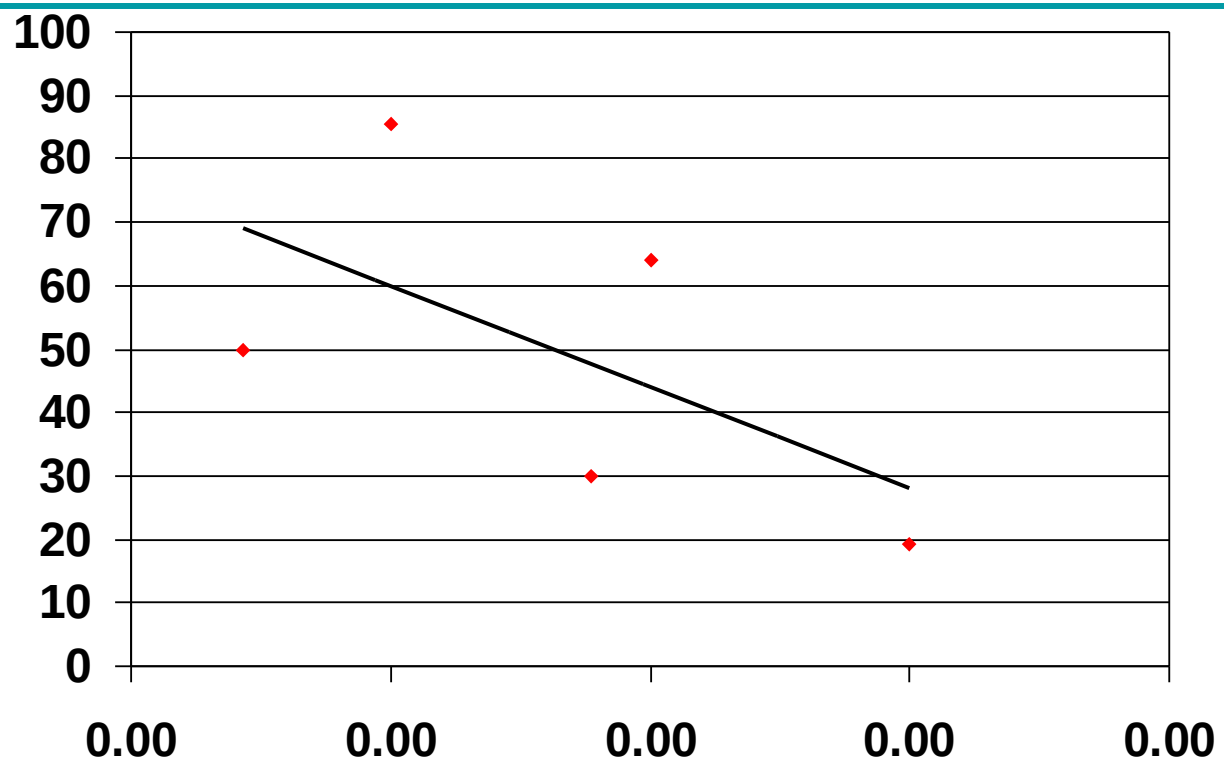
- S e' la disgiunzione degli esempi positivi visti finora
- G e' la negazione della disgiunzione degli esempi negativi visti finora

Completamente inutile. Nessuna generalizzazione!

Una proprietà fondamentale dell'inferenza induttiva

- Un algoritmo di apprendimento che non fa nessuna assunzione a priori riguardo l'identità del concetto target non ha nessuna base razionale per classificare le istanze non viste.
- Assunzione a priori=inductive bias
- Inductive bias dell'algoritmo Candidate-Elimination: il concetto target appartiene allo spazio delle ipotesi.
- Se questa assunzione e' corretta (e gli esempi sono privi di errori) la classificazione di nuove istanze sara' corretta
- Se non e' corretta, e' sicuro che Candidate-Elimination classifichera' male alcune istanze di X

Esempio: regressione lineare



- L’assunzioni a priori sottostante (cioè l’inductive bias) è che la relazione tra X e Y sia lineare

Una definizione formale di bias induttivo

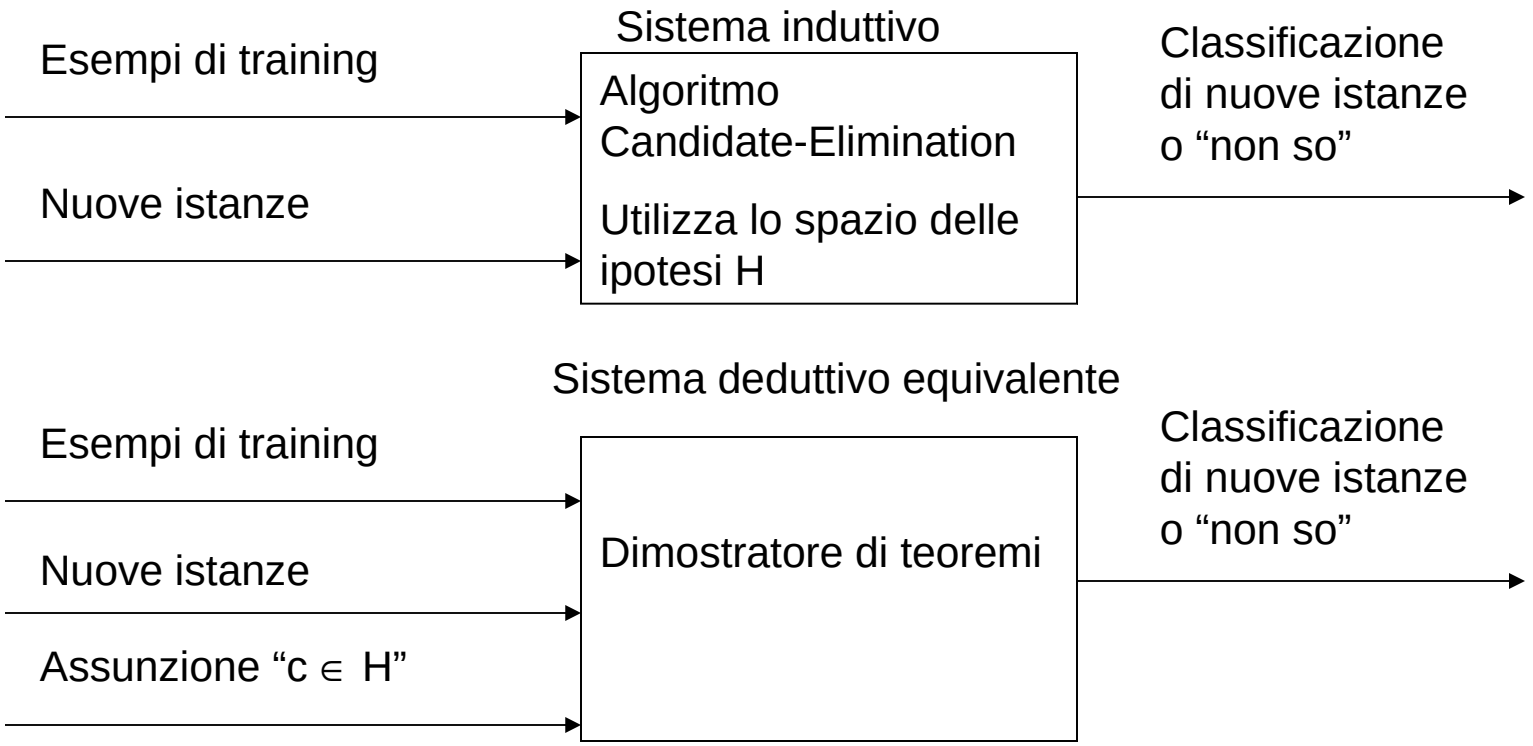
- Si consideri:
  - un algoritmo per l'apprendimento di concetti L
  - un insieme di istanze X
  - un concetto target c
  - un insieme di esempi D_c={<x,c(x)>}
  - sia $L(x_j,D_c)$ la classificazione assegnata all'istanza x_j da L dopo il training sui dati D_c
- Il bias induttivo di L è ogni minimo insieme di asserzioni B tali che per ciascun concetto target c e corrispondenti esempi di training D_c

$$(\forall x_j \in X)[(B \wedge D_c \wedge x_j) \vdash c(x_j)=L(x_j,D_c)]$$

Inductive bias di Candidate-Elimination

- Si consideri il seguente algoritmo: dati D_c, l'algoritmo Candidate-Elimination genera prima lo spazio delle versioni e poi classifica una nuova istanza x_i mediante voto delle ipotesi nello spazio delle versioni. Supponiamo che produca una classificazione solo se il voto e' unanime, altrimenti non produca una classficazione.
- L'inductive bias di questo algoritmo e' l'assunzione che $c \in H$
- Da $c \in H$ segue che $c \in VS_{H,Dc}$
- L genera la classificazione $L(x_i,D_c)$ se e solo se ogni ipotesi in $VS_{H,Dc}$ produce questa classificazione, inclusa c, quindi $c(x_i)= L(x_i,D_c)$

Modellazione di sistemi induttivi mediante sistemi deduttivi equivalenti



- Il bias induttivo reso esplicito nel sistema deduttivo e’ implicito in quello induttivo

Usi del bias induttivo

- Il bias induttivo e' un modo non procedurale per caratterizzare la politica dell'algoritmo di apprendimento per generalizzare oltre i dati osservati
- Il bias induttivo consente di confrontare diversi algoritmi di apprendimento sulla base del bias induttivo che adottano. Esempi di bias induttivi di forza crescente:
  - Rote-learner: memorizza gli esempi. Classifica x solo se e' uguale ad un esempio memorizzato =>Nessun bias induttivo
  - Candidate-Elimination: $c \in H$
  - Find-S: $c \in H$ + tutte le istanze sono negative a meno che non si provi l'opposto

Bias induttivo

- Piu’ il bias induttivo e’ forte, maggiore e’ il numero delle istanze che vengono classificate

Software

- L’implementazione in Prolog dell’algoritmo CandidateElimination è disponibile nella sezione software del sito del corso

Bibliografia

[Ber96] F. Bergadano e D. Gunetti, *Inductive Logic Programming - From Machine Learning to Software Engineering*, MIT Press, Cambridge, Massachusetts, 1996

[Mit97] T. M. Mitchell, *Machine Learning*, McGraw-Hill, 1997

[Michalski 1986] Michalski, R. S. “Understanding the nature of learning: Issues and research directions” in Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, Machine Learning - An Artificial Intelligence Approach, Volume II, Morgan Kaufmann Publishers, Los Altos, California, pages 3—26, 1986.

Bibliografia

[Simon 1984] Simon, H. A. “Why should machines learn” In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, Machine Learning - An Artificial Intelligence Approach, Springer-Verlag, Berlin, pages 25—37, 1984.