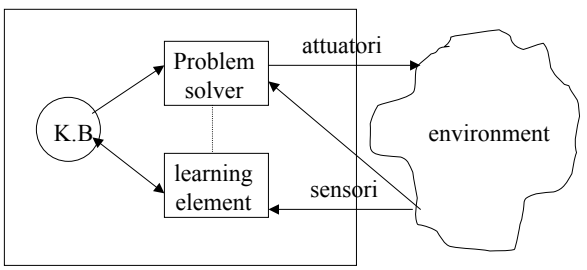
Machine Learning

- Capacità di apprendimento è fondamentale in un sistema "intelligente" per
  - risolvere nuovi problemi
  - non ripetere gli errori fatti in passato
  - risolvere problemi più efficientemente o in modo migliore
  - autonomia nell'affrontare e risolvere problemi (indipendenza da esperto che fornisce la conoscenza)
  - adattarsi ai cambiamenti dell'ambiente circostante
- Un sistema senza queste caratteristiche difficilmente potrebbe essere considerato intelligente
- Machine learning
  - importante area della AI a partire dagli anni '60
  - importanti risultati: vari approcci all'apprendimento
  - importanti relazioni con altre aree di ricerca, non solo della AI o dell'informatica



- Learning element estende la K.B.
  - Interagendo con l'ambiente
  - usando il problem solver
  - usando conoscenza già disponibile
- Molti approcci per definire il "learning element"
  - obiettivo del processo di apprendimento
  - tipo di informazione disponibile
  - tipo di conoscenza disponibile
  - forma della conoscenza da apprendere

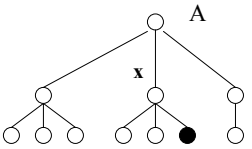
- Ad esempio
  - "rote learning": semplice caching di casi da riusare in futuro
    - evoluzione verso il Case-based Reasoning (CBR)
  - learning senza conoscenza a priori
    - learning da esempi (induttivo o generalizzazione)
  - learning con conoscenza a-priori
    - learning da esempi (basato sulla spiegazione)

Rote learning

- Forma elementare di apprendimento
- alla fine della soluzione di un problema
  - problem solver passa la soluzione al learning element per il caching
- all'inizio nella soluzione di un problema si guarda se per caso non c'è già la soluzione in cache
  - se c'è: retrieve
  - se non c'e: risolvi il problema

- Esempio: strategie per giochi
  - min-max (con alfa-beta) per scegliere la mossa migliore in una posizione
    - espansione dell'albero fino ad un livello N prefissato
    - euristica sulle foglie
    - propagazione verso l'alto e scelta della mossa migliore
    - esempio

N=2

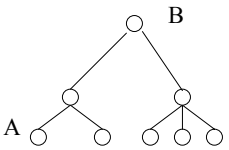


euristica migliore (es. 10)

in posizione A, x è la mossa migliore con stima euristica: 10

- memorizzazione (caching) di questo risultato
  - ogni volta che ci si trova in posizione A non si deve espandere di nuovo l'albero ma si trova la soluzione nella cache
  - molto utile anche in posizioni intermedie: permette di migliorare significativamente nel tempo le prestazioni

- Esempio



  - in A invece di usare la stima euristica si usa il valore nella cache: la mossa migliore ha valore 10
  - in questo modo è come se l'espansione avvenisse a più livelli
- Forma molto elementare di apprendimento
  - non particolarmente "intelligente"
  - funziona solo se in futuro si ripete la stessa situazione
- Forme più evolute di caching dei risultati studiati in anni più recenti case-based reasoning
  - memorizzazione di casi attraverso "features"
  - meccanismi di generalizzazione dei casi e delle soluzioni
  - tecniche di gestione della memoria dei casi, di match tra casi e di retrieval di casi
- Instance-based Learning

Apprendimento induttivo

- Apprendimento a partire da esempi forniti da un supervisore
- Induzione: una possibile visione matematica
  - dato un insieme di coppie $<a_i, f(a_i)>$
  - determinare quale è la funzione f(x) di cui le coppie sono istanze
- Terminologia:
  - Esempio = una istanza $<a_i, f(a_i)>$ della funzione da apprendere
  - Contro-esempio = una coppia $<a_j,b>$ per cui si ha che $b \neq f(a_j)$
  - Ipotesi = una possibile definizione h per la funzione da apprendere
  - Spazio di ipotesi = l'insieme delle ipotesi possibili
  - Bias = un criterio di preferenza tra le ipotesi
    - ad esempio forma sintattica delle funzioni
- Induzione = generalizzazione da esempi

  obiettivo: trovare una ipotesi h che sia in accordo con tutti gli esempi e i contro-esempi
- Induzione: forma di inferenza logica

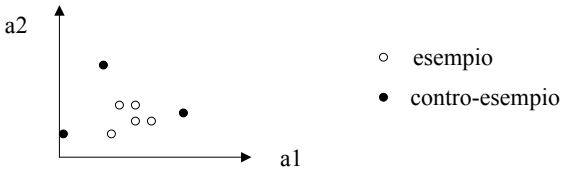
- Concept learning: un'altra visione dell'induzione
  - problema:
    - dato insieme di concetti (classi di oggetti)
    - dato un insieme di esempi e contro-esempi per i concetti
      - esempi: istanze del concetto
      - contro-esempi: individui che non sono istanze del concetto
    - apprendere la descrizione dei concetti
  - esempio: classificazione (vedremo)
- Molti approcci all'apprendimento induttivo
  - tipo di conoscenza da apprendere (e bias)
  - tipo di esempi
  - diverse forme (più o meno forti) di generalizzazione
- Vedremo
  - apprendimento di alberi decisionali per la classificazione
  - apprendimento di descrizioni strutturate di concetti
    - reti semantiche
    - formule logiche: version space
  - apprendimento in reti neurali

Classificazione

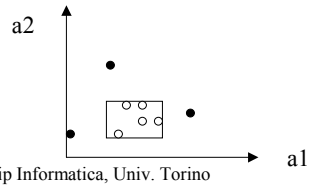
- Forma di problem solving comune in molti task
- dato un insieme di classi C1, C2, …, Cn
  - per ogni classe è fornita una definizione (descrizione)
    per ognuna delle Ci è dato un insieme di features fi1, …, fin
- dato un oggetto a: stabilire a quale delle classi appartiene
- Apprendimento per classificazione:
  - dato un insieme di esempi (e controesempi)
  - apprendere la descrizione delle classi
- Esempio
  - descrizione di classi
    - classe C1=auto: ruote=4, motore=presente
    - classe C2=bici: ruote=2, motore=assente
  - esempi
    - a=auto motore=presente
    - a not = auto ruote=2
- Molte possibili rappresentazioni delle classi e approcci alla classificazione

Una visione grafica

- Classe C descritta da due attributi a1 e a2
- rappresentazione grafica di esempi e contro-esempi

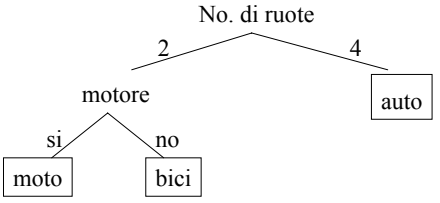


- Concetto da apprendere: figura nel piano di a1, a2 che
  - copre tutti gli esempi
  - non copre contro-esempi
- bias: forma geometrica della figura
- esempio



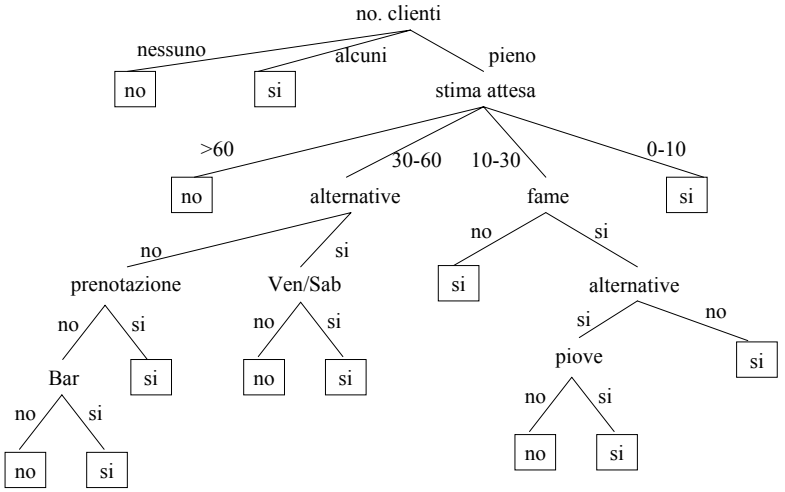
Alberi decisionali

- Una rappresentazione operazionale per un sistema di decisione (e di classificazione)
- Alberi in cui
  - i nodi con figli corrispondono a domande (attributi)
  - gli archi che escono dai nodi alle possibili risposte (valori degli attributi)
  - le foglie sono decisioni (le classi)
- Esempio



- Una versione più semplice: albero di decisione per un concetto booleano (appartenenenza a una classe)

- Esempio [Russel, Norvig]: entrare in un ristorante?



si e no sono le due classi di decisione

- Apprendimento: costruire l'albero a partire da esempi
  - esempi di decisioni
    - si/no (esempi e contro-esempi) nel caso di decisioni binarie
  - esempi delle varie classi corrispondenti alle foglie
- Esempio:

Es.	**attributi**										**Dec**
	alt.	bar	V/S	fame	noC	prez	piov	pren	tipo	att	
x1	si	no	no	si	alc	£££	no	si	F	0-10	Si
x2	si	no	no	si	pieno	£	no	no	Thai	30-60	No
x3	no	si	no	no	alc	£	no	no	hamb	0-10	Si
x4	si	no	si	si	pieno	£	no	no	Thai	10-30	Si
x5	si	no	si	no	pieno	£££	no	si	F	>60	No
x6	no	si	no	si	alc	££	si	si	I	0-10	Si
x7	no	si	no	no	ness	£	si	no	hamb	0-10	No
x8	no	no	no	si	alc	££	si	si	Thai	0-10	Si
x9	no	si	si	no	pieno	£	si	no	hamb	>60	No
x10	si	si	si	si	pieno	£££	no	si	I	10-30	No
x11	no	no	no	no	ness	£	no	no	Thai	0-10	No
x12	si	si	si	si	pieno	£	no	no	hamb	30-60	Si

- Una soluzione banale
  - costruire un ramo per ognuno degli esempi (una forma di rote learning)
- Obiettivo
  - costruire un albero "semplice"
  - Occam's razor: il più semplice tra quelli consistenti con gli esempi
    - in generale indecidibile
    - euristiche per costruirne uno "ragionevolmente semplice"
- Un algoritmo: ID3 [Quinlan]

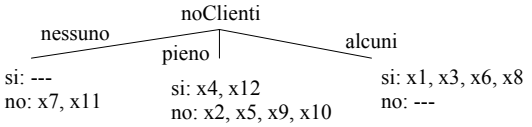
  if non ci sono esempi then termina (o classifica a maggioranza)
  if tutti gli esempi hanno la stessa classificazione
  then etichetta il nodo con la classificazione
  else if nessun attributo rimasto then termina con problema
  seleziona attributo ai che discrimina meglio
  dividi gli esempi in base ai valori di ai
  costruisci un nodo etichettato con ai e con un ramo in uscita per ogni valore vj di ai
  richiama ricorsivamente per ogni vj con gli esempi corrispondenti

- Attributo ideale ai da selezionare ad ogni passo
  - discrimina completamente tra gli esempi, ossia
    - dati valori v1, …, vk
    - per ogni vj, tutti gli esempi con lo stesso valore vj hanno la stessa classificazione
- Attributo da scegliere
  - quello che discrimina meglio tra gli esempi
    - per la maggior parte dei vj, gli esempi con lo stesso valore di vj hanno quasi tutti la stessa classificazione
- Una misura della bontà di un attributo: entropia
  - studiata in information theory [Shannon, Weaver]
  - misura la quantità di informazione fornita da ognuno degli attributi
    - per attr ai, e ogni valore vj costruiamo distribuzione di probabilità delle classi P(C1), …, P(Cm)
    - I(ai) informazione fornita da ai definita nel modo seguente

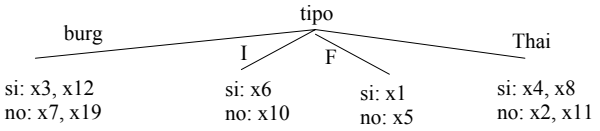
      $I(ai) = \sum I(vj)$

      $I(vj) = \sum -P(Ck) \log_2 P(Ck) \qquad 1<=k<=m$
    - ha valore minimo (0) quando una P(Ck) vale 1, ossia quando ogni valore dell'attributo discrimina completamente

- Esempio
  - attributo noClienti: per due valori discrimina completamente

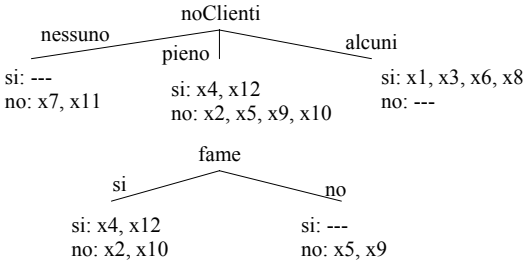


  - attributo tipo: discrimina male per tutti i valori



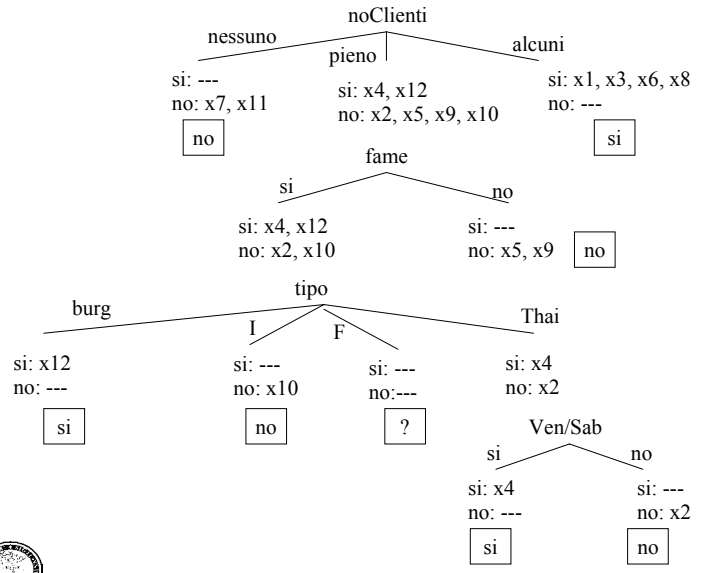
  - Tra i due noClienti è la scelta migliore
  - in generale tra tutti è quello con entropia più bassa

  - Algoritmo procede ricorsivamente considerando il valore "pieno" di noClienti e considerando gli esempi per quel valore
    - si analizzano gli altri attributi
    - si seleziona quello che discrimina meglio
      nel caso "fame": per uno dei due valori si ha classificazione completa



  - Analogamente si procede ricorsivamente su ramo "si"

– Alla fine albero risultante (diverso da quello visto prima)



Valutazione delle prestazioni

- Conoscenza appresa influenzata dagli esempi
  - potrebbe classificare in modo sbagliato nuove istanze
  - esempi devono essere un buon campione della popolazione
- Una strategia di verifica della conoscenza appresa da esempi
  - collezionare un largo insieme di esempi che sia un campione significativo
  - dividere l'insieme in due parti
    - training set T
    - test set S

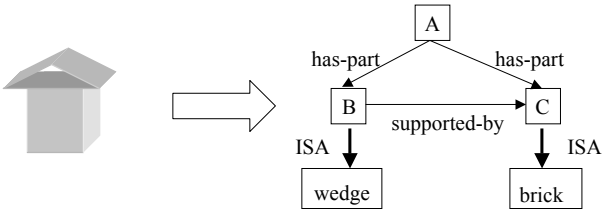
    in modo uniforme (secondo la stessa distribuzione degli esempi)
  - effettuare l'apprendimento usando T
  - valutare la percentuale di corretta classificazione usando S
  - ripetere lo stesso processo più volte con diversa divisione tra T e S e per diverse dimensioni del training set

Learning di descrizioni strutturate

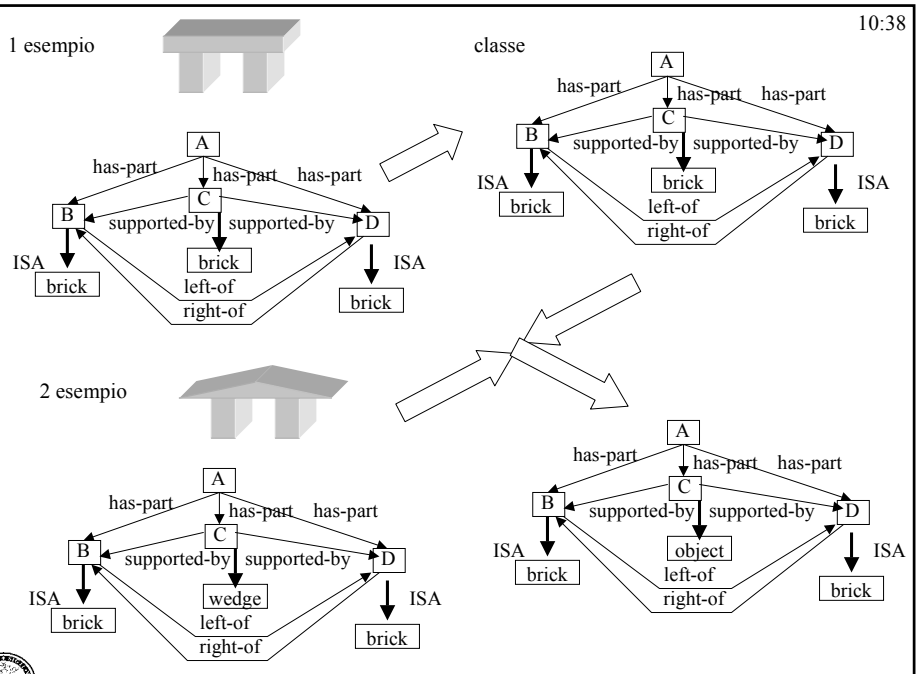
- In alberi decisionali le classi non sono rappresentate in modo esplicito
- Altri approcci: rappresentazione strutturata delle classi usando un linguaggio di rappresentazione della conoscenza
  - reti semantiche
  - logica
  - …
- Vedremo due approcci
  - approccio di Winston basato su reti semantiche
  - version space [Mitchell]: apprendimento di descrizioni basate su formule logiche

L'approccio di Winston

- Concetti rappresentati mediante reti semantiche
  - nodi: entità
  - archi: relazioni
    - ISA
    - proprietà (e quindi relazioni tra entità)
- Esempio
  - descrizione di oggetti architettonici (stilizzati)
    - casa



- Istanze (esempi): descritti mediante reti semantiche
- Learning: generalizzare da reti semantiche corrispondenti agli esempi per ottenere quelle delle classi
- Metodo incrementale
  - si parte da un esempio
    - rete della classe coincide con quella dell'esempio
  - dato altro esempio
    - si generalizza rete della classe per coprire nuovo esempio (se non lo copre già)
  - vari tipi di generalizzazioni possibili
    - sfruttando ISA
    - eliminando proprietà
    - generalizzando proprietà
- Oss: in realtà si usa linguaggio testuale che descrive le reti semantiche
- Esempio: apprendimento della descrizione della classe arco



Apprendimento di formule logiche

- Descrizione classi
  - formula logica F i cui predicati sono attributi delle classi
  - F consistente con esempi
    - F vera su esempi
    - F falsa su contro-esempi
  - *estensione* di F: insieme delle istanze per cui F è vera
- Esempi
  - formule logiche e classe di appartenenenza

    <Fi, classe> (spesso Fi ground)
- Esempio
  - classe attesa al ristorante

    $\forall x\ attesa(X) \leftrightarrow noClienti(X,alcuni) \vee$
    $(noClienti(X,molti) \wedge \neg fame(X) \wedge tipo(X,f)) \vee$
    $(noClienti(X,molti) \wedge \neg fame(X) \wedge tipo(X,thai) \wedge ven\text{-}sab(X)) \vee$
    $(noClienti(X,molti) \wedge \neg fame(X) \wedge tipo(X,burger))$
  - istanza

    $alternative(x1) \wedge \neg bar(x1) \wedge \neg ven\text{-}sab(x1) \wedge fame(x1)$ …. CLASSIF $attesa(x1)$

- Hypothesis space
  - spazio delle possibili formule logiche per definire ogni classe C
- Hypothesis
  - una particolare definizione H per C
- Estensione di una ipotesi
  - insieme di istanze di H
- Falso negativo per H

  esempio x è un falso negativo per H se e solo se
  - x è un esempio per C
  - x non è istanza di H (H prevede che sia negativo)
- Falso positivo per H

  esempio x è un falso positivo per H se e solo se
  - x non è un esempio per C
  - x è istanza di H (H prevede che sia positivo)
- Falso negativo e falso positivo indicano che H è sbagliata e deve essere modificata

- Generalizzazione e specializzazione: due operatori per modificare formule

  date ipotesi H e H' con estensioni e(H) e e(H')
  - H è più specifica di H' (H è una specializzazione di H')
    se e solo se e(H) $\subseteq$ e(H') (alt. $\vdash$ H $\rightarrow$ H')
  - vv. H' è più generale di H

  esempio
  - cubo(x) $\wedge$ rosso(x) è più specifica di cubo(x)
  - cubo(x) è più specifica di cubo(x) $\vee$ piramide(x)

  ATTN: possono anche essere definiti rispetto a teoria T
- Operatori di generalizzazione
  - dropping condition: da cubo(x) $\wedge$ rosso(x) a: cubo(x)
  - sostituzione con predicati più generali: da cubo(x) a figGeo(x)
    (se cubo(x) $\rightarrow$ figGeo(x))
- Operatori di specializzazione (duali)
  - aggiunta di condizioni
  - sostituzione con predicati più specifici

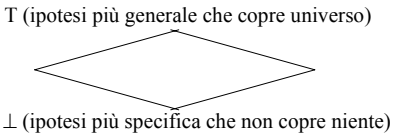
- Apprendimento incrementale (simile a Winston)

  parto da un esempio x1, con H $\equiv$ x1
  while ci sono esempi do
  seleziona esempio xi
  if H consistente con xi
  then H invariata
  else if xi è un falso negativo per H (H dice neg ma xi è esempio)
  then H := generalizzazione di H consistente con xi
  if xi è un falso positivo per H (H dice pos ma xi non è esempio)
  then H := specializzazione di H consistente con xi
- Problemi
  - Hypothesis space enorme
  - molti modi per generalizzare e specializzare
- Una soluzione: version space
  - introdurre un bias
  - gestire la ricerca con least-committment

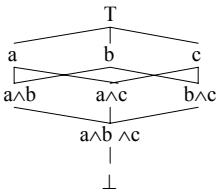
Version space

- Apprendere formule logiche da esempi ma:
  - non mantenere singola ipotesi che viene modificata
  - mantenere in forma compatta l'intero spazio di ipotesi consistente
- bias: ipotesi in forma congiuntiva
  - consente di strutturare Hypothesis space e di rappresentare in forma compatta insiemi di ipotesi

- Hypothesis space: reticolo

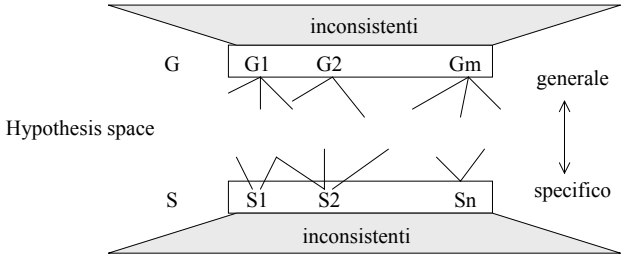


- Esempio: predicati a, b, c



- Algoritmo di apprendimento
  - in ogni istante una parte del reticolo contiene ipotesi consistenti con esempio
    - all'inizio intero reticolo
  - parte del reticolo caratterizzabile in modo compatto con boundary set
- Boundary set:
  - insieme G delle ipotesi più generali consistenti con esempi (ossia delle ipotesi H / non esiste H' più generale di H consistente con esempi)
  - insieme S delle ipotesi più specifiche consistenti con esempi (ossia delle ipotesi H / non esiste H' più specifico di H consistente con esempi)
- rappresentazione compatta
  - ogni nodo tra quelli in G e S nel reticolo è una ipotesi
- Boundary set iniziale:

  G= {T } S= { ⊥ }
- Ad ogni passo
  - modificare boundary set in accordo con gli esempi
    - specializzare ipotesi in G quando si hanno contro-esempi (per escluderli)
    - generalizzare ipotesi in S quando si hanno esempi (per coprirli)



- Algoritmo
  - se esempio xi è un falso positivo per Si allora eliminare Si da S (vuol dire che Si è troppo generale, ma tutte sue specializzazioni sono inconsistenti)
  - se esempio xi è un falso positivo per Gj, allora sostituire Gj con le sue specializzazioni più generali consistenti con xi
  - se esempio xi è un falso negativo per Si allora sostituire Si con le sue generalizzazioni più specifiche consistenti con xi
  - se esempio xi è un falso negativo per Gj, allora eliminare Gj da G (vuol dire che Gi è troppo specifico, ma tutte sue generalizzazioni sono inconsistenti)

- Esempio
  - apprendere concetto di auto economica
  - predicati:
    - origine: Jap, I, F, D, USA
    - colore: b, r, g, …
    - tipo: berlina, coupè, ..
    - decade: 70, 80, 90
    - marca: Fiat, Alfa, Mercedes, BMW, Renault, Honda, Ford, …
  - inizio
    - G= {T } S= { ⊥ }
  - primo esempio (positivo)
    - origine(Jap), colore(b), marca(Honda), tipo(berlina), decade(70)
    - falso negativo per ⊥ che deve essere generalizzato
      G= {T }
      S= {origine(Jap)∧ colore(b) ∧ marca(Honda) ∧ tipo(berlina) ∧ decade(70) }
  - secondo esempio (negativo, ossia contro-esempio)
    - origine(USA), colore(r), marca(Chevrolet), tipo(coupè), decade(90)
    - falso positivo per T che deve essere specializzato e ho cinque modi possibili (non confrontabili) di specializzarlo: uno per ogni predicato

G= {origine(Jap), colore(b), marca(Honda), tipo(berlina), decade(70) }

S= {origine(Jap)∧ colore(b) ∧ marca(Honda) ∧ tipo(berlina) ∧ decade(70) }

- terzo esempio (positivo)
  - origine(Jap), colore(b), marca(Toyota), tipo(berlina), decade(80)
  - si deve rimuovere da G ogni ipotesi inconsistente con esempio

    G= {origine(Jap), colore(b), tipo(berlina) }
  - falso negativo per S che deve essere generalizzato

    S= {origine(Jap)∧ colore(b) ∧ tipo(berlina) }
- quarto esempio (negativo)
  - origine(I), colore(b), marca(Alfa), tipo(coupè), decade(80)
  - falso positivo per colore(b) in G che deve essere sostituito da sue specializzazioni più generali che sono
    - colore(b) ∧ tipo(berlina) colore(b) ∧ origine(Jap)

    entrambe più specifiche di ipotesi già in G

    G= {origine(Jap), tipo(berlina) }
  - S= {origine(Jap)∧ colore(b) ∧ tipo(berlina) }
- quinto esempio (positivo)
  - origine(Jap), colore(r), marca(Honda), tipo(berlina), decade(80)
  - falso negativo per S che deve essere generalizzato

    S= {origine(Jap) ∧ tipo(berlina) }
  - G= {origine(Jap), tipo(berlina) }

- Osservazioni
  - importanza del bias
  - forma di least-commitment nel senso che si specializza (generalizza) sempre il meno possibile
  - collassamento del version space in caso di errori(rumore) negli esempi
  - se si dispone di molti esempi si può arrivare ad una singola formula
  - conoscenza a-priori potrebbe essere utile nel processo di specializzazione/generalizzazione (vedremo)

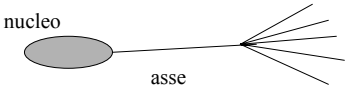
    es: origine(Jap) → origine(asia)

    origine(Corea) → origine(asia)

    potrei usare questa conoscenza per generalizzare/specializzare

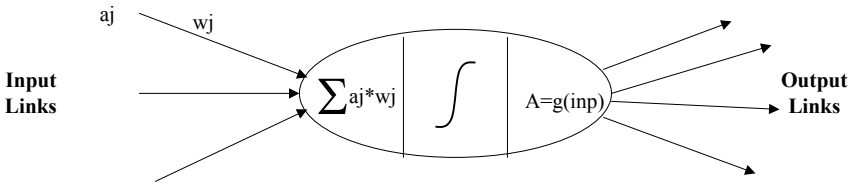
Approcci sub-simbolici (connessionistici)

- Approcci ad AI visti fino ad ora sono simbolici
  - rappresentazione simbolica della conoscenza
  - tecniche simboliche di ragionamento
- Lunga tradizione in AI (e da prima in cibernetica) per approcci non simbolici alla computazione (da anni '40)
  - reti neurali
  - reti connessionistiche
  - algoritmi genetici
  - …
- Diverso approccio alla computazione
- Modelli matematici del funzionamento del cervello
- Neuroni



Sinapsi (collegamenti con altri neuroni)

- Neurone come unità computazionale



- Le reti neurali

Conoscenza e learning (learning con prior knowledge)

- Approcci visti fino ad ora: apprendimento da esempi
  - nessuna conoscenza a priori richiesta
  - K.B. appresa dipende solo da esempi (e da bias)
  - K.B. appresa a volte di difficile comprensione anche per esperti di un settore (esperimenti in sistemi esperti)
  - processo di apprendimento lento (servono molti esempi)
- Altro gruppo di approcci da metà anni '80
  - sfruttare K.B. a -priori nel processo di apprendimento (conoscenza di background)
  - K.B. consente di apprendere di più e in modo più efficace (più velocemente)
- Vedremo due approcci
  - Knowledge-based induction (e inductive logic programming)
  - Explanation Based Learning (Explanation Based Generalization)

Knowledge-based induction

- Induzione da esempi
  - dati esempi e1, e2, …, en
  - trovare una ipotesi H tale per cui $H \vdash ei$ (per ogni ei)
  - ad es.
    - esempi di classificazione (ei descrizione esempio, c classe)
      $e1 \rightarrow c$, $e2 \rightarrow c$, …., $eN \rightarrow c$
    - trovare H tale per cui
      $H \rightarrow (ei \rightarrow c)$ ossia $H \wedge ei \rightarrow c$ (per ogni i)
  - esempio
    - dati esempi
      giallo(e1) → limone(e1)
      …
      giallo(eN) → limone(eN)
    - si può generare ipotesi
      $\forall X$ giallo(X) → limone(X)
- Induzione può essere effettuata anche con conoscenza a-priori

- Induzione con K.B.
  - effettuare induzione con conoscenza di background per generare le ipotesi
  - dati
    - K: conoscenza di background
    - e. osservazione (esempio)
  - trovare H tale per cui

    $K \cup H \vdash e$ (ossia $K \wedge H \vdash e$)
- Osservazione: confronto tra abduzione e induzione
  - due forme di inferenza (non truth-preserving) che generano spiegazioni
  - abduzione: spiegazione causale delle osservazioni
  - induzione: generalizzazione
  - esempio
    - data teoria su elettricità e osservazione che una lampadina si accende quando si gira interruttore
      - abduzione: genera spiegazione causale
      - induzione: spiega osservazione con il fatto che in esempi passati è sempre stato così

- Esempio
  - data K.B. $\forall X$ quadrato(X) $\rightarrow$ poligono(X)

    $\forall X$ rettangolo(X) $\rightarrow$ poligono(X)
  - data osservazione quadrato(a) $\rightarrow$ rosso(a)
  - spiegazioni possibili (induttive)
    - $\forall X$ quadrato(X) $\rightarrow$ rosso(X)
    - $\forall X$poligono(X) $\rightarrow$ rosso(X)
- Problema. Avere operatori per generare ipotesi induttive
- Inverse resolution [Muggleton, 92]
  - invertire la regola di risoluzione fornisce insieme di operatori che permettono di derivare tutte le possibili ipotesi induttive
- Risoluzione

  date due clausole C1 e C2 deriva una clausola C
- Inverse resolution

  applicare la regola in modo inverso (vari modi possibili):
  - dato C trovare C1 e C2
  - dati C e C1, trovare C2 (o, dati C e C2, trovare C1)

Inverse resolution: alcuni operatori

- Truncation
  - dato cane(X) $\wedge$ domestico(X) $\rightarrow$ haNome(X)
  - genera cane(X) $\rightarrow$ haNome(X)
- Assorbimento (absorption)
  - dati
    - cane(X) $\rightarrow$ animale(X)
    - cane(X) $\wedge$ domestico(X) $\rightarrow$ haNome(X)
  - genera
    - animale(X) $\wedge$ domestico(X) $\rightarrow$ haNome(X)
  - ossia generalizza a p1 le proprietà di p
- Molti modi possibili di applicare operatori: spazio di ricerca di ipotesi induttive
- Inductive Logic Programming
  - regole di risoluzione inversa su programmi logici
    - applicazione di inverse resolution
    - apprendimento di programmi da esempi (e background di altri programmi)

Explanation Based Learning

- Apprendimento da esempi richiede molti esempi (e contro-esempi)
- Esseri umani sono in molte situazioni in grado di imparare da un singolo esempio

  in quanto

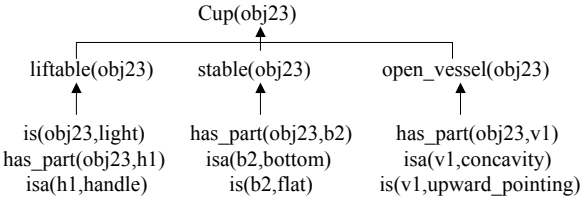
  capacità di usare la conoscenza di background
- EBL: approccio per emulare queste capacità
- intuitivamente
  - data una K.B. e un esempio (e.g., la classificazione di una istanza)
  - costruire una spiegazione (dimostrazione) dell'esempio (e.g. una dimostrazione che è istanza di una classe)
  - generalizzare tale dimostrazione (e.g., per trovare una regola che fornisce una descrizione della classe)

  il risultato di tale generalizzazione è nuova conoscenza appresa dal singolo esempio grazie alla background knowledge

E.B.L.: definizione

- [Michell et al. 86][DeJong, Mooney, 86]
- Dati
  - una base di conoscenza (teoria del dominio)
  - un goal, ossia un predicato da apprendere
  - un singolo esempio (del predicato)
  - un criterio di operazionalità che specifica quali predicati possono occorrere all'interno della conoscenza appresa
- Generare
  - una generalizzazione dell'esempio che
    - è una descrizione sufficiente del predicato da apprendere
    - soddisfa il criterio di operazionalità
- Algoritmo
  - generare una dimostrazione dell'esempio a partire dalla teoria del dominio
  - generalizzare la dimostrazione

- Esempio [Mitchell 86]
  - apprendere la descrizione di cup (tazza) per cui si ha
    $cup(X) \leftrightarrow liftable(X) \wedge stable(X) \wedge open\_vessel(X)$
  - Teoria del dominio
    $is(X,light) \wedge has\_part(X,Y) \wedge isa(Y,handle) \rightarrow liftable(X)$
    $has\_part(X,Y) \wedge isa(Y,bottom) \wedge is(Y,flat) \rightarrow stable(X)$
    $has\_part(X,Y) \wedge isa(Y,concavity) \wedge is(Y,upward\_pointing) \rightarrow open\_vessel(X)$
  - criterio di operazionalità
    descrizione in termini di caratteristiche fisiche
  - esempio
    $owner(obj23,ralph) \wedge has\_part(obj23,c1) \wedge is(obj23,light) \wedge color(obj23,brown)$
  - dimostrazione



– Generalizzazione: rimpiazzare costanti con variabili

is(W,light)	has_part(W,K)	has_part(W,V)
has_part(W,Z)	isa(K,bottom)	isa(V,concavity)
isa(Z,handle)	is(K,flat)	is(V,upward_pointing)

descrizione operazionale del concetto di tazza

– Oss: non sempre generalizzare è così facile (cambiare costanti in variabili)

$$\text{cup}(W) \leftrightarrow \text{is}(W,\text{light}) \wedge \text{has\_part}(W,Z) \wedge \text{isa}(Z,\text{handle}) \wedge \text{has\_part}(W,K)$$
$$\text{isa}(K,\text{bottom}) \;\; \text{is}(K,\text{flat}) \wedge \;\; \text{has\_part}(W,V) \;\; \text{isa}(V,\text{concavity})$$
$$\text{is}(V,\text{upward\_pointing})$$

- Osservazioni
  - EBL dipendente dalla teoria del dominio
    - teoria deve essere corretta e completa
  - Esempio non è strettamente necessario