Introduzione al Corso di

Sistemi di Elaborazione dell'Informazione

Alessandro Sperduti

Dipartimento di Matematica Pura ed Applicata

Università di Padova

Riferimenti Bibliografici:

S. Russell & P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 1995

K. Marriott & P. Stuckey, Programming with constraints, MIT Press, 1998

Tom Mitchell, Machine Learning, McGraw Hill, 1997

Scopo del Corso

Il corso si propone di presentare alcuni degli approcci principali, all'interno della Intelligenza Artificiale, per la soluzione di problemi difficili.

Un problema è considerato difficile se:

- può essere formalizzato in modo preciso ma non si conosce un algoritmo che lo risolve in tempo polinomiale;
- NON può essere formalizzato in modo preciso (mancanza dati, rumore, incertezza,...)

In particolare, verranno presentate tecniche di

- Ricerca in uno Spazio di Soluzioni;
- Risoluzione di Problemi di Vincoli;
- Apprendimento Automatico;

Contenuti del Corso (1)

Ricerca in uno Spazio di Soluzioni

- Formulazione di un problema di ricerca;
- Esempi di formulazioni di problemi di ricerca;
- Ricerca delle soluzioni di un problema;
- Alcune strategie di ricerca di una soluzione (breadth-first, depth-first,...)
- Funzioni Euristiche;
- Ricerca Locale

Contenuti del Corso (2)

Risoluzione di Problemi di Vincoli

- Introduzione ai Problemi di Vincoli;
- Alcuni risolutori di vincoli
  - Gauss, Unificazione, Booleano;
- Semplificazione e proiezione (algoritmo di Fourier);
- Ottimizzazione;
- Vincoli su Domini Finiti
  - consistenza sui nodi, archi e limiti;
  - ricerca con backtracking;
  - branch and bound
- Vincoli con Preferenze;

Contenuti del Corso (3)

Apprendimento Automatico

- Cenni all'apprendimento di concetti e alla Teoria Computazionale dell'Apprendimento;
- Apprendimento di Alberi di Decisione;
  - Algoritmo ID3;
  - Entropia;
  - Casi Speciali (attributi continui e/o mancanti, costi, ...);
  - Pruning
- Reti Neurali
  - Perceptron
  - Back-Propagation
- Apprendimento con Rinforzo (Q-Learning);
- Altri algoritmi (k-nn, clustering gerarchico,...)

Problemi Difficili

Problema $\rightarrow$ Descrizione Formale $\rightarrow$ Rappresentazione + Algoritmo/i $\rightarrow$ Soluzione

Un problema è semplice se è conosciuto un algoritmo che impiega tempo polinomiale per trovare una soluzione

Es.: Ordinamento di una sequenza di numeri (23, 14, 1, 267, 17, ...)

PROBLEMA DIFFICILE:

- NON è conosciuto un algoritmo polinomiale per trovare una soluzione
  - $\rightarrow$ Ricerca in uno Spazio di Soluzioni
  - $\rightarrow$ Risoluzione di Problemi di Vincoli
- NON è facile (o possibile) formalizzare precisamente il problema
  - $\rightarrow$ Apprendimento Automatico

Risoluzione di Problemi Difficili: Ricerca nello Spazio delle Soluzioni

Molti problemi difficili si possono descrivere tramite Modelli a Stato:

- Spazio degli Stati (discreto e finito);
- Spazio delle Azioni (discreto e finito);
- funzione di transizione: dato uno stato ed una azione restituisce un nuovo stato;
- stato iniziale;
- stati finali (goals);
- (ogni azione può avere un costo associato)

Una soluzione è costituita da una sequenza di azioni che mappa lo stato iniziale in uno degli stati finali [... ed è ottima se minimizza una data funzione dei costi associati alle azioni usate]

Esempio di Spazio di Ricerca (o delle Soluzioni)

Spazio degli Stati:

possibili configurazioni

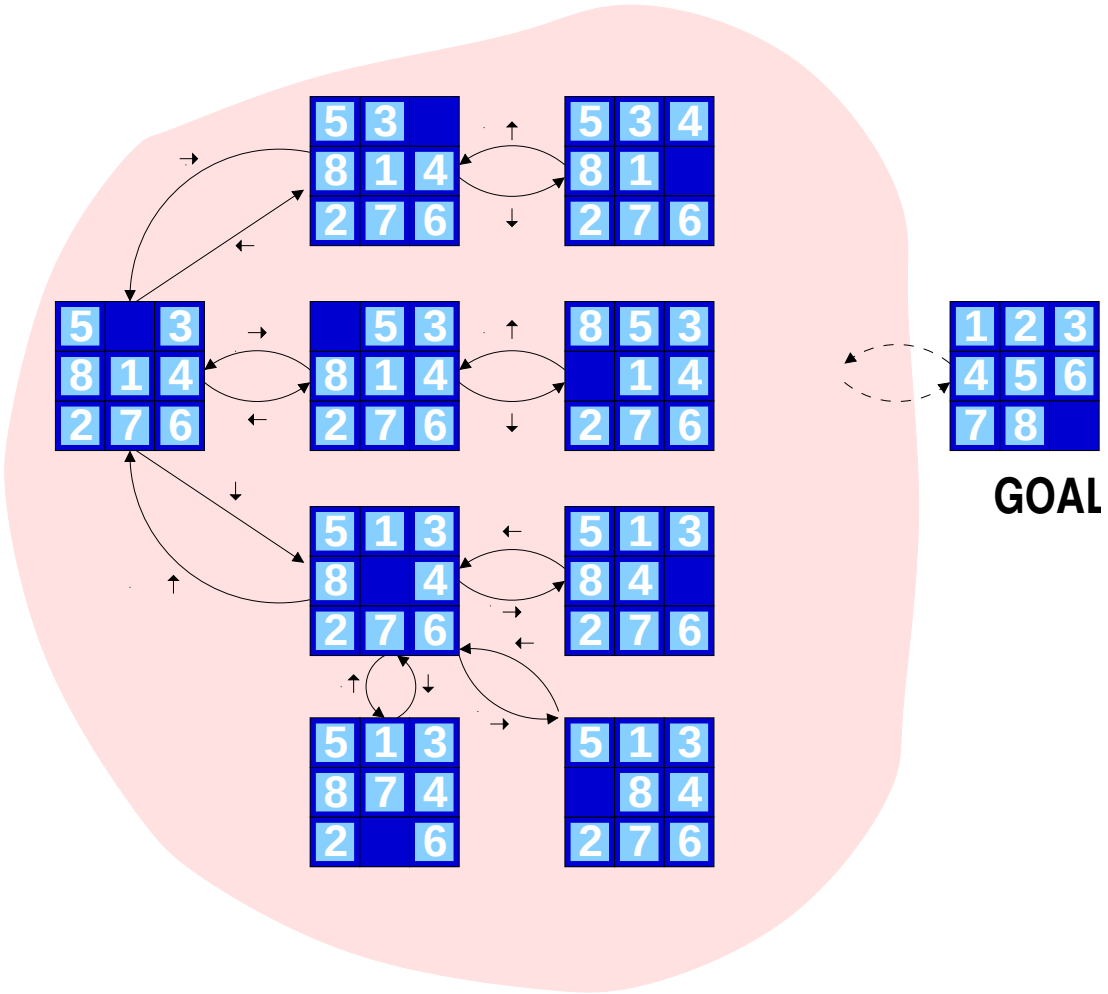
dei tasselli

Spazio delle Azioni (mosse):

possibili "spostamenti"

dello spazio vuoto

($\leftarrow, \rightarrow, \uparrow \cdot \downarrow$)



Strategie di Ricerca

Dato un problema descritto come un Modello a Stati esistono varie strategie di ricerca:

- Generali (non utilizzano alcuna conoscenza a priori sul problema da risolvere);
- Specifiche (utilizzano conoscenza a priori sul problema da risolvere);

In ogni caso, queste vengono valutate in base ai seguenti criteri:

- Completezza: si trova usa soluzione se questa esiste ?
- Tempo di esecuzione della ricerca;
- Spazio di memoria utilizzato durante la ricerca;
- [Ottimalità: si trova la soluzione a costo minimo ?]

Argomenti coperti da $\rightarrow$ Ricerca in uno Spazio di Soluzioni

Problemi di Vincoli

Alcuni problemi difficili si descrivono bene come Problemi di Vincoli:

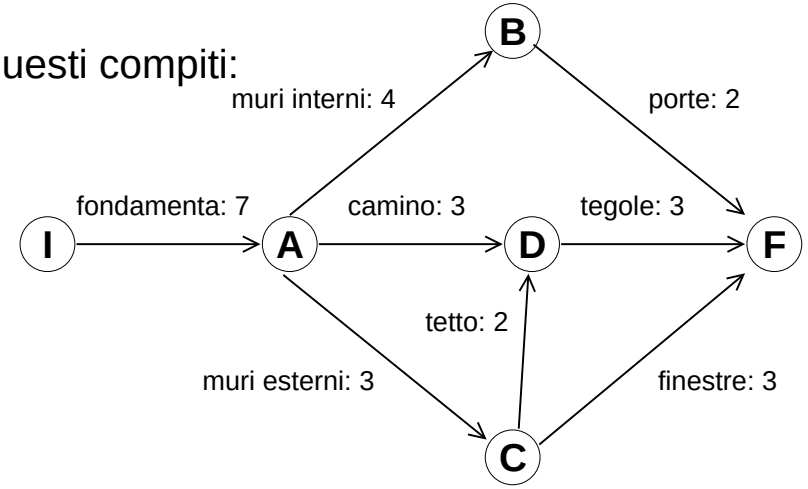
- Insieme di Variabili (discreto e finito);
- Domini per le variabili: i valori che ogni variabile può assumere;
- Vincoli fra variabili: quali "configurazioni" di valori da assegnare alle variabili sono ammessi;

Una soluzione è costituita da un assegnamento di valori a tutte le variabili tali che tutti i vincoli sono soddisfatti

Argomenti coperti da $\rightarrow$ Risoluzione di Problemi di Vincoli

Esempio di Problema di Vincoli: costruzione di una casa

- Compiti di base (con durata): fondamenta, muri esterni, muri interni, camino, tetto, porte, finestre, tegole;
- Restrizione sull'ordine in cui svolgere questi compiti:



Problema di vincoli corrispondente:

Inizio: $T_I \geq 0$; fondamenta: $T_A \geq T_I + 7$; muri interni: $T_B \geq T_A + 4$;

muri esterni: $T_C \geq T_A + 3$; camino: $T_D \geq T_A + 3$; tetto: $T_D \geq T_C + 2$;

porte: $T_F \geq T_B + 2$; tegole: $T_F \geq T_D + 3$; finestre: $T_F \geq T_C + 3$;

una soluzione: $\{T_I := 0, T_A := 7, T_B := 11, T_C := 10, T_D := 12, T_F := 15\}$

Come Risolvere i Problemi di Vincoli

- in generale sono NP completi;
- alcune sottoclassi possono essere risolte in tempo polinomiale;
- ...altrimenti si può risolverli in modo approssimato (a volte non si sa trovare una soluzione anche se c'è);
- oppure usare algoritmi di ricerca:
  - euristiche, per dirigersi verso una soluzione con numero minimo di passi
  - semplificazione + proiezione, per scartare dei pezzi dell'albero di ricerca

Quando è Necessario l'Apprendimento (Automatico) ?

Quando il sistema deve...

- adattarsi all'ambiente in cui opera (anche personalizzazione automatica);
- migliorare le sue prestazioni rispetto ad un particolare compito;
- scoprire regolarità e nuova informazione (conoscenza) a partire da dati empirici;
- acquisire nuove capacità computazionali.

Perchè non usare un approccio algoritmico tradizionale ?

- impossibile formalizzare esattamente il problema (e quindi dare una soluzione algoritmica);
- presenza di rumore e/o incertezza ;
- complessità alta nel formulare una soluzione: non si può fare a mano;
- mancanza di conoscenza "compilata" rispetto al problema da risolvere;

Ruolo dei Dati

Tipicamente...

- si hanno a disposizione (molti ?) dati
  - ottenuti una volta per tutte;
  - acquisibili interagendo direttamente con l'ambiente;
- (forse) conoscenza del dominio applicativo, ma
  - incompleta;
  - imprecisa (rumore, ambiguità, incertezza, errori, ...);

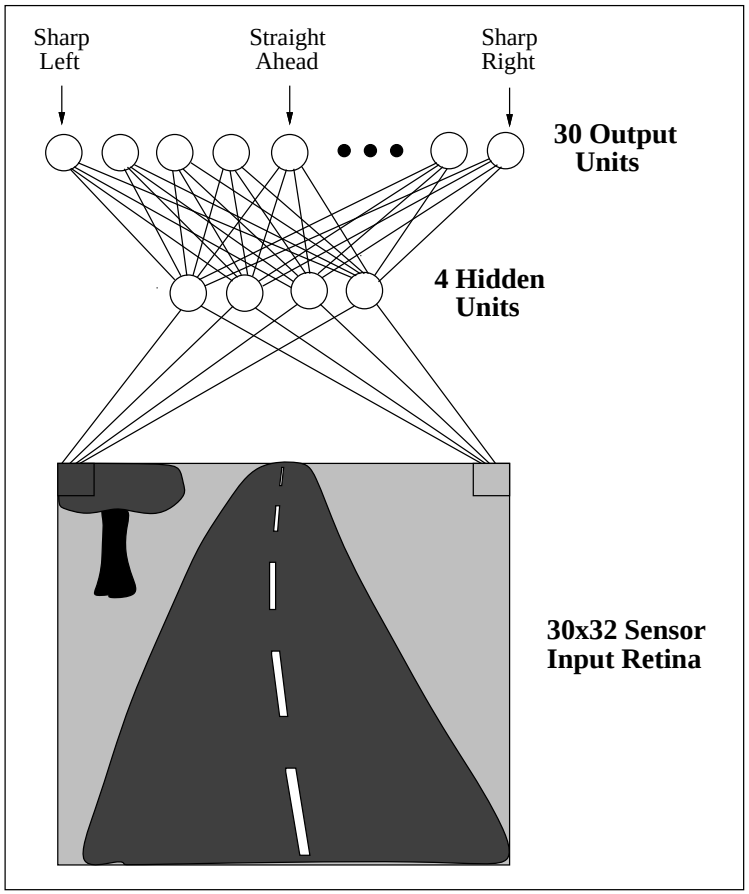
Desiderio: usare i dati per

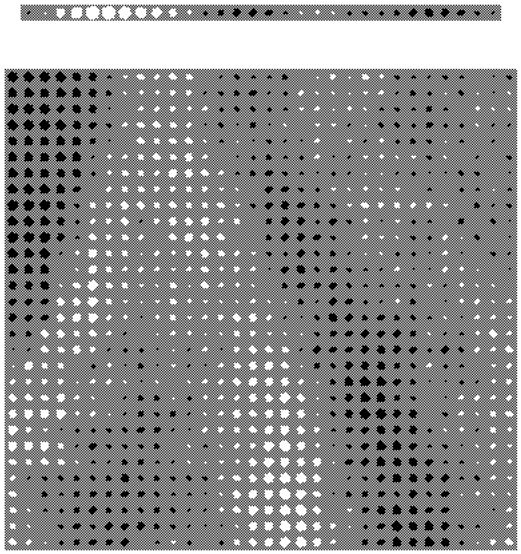
- ottenere nuova conoscenza;
- raffinare la conoscenza di cui si dispone;
- correggere la conoscenza di cui si dispone;

Es. - Riconoscimento di Cifre Manoscritte

- impossibile formalizzare esattamente il problema: disponibili solo esempi;
- possibile presenza di rumore e dati ambigui;

Es. - Guidare una Automobile





Es. - Estrarre Conoscenza Medica dai Dati

Patient103 time=1 → *Patient103* time=2 ··· → *Patient103* time=n

Age: 23	Age: 23	Age: 23
FirstPregnancy: no	FirstPregnancy: no	FirstPregnancy: no
Anemia: no	Anemia: no	Anemia: no
Diabetes: no	Diabetes: YES	Diabetes: no
PreviousPrematureBirth: no	PreviousPrematureBirth: no	PreviousPrematureBirth: no
Ultrasound: ?	Ultrasound: abnormal	Ultrasound: ?
Elective C−Section: ?	Elective C−Section: no	Elective C−Section: no
Emergency C−Section: ?	Emergency C−Section: ?	Emergency C−Section: **Yes**
...	...	...

Linee di Ricerca all'interno dell' Apprendimento Automatico

- induzione di regole/alberi di decisione,
- algoritmi connessionisti (reti neurali),
- "clustering" & "discovery",
- apprendimento basato sulle istanze
- apprendimento Bayesiano,
- apprendimento basato sulla spiegazione,
- apprendimento con rinforzo,
- apprendimento induttivo guidato dalla conoscenza,
- ragionamento per analogia & basato sui casi,
- algoritmi genetici,
- programmazione logica induttiva, . . .

Principali Paradigmi di Apprendimento

Apprendimento Supervisionato:

- dato in insieme di esempi pre-classificati, $Tr = \{(x^{(i)}, f(x^{(i)}))\}$, apprendere una descrizione generale che incapsula l'informazione contenuta negli esempi (regole valide su tutto il dominio di ingresso)
- tale descrizione deve poter essere usata in modo predittivo (dato un nuovo ingresso $\tilde{x}$ predire l'output associato $f(\tilde{x})$)
- si assume che un esperto (o maestro) ci fornisca la supervisone (cioè i valori della $f()$ per le istanze x dell'insieme di apprendimento)

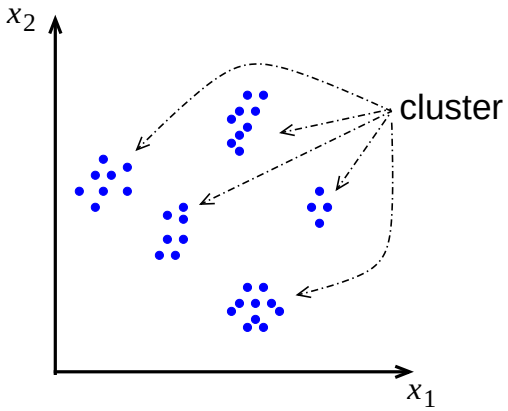
Esempio di applicazione: classificazione di caratteri manoscritti

Principali Paradigmi di Apprendimento

Apprendimento Non-supervisionato:

- dato in insieme di esempi $Tr = \{x^{(i)}\}$, estrarre regolarità e/o pattern (valide(i) su tutto il dominio di ingresso)
- non esiste nessun esperto (o maestro) che ci fornisca un aiuto

  – Clustering



  – Scoperta di Regole (Discovery)

Esempio di applicazione: data mining su database strutturati

Principali Paradigmi di Apprendimento

Apprendimento con Rinforzo:

- Sono dati:
  - agente (intelligente ?), che può
    - trovarsi in uno stato s, ed
    - eseguire una azione a (all'interno delle azioni possibili nello stato corrente)
  - ed opera in un ambiente e, che applicando una azione a nello stato s restituisce
    - lo stato successivo, e
    - una ricompensa r, che può essere positiva (+), negativa (-), o neutra (0).
- Scopo dell'agente è quello di massimizzare una funzione delle ricompense (es. ricompensa scontata: $\sum_{t=0}^{\infty} \gamma^t r_{t+1}$ dove $0 \leq \gamma < 1$)

Esempio di applicazione: navigare sul Web alla ricerca di informazione focalizzata

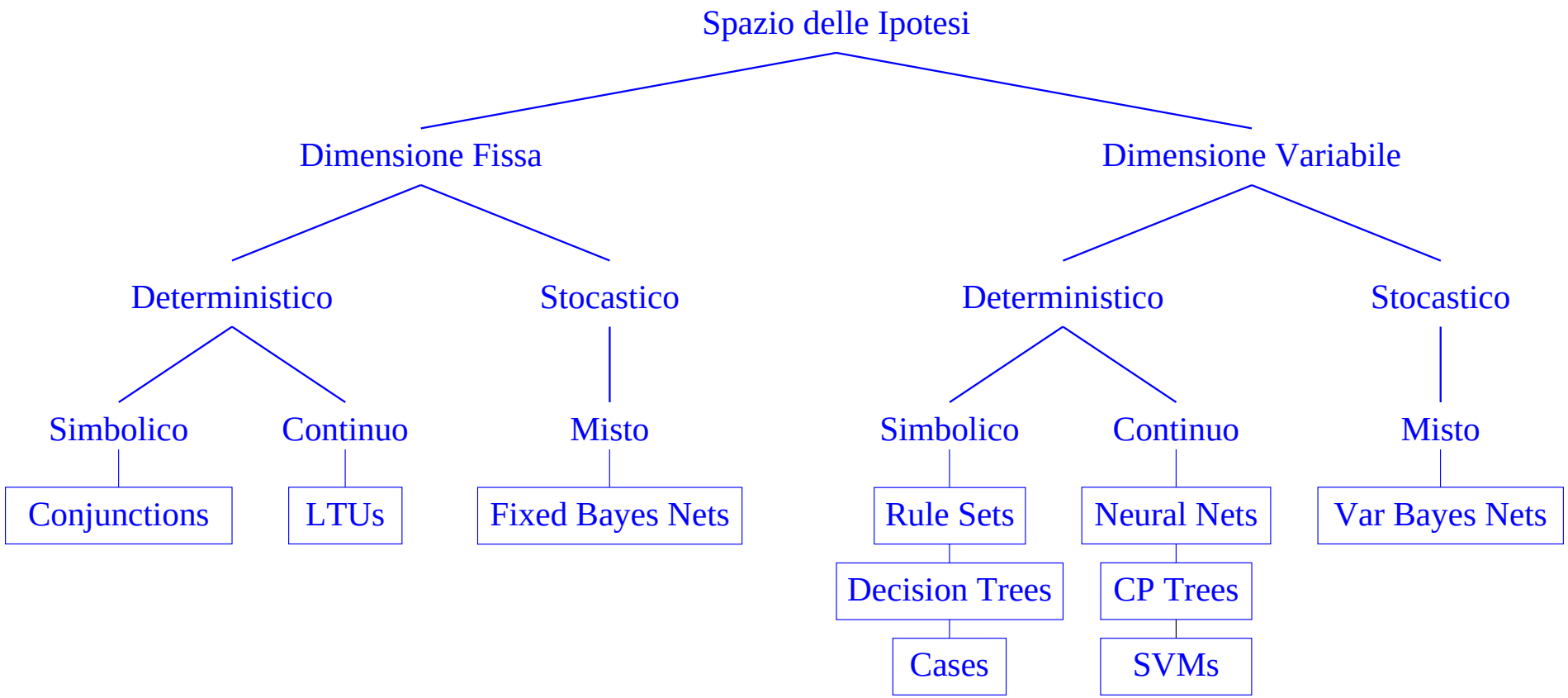
Ingredienti Fondamentali

- Dati di Allenamento
- Spazio delle Ipotesi, $\mathcal{H}$
  - costituisce l'insieme delle funzioni che possono essere realizzate dal sistema di apprendimento;
  - si assume che la funzione da apprendere f possa essere rappresentata da una ipotesi $h \in \mathcal{H}$... (selezione di h attraverso i dati di apprendimento)
  - o che almeno una ipotesi $h \in \mathcal{H}$ sia simile a f (approssimazione);
- Algoritmo di Ricerca nello Spazio delle Ipotesi, alg. di apprendimento

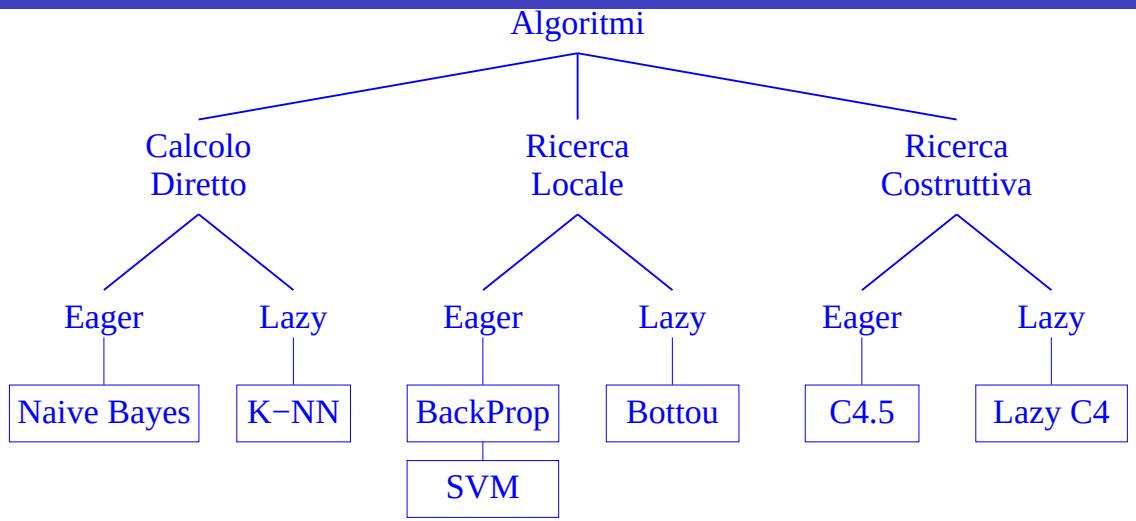
ATTENZIONE: $\mathcal{H}$ non può coincidere con l'insieme di tutte le funzioni possibili e la ricerca essere esaustiva $\rightarrow$ Apprendimento è inutile!!!

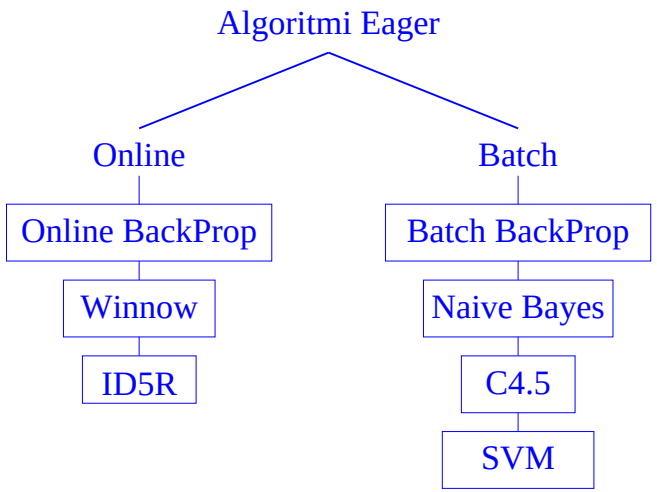
Si parla di **Bias Induttivo**: sulla rappresentazione ($\mathcal{H}$) e/o sulla ricerca (alg. di apprendimento)

Tassonomia (non completa) dello Spazio delle Ipotesi



Tassonomia (non completa) degli Algoritmi di Apprendimento





Vediamo una Istanza di Apprendimento

Apprendimento $\rightarrow$ migliorare con l'uso dell'esperienza nell'eseguire un compito

Elementi Base:

- Il compito: T,
- Misura di prestazione: P,
- Esperienza: E.

Esempio:

- T: giocare a dama,
- P: percentuale di partite vinte,
- E: esperienza acquisita giocando contro se stesso!

Es. - T: giocare a dama, P: percentuale di partite vinte

- Che tipo di esperienza E usare ?
  - diretta (maestro: situazione scacchiera $\rightarrow$ mossa)
  - indiretta (giocare contro un maestro, giocare contro se stessi)

  Problema: E deve rappresentare **tutte** le caratteristiche di T
- Cosa apprendere ?
  - scegli_mossa:Scacchiera$\rightarrow$ Mossa
  - valutazione:Scacchiera$\rightarrow \mathbb{R}$
- Come rappresentarla ?
  - regole, rete neurale, polinomio, . . .
- Quale algoritmo di apprendimento ?
  - discesa di gradiente, programmazione lineare, algoritmi genetici, . . .

Es. Definizione della Funzione Target

Funzione di valutazione

$$V\text{:Scacchiera} \rightarrow I\!R$$

- se la posizione della scacchiera è vincente allora V(Scacchiera) = 100
- se la posizione della scacchiera è perdente allora V(Scacchiera) = -100
- se la posizione della scacchiera è pari allora V(Scacchiera) = 0
- altrimenti V(Scacchiera) = V(Scacchiera'), dove Scacchiera' è la migliore posizione sulla scacchiera che si può raggiungere giocando in modo ottimo a partire da Scacchiera

Regola operativa: $\boxed{V_{train}(b) \leftarrow \hat{V}(successor(b))}$,

dove $\hat{V}(\cdot)$ è una implementazione di $V(\cdot)$

Es. Rappresentazione della Funzione Target

Funzione Lineare $\hat{V}$ (assumiamo che $V(\cdot)$ sia lineare, ma non è detto che lo sia !!)

$$\hat{V}(s) = w_0 + w_1 \cdot pn(s) + w_2 \cdot pb(s) + w_3 \cdot dn(s) + w_4 \cdot db(s) + w_5 \cdot np(s) + w_6 \cdot bp(s)$$

dove s= Scacchiera, e

- $pn(s)$ numero di pedine nere su s (c_1)
- $pb(s)$ numero di pedine bianche su s (c_2)
- $dn(s)$ numero di dame nere su s (c_3)
- $db(s)$ numero di dame bianche su s (c_4)
- $np(s)$ numero di pezzi neri in presa su s (c_5)
- $bp(s)$ numero di pezzi bianchi in presa su s (c_6)

sono le caratteristiche c_i usate per descrivere s

Es. di Algoritmo di Apprendimento

Regola LMS: (discesa di gradiente)

Ripetere fino al raggiungimento del criterio di stop

- Selezionare a caso un esempio di training $(s, V_{train}(s))$
  1. calcolare $errore(s) = V_{train}(s) - \hat{V}(s)$
  2. per ogni caratteristica i, aggiornare i pesi w_i nel seguente modo
     $w_i \leftarrow w_i + \eta \cdot c_i \cdot errore(s)$

η è il coefficiente di apprendimento (passo di discesa di gradiente)

Fasi di Progettazione