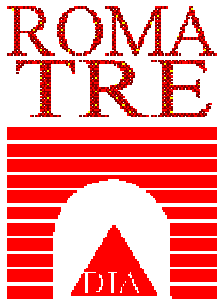
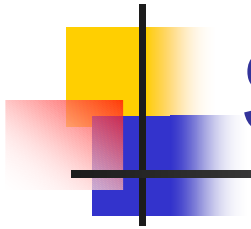


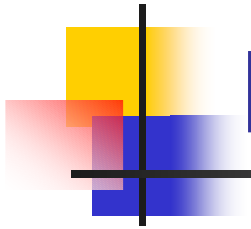
Machine Learning -1

Seminari di Sistemi Informatici



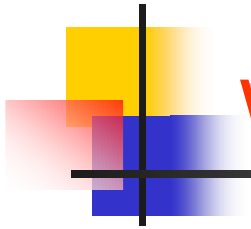
Sommario

- Problemi di apprendimento **Well-Posed**
  - Esempi di problemi well-posed
- **Progettazione** di un sistema di apprendimento
  - Scelta della *Training Experience*
  - Case study: il gioco della dama
  - Schema di un sistema generale di apprendimento
- Riflessioni



Riferimenti

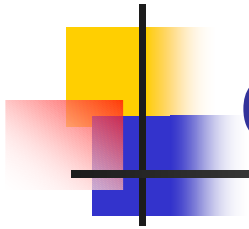
- Mitchell, T.M. (1997). *Machine Learning.* McGraw Hill Int. Ed*..* Capp. I-II.(via Pincherle)
- *Russel, S., Norvig, P.* (1995). *Artificial Intelligence A Modern Approach.* Prentice Hall Int. Ed.. Sez.18.5-18.6.
- Lucidi (disponibili sul sito)



Well-Posed Learning Problems-1

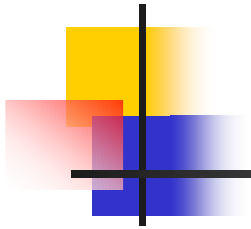
"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T , as measured by P, **improves with experience E**."

(Mitchell, 1997)



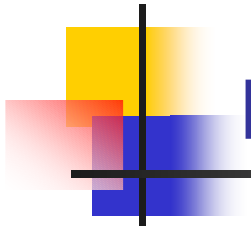
Concetti fondamentali

- **Task T**: obiettivo del sistema
  - Giocare a dama
  - Guidare un autoveicolo
  - Riconoscere parole parlate
  - ...
- **Experience E**: insieme di addestramento dal quale apprendere
  - Partite giocate
  - Percorsi
  - ...
- **Performance measure P**: misura della capacità di eseguire il task
  - # di partite vinte
  - # di parole ben classificate
  - ...



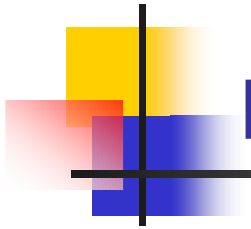
Esempi

- Speech Recognition
  - SPHINX system (Lee, 1989): riconoscimento fonemi
- Guida autoveicolo
  - ALVINN system (Pomerlau, 1989)
- Classificazione nuove strutture astronomiche
  - NASA: classificazione oggetti celesti (Fayyad et al., 1995)
- Backgammon
  - TD-Gammon (Tesauro, 1992, 1995): apprendimento su 1 milione di partite giocate contro se stesso.
- Checkers: apprendimento del gioco della dama



Problemi well-posed

- Gioco della Dama
  - Task T: giocare a dama
  - Performance measure P: % di partite vinte
  - Training Experience E: partite giocate contro se stesso
- Riconoscimento di caratteri scritti a mano
  - Task T: riconoscere e classificare parole scritte a mano memorizzate come immagini
  - Performance measure P: % di parole correttamente classificate
  - Training Experience E: un database di parole scritte a mano insieme alla loro classificazione



Problemi well-posed

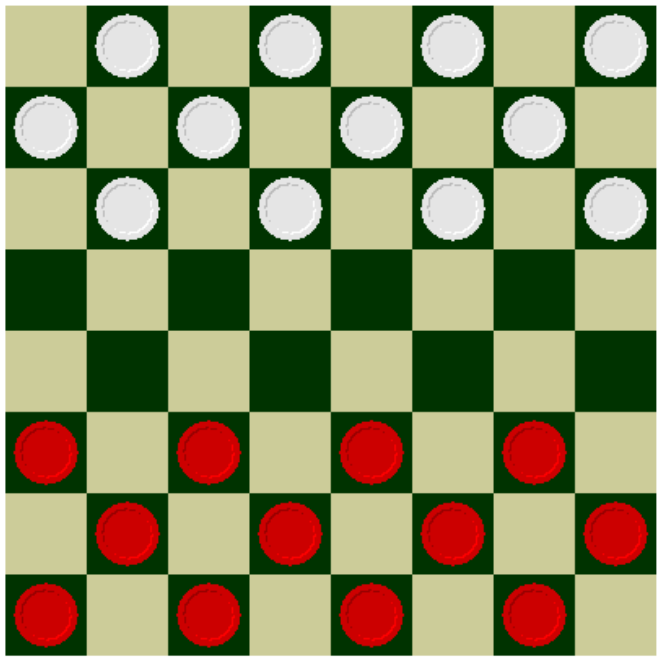
- Robot che guida un autoveicolo
  - Task T: guidare un autoveicolo su una strada pubblica
  - Performance measure P: distanza percorsa prima di un errore
  - Training Experience E: sequenza di immagini

Obiettivi generali

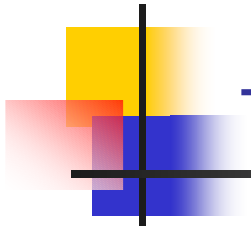
- Definire in modo preciso una classe generale di problemi che includono forme interessanti di apprendimento
- Esplorare algoritmi che risolvono problemi di apprendimento
- Comprendere la struttura fondamentale di problemi e processi di apprendimento

Designing a Learning System:Checkers

Checkers

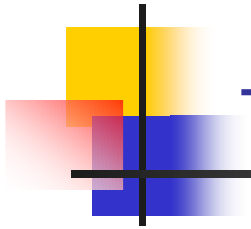


Case study: progettare un sistema che apprenda a giocare a dama.
=> **Determinare T, E e P**.



Training Experience

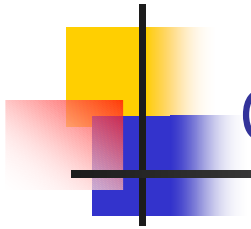
1. Feedback
   - Direct feedback training examples: insieme di singole configurazioni della scacchiera insieme alla mossa corretta per ciascuno di essi.
   - Indirect feedback training examples:insieme di sequenze di mosse insieme ai risultati finali delle partite giocate (è difficile ricavare informazioni sulla bontà delle singole mosse)
2. Controllo della sequenza di training examples
3. Distribuzione dei training examples
   - Quanto rappresenta bene la distribuzione di esempi sui quali il sistema verrà misurato



Training Experience: assunzioni

- Il sistema apprende attraverso partite contro sè stesso
  - Non necessita di un external trainer
  - Possono essere generati molti training esempi

Possiamo definire il problema di apprendimento della dama

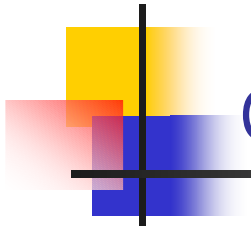


Checkers: definizione del problema

- Task T: giocare a dama
- Performance measure P: % di partite vinte in un dato torneo
- Training Experience E: partite giocate contro sè stesso

Da stabilire:

1. Il tipo esatto di conoscenza da apprendere
2. Una rappresentazione per la conoscenza target
3. Un meccanismo di apprendimento



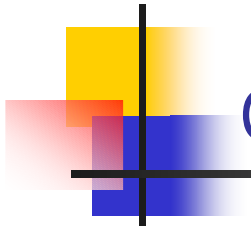
Checkers: Choosing the target Function

- Determinare esattamente che tipo di conoscenza deve essere appresa
- Come questa conoscenza sarà utilizzata dal programma
- Il programma deve, fra tutte le mosse legali possibili, scegliere la migliore a partire da uno stato qualunque della scacchiera

ChooseMove : B -> M

Input: qualunque stato ammesso della scacchiera
Output: una qualche mossa dall'insieme *M* delle mosse legali

=> Migliorare P per il task T significa trovare una *funzione target*



Checkers: Choosing the target Function

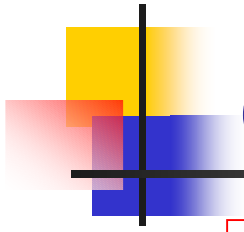
- *ChooseMove* è una scelta naturale della target function
- *ChooseMove* è però difficile da apprendere per via del tipo di indirect experience disponibile al sistema

=>Nuova Target Function

V : B -> R

Target Function V: mapping tra ogni stato ammesso della scacchiera e l'insieme dei numeri reali: punteggio più alto allo stato più promettente per l'esito della partita

=> Il sistema può selezionare la mossa migliore a partire da qualunque stato della scacchiera



Checkers: Choosing the target Function

- Criteri di assegnazione valore di V a partire da B:
  - Punteggi più alti a stati migliori
- Definiamo noi una funzione (ricorsiva) V(b), b= stato scacchiera, b$\in$B:
  - Se b è uno stato finale positivo => V(b)=100
  - Se b è uno stato finale negativo => V(b)=-100
  - Se b è uno stato finale “patta” => V(b)=0
  - Se b è uno stato intermedio => V(b) = V(b’) essendo b’ il migliore stato finale che può essere raggiunto a partire dallo stato b e giocando in modo ottimo fino alla fine del gioco.
  - E’ pero una definizione **nonoperazionale**: V non può essere realmente computata.

Checkers: Choosing the target Function

Definizione **operazionale** di V: una definizione che può essere realmente utilizzata dal sistema per valutare stati e selezionare le giuste mosse in limiti di tempo realistici

⇓

Ricerca di una descrizione operazionale della funzione (ideale) target V

= > Si approssima la funzione target ideale V con una funzione $\hat{V}$

Checkers: Function Approximation

- $\hat{V}$: funzione realmente appresa dal sistema
- V: funzione target ideale da approssimare con $\hat{V}$
- Definizione di $\hat{V}$:
  - X_1: numero di pedine nere sulla scacchiera
  - X_2: numero di pedine bianche sulla scacchiera
  - X_3: numero di dame nere sulla scacchiera
  - X_4: numero di dame bianche sulla scacchiera
  - X_5 : numero di pedine nere mangiate dal bianco
  - X_6 : numero di pedine bianche mangiate dal nero

Stato scacchiera

$$\hat{V}(b) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + w_5 x_5 + w_6 x_6$$

W_k: pesi da determinare da parte dell'algoritmo di apprendimento

Checkers: Partial Design

- **Learning Task**
  - Task T: giocare a dama
  - P: % di partite vinte in un torneo
  - E: partite giocate contro sè stesso
- **Design Choices**
  - Target Function V : Board -> R
  - Target Function representation:

$$\hat{V}(b) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + w_5 x_5 + w_6 x_6$$

Checkers: Choosing a Function Approximation ($\hat{V}$) Algorithm

- Esempio di training: coppia ordinata della forma <b, $V_{train}(b)$> essendo b lo stato della scacchiera e $V_{train}(b)$ il valore associato ad esso
- Esempio di training: <3,0,1,0,0,0,+100>: il bianco ha vinto.
- Gli esempi di training disponibili per l'apprendimento sono però di tipo indiretto: le informazioni disponibili sono solo vittoria o sconfitta (quindi di fine partita)
- => Costruire una procedura che determini gli esempi di training a partire dall'esperienza indiretta disponibile al learner
- => Modificare i pesi w in modo tale da approssimare al meglio la funzione target V

Checkers: Estimating Training Values

- Si ha bisogno quindi di esempi che assegnino agli stati della scacchiera un punteggio
- Per gli stati finali è facile mentre per gli stati intermedi?

=>Poniamo il valore associato a ciascuno stato intermedio uguale al valore dello stato successivo:

$$V_{train}(b) \leftarrow \hat{V}(Successor(b))$$

- b: stato scacchiera
- Successor(b): lo stato successivo a b per il quale tocca al programma muovere
- V: target ideal function

Sotto certe condizioni i valori tendono a V_{train}

Checkers: Adjusting the weights w

- Determinare i pesi w in modo tale da adattare al massimo (best fit) l'algoritmo agli esempi di addestramento $<b, V_{train}(b)>$
- Regola Least Mean Squares (LMS): si definisce la migliore hypothesis o insieme dei pesi come quella/o che minimizza l'errore quadratico E:

$$E \equiv \sum_{\forall <b, V_{train}(b)> \in \text{training examples}} \left(V_{train}(b) - \hat{V}(b) \right)^2$$

- **Regola LMS per l'aggiornamento dei pesi w:**
  - Per ogni training example $<b, V_{train}(b)>$
    - Calcola con i pesi attuali $\hat{V}$
    - Aggiorna ciascun peso w_i con la regola

η: costante (0.1) che modera la velocità di aggiornamento dei pesi w

$$w_i \leftarrow w_i + \eta \left(V_{train}(b) - \hat{V}(b) \right) x_i$$

Checkers: LMS rule

- Perchè questa regola funziona?

$$\left(V_{train}(b) - \hat{V}(b)\right) = 0 \Rightarrow \text{i pesi w non cambiano}$$

$$\left(V_{train}(b) - \hat{V}(b)\right) > 0 \Rightarrow \text{i pesi w crescono}$$

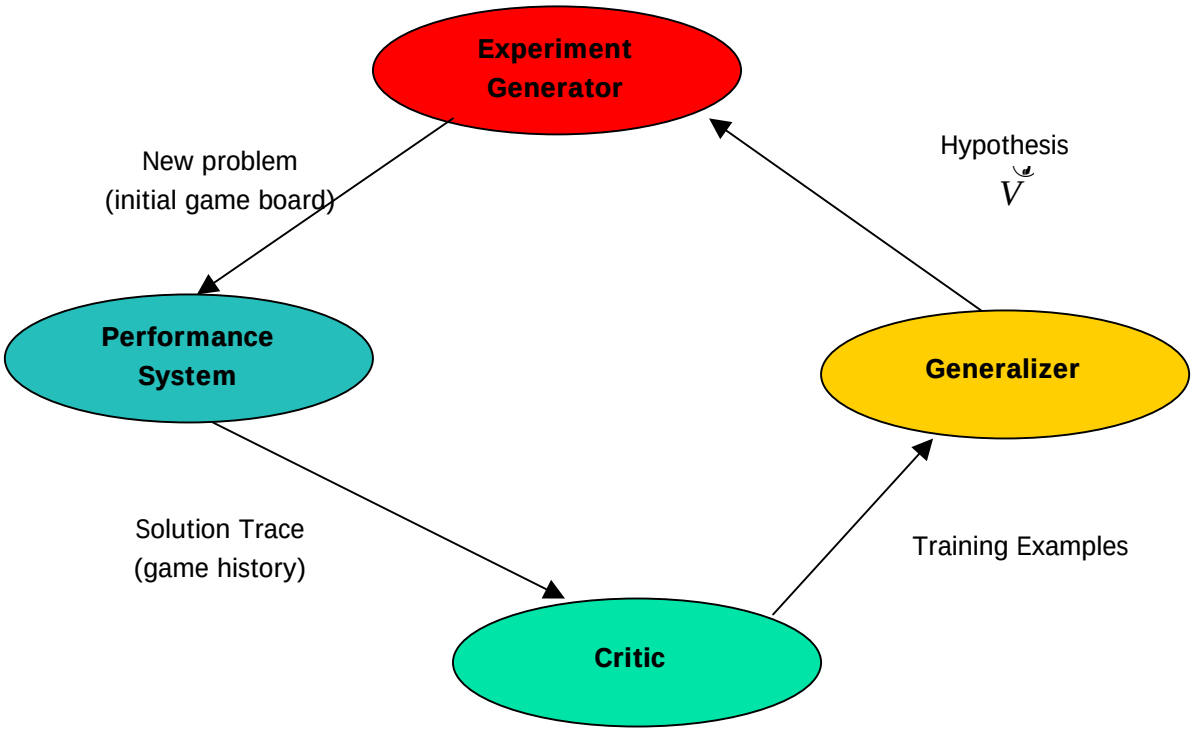
$$\left(V_{train}(b) - \hat{V}(b)\right) < 0 \Rightarrow \text{i pesi w decrescono}$$

Questo semplice tuning dei pesi in alcuni contesti converge al LSE dei valori di V_{train}

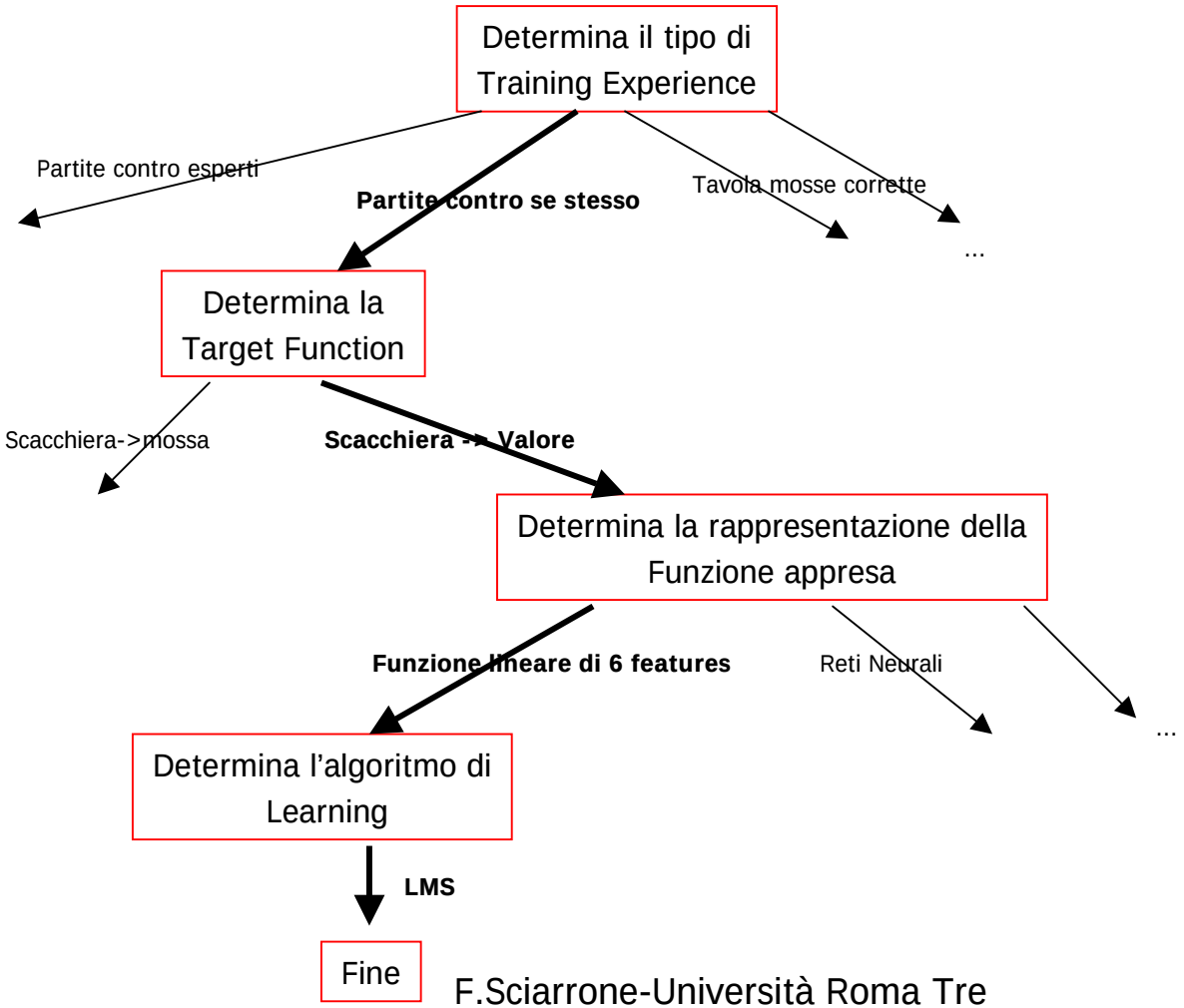
Progetto completo di un Learning System

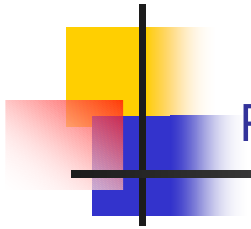
- Il modulo Performance System: è il modulo che gioca a dama utilizzando la $\hat{V}$. La sua performance migliora con il numero di partite giocate
- Il modulo Critic: prende in input una partita e produce esempi di training
- Il modulo Generalizer: implementa la LMS rule modificando i pesi w(generalizza sempre di più l'ipotesi $\hat{V}$)
- Il modulo Experiment Generator: prende in input la ipotesi $\hat{V}$ corrente e genera un nuovo problema da esplorare (una nuova partita)

Progetto completo di un generico Learning System



Progettazione del sistema Checkers





Riflessioni

- Training Experience
  - Quanti esempi di training sono necessari al sistema?
  - Come correlare il training set all'inferenza sulla funzione da imparare
- Algoritmi
  - Quali algoritmi esistono per apprendere funzioni target generali a partire da specifici esempi di training
  - In quale contesto convergono gli algoritmi dato un set di training data
  - .....