Algorithmic Game Theory

Daniele Avolio

A.A. 2023/2024

Contents

1 Introduzione

Appunti

2 Cos'è la Game Theory — Teoria dei giochi

La **teoria dei giochi** è una disciplina che studia il comportamento decisionale multi-persona, usato per fare predizioni su come **agenti razionali multipli** interagiscono o si comportano in situazioni di *cooperazione* o in situazione di *conflitto*.

Alcune definizioni di termini:

- **Conflitto**: le azioni dei giocatori hanno effetto sugli Algorithmic
- **Cooperazione**: I giocatori possono collaborare per raggiungere un obiettivo
- **Comportamento razionale**: I giocatori vogliono massimizzare la loro *utilità attesa — expected utility*
- **Predizione**: Il nostro obiettivo è sapere cosa faranno i giocatori, utilizzando *solution concepts - concetti di soluzione*

2.1 E cosa significa Algorithmic Game Theory?

Possiamo dire che algorithmic game theory è un punto d'incontro tra **game theory** e **algorithm design** che punta a ***progettare algoritmi*** *che permettono degle strategie in specifici ambienti.*

2.2 Coalition Games

La **coalition game theory** è una branca della game theory che studia le interazioni tra gruppi di giocatori, che **collaborano** per *raggiungere un obiettivo comune.*

Nota - Shapley Values: Il concetto di **Shapley Values** è un concetto che permette di *spiegare*, circa, come un algoritmo di **machine learning** ha preso una decisione. Ad esempio, mostra le *feature* che hanno avuto un impatto maggiore nella decisione finale della predizione. In pratica mostra i vari *Join — Coalizioni* di features.

Quali sono le domande più importanti in questa sezione?

- Quale coalizione è più probabile che venga formata?
- In che modo i giocatori devono dividere il premio? *(Payoff)*

2.3 Non-Cooperative Games

In questo tipo di giochi, i giocatori **non hanno coalizioni** o comunque non ne hanno bisogno.

Alcuni giochi che fanno parte di questa categoria:

- Scacchi
- Sasso-Carta-Forbice
- *Il dilemma del prigioniero*

	Giocatore 1	
Giocatore 2	Collabora	Tradisci
Collabora	(-1,-1)	(-5,0)
Tradisci	(0,-5)	(-3,-3)

Figure 1: Esempio di dilemma del prigioniero

In questo gioco, la **strategia** migliore per il singolo è quella di **tradire** l'altro giocatore, in quanto è quella che massimizza la sua utilità, precisamente andrebbe a **perdere 0 punti**, mentre l'altro giocatore ne perderebbe 5.

2.4 Tree Decomposition

Alcuni problemi sui grafi hanno una complessità di **NP-HARD** su dei grafi arbitrari, e hanno bisogno di alcune soluzioni che avranno implementazioni complesse e **programmazione dinamica.**

2.5 Computational Social Choice

Questa sezione parla di computazione di risultati risultanti da **regole di voto** — **voting rules** e quali problemi ci possono essere nel rappresentare le preferenze dei giocatori.

	Verdetto		
	Evidenza1	Evidenza2	Colpevole
Giudice1	1	0	Innocente
Giudice2	0	1	Innocente
Giudice3	1	1	Colpevole

Figure 2: Esempio di votazione

Maggiore è il numero di persone che votano, maggiore è la probabilità che il risultato sia corretto.

2.6 Mechanism Design

E' un tipo di **reverse game theory**. Invece di analizzare come i giocatori si comportano in un gioco, lo scopo del *mechanism design* è quello di **creare un gioco** per portare i giocatori a ***comportarsi in un modo specifico*** che *vogliamo noi.* Un esempio molto semplice è il *maccanismo di asta di Ebay.* Altro esempio è quello dei *carrelli dei supermecati.* Il fatto di dover utilizzare una moneta per utilizzare il carrello **porta la persona** a dover riportare il carrello nello stesso posto, invece di lasciarlo in un luogo qualsiasi del supermercato.

2.7 Fair Division of Indivisible Goods

In questa sezione si parla di come dividere delle risorse in modo **fair** tra i giocatori. Ok?

2.8 Cake Cutting

In questa sezione si parla di come dividere dei **beni continui** in bsae alle *preferenze dei giocatori.*

1. Fairness
2. Proportionality
3. Envy-freeness

Appunti di Laboratorio

3 Introduzione alla teoria dell'utilità e decision making

3.1 Concetti di base

3.1.1 ALTERNATIVE

Parliamo di *agenti* che devono scegliere un'*alternativa* da un'insieme $\mathcal{X}$ di alternative. Questo insieme di alternative ha degli elementi che possono essere **esaustivi** o **mutualmente esclusivi**.

Esepmio: § {

- DL = Deep Learning
- AGT = Algorithmic Game Theory
- DLAGT = Deep Learning Algorithmic Game Theory
- N = None

}

3.1.2 PREFERENZE

Con il termine **preferenze** identifichiamo una relazione $\succcurlyeq$ su $\mathcal{X}$, che è un sottoinsieme di $\mathcal{X} \times \mathcal{X}$. Le preferenze possono essere:

- **complete** se $\forall x, y \in \mathcal{X}$ vale $x \succcurlyeq y$ oppure $y \succcurlyeq x$
- **transitive** se $\forall x, y, z \in \mathcal{X}$ vale $x \succcurlyeq y$ e $y \succcurlyeq z$ allora $x \succcurlyeq z$

3.1.3 Relazione di Preferenza

Una preferenze è una **relazione di preferenza** se è sia **completa** che **transitiva**.

Si chiama preferenza **stretta** se $x \succ y \iff x \succcurlyeq y$ e $x \not\succcurlyeq y$.
Si chiama **indifferenza** se $x \sim y \iff x \succcurlyeq y$ e $x \preccurlyeq y$.

3.1.4 Rappresentare le preferenze come utilità

Una relazione di preferenza puà essere tradotta in una funzione di utilità del tipo $u : \mathcal{X} \to \mathbb{R}$. Si può fare in questo modo:

$$x \succcurlyeq y \iff u(x) \geq u(y) \quad \forall x, y \in \mathcal{X} \tag{1}$$

Esempio: Se un agente dovesse trovare x almeno buono quanto y, allora la funzione di utilità $u(x)$ deve essere almeno alta quanto $u(y)$. Cioè l'agente è come se stesse **massimizzando** il valore di $u(var)$.

Rappresentazione ordinale Teorema 1: Sia $\mathcal{X}$ un insieme finito di alternative e sia $\succcurlyeq$ una relazione di preferenza su $\mathcal{X}$. Allora una preferenza può essere rappresentata come una funzione di utilità $u : \mathcal{X} \to \mathbb{R}$ se e solo se è **completa** e **transitiva**. In più, se $f : \mathcal{R} \to \mathcal{R}$ è una funzione monotona crescente, allora $f \circ u$ rappresenta la stessa preferenza di $u \succcurlyeq$

Nota: dall'ultimo statement, l'ordine ha rilevanza.
Per essere valido ci sono 2 condizioni necessarie:

- Transitività: Cioè, dato $\mathcal{X} = \{a, b, c\}$, suponiamo che $a \succ b \succ c \succ a \implies u(a) > u(b) > u(c) > u(a)$. Questo sarebbe **assurdo**.
- Completezza: Se abbiamo preferenze incomplete, allora al massimo possiamo costruire un ordine per un sottoinsieme di $\mathcal{X}$.

Dimostrazione
La transitività e la completezza sono necessarie e sufficienti. Supponiamo di avere l'insieme $X = \{X_1, \ldots, X_n\}$. Possiamo suddividere gli elementi di X in k classi di indifferenza $C_1, \ldots, C_k$ tali che $C_1 \succ C_2 \succ \ldots \succ C_k$. In questo modo, possiamo definire la funzione di utilità u in modo che:

$$\begin{aligned} u(x) &= k \quad \forall x \in C_1, \\ u(x) &= k - 1 \quad \forall x \in C_2, \\ &\ldots \\ u(x) &= 1 \quad \forall x \in C_k. \end{aligned}$$

In questo contesto, $\succ$ rappresenta la relazione di preferenza.

3.2 Le lotterie

Una lotteria è una tupla $\mathcal{L} = (p_1, x_1; p_2, x_2 \ldots, p_n, x_n)$.

- Con prezzo monetario $x_1, x_2, \ldots, x_n \in X \subseteq R$.
- Distribuzione di probabilità $(p_1, p_2, \ldots, p_n)$.

Quindi con $\mathcal{L}$ viene definito l'insieme delle lotterie semplici.