Agente Intelligente per il Blackjack

Alessio Lavore 1985851 Corso di Intelligenza Artificiale 2024, Sapienza

Indice

1. Introduzione e Obiettivi 2
1.1. Scopo e obiettivi del progetto 2
1.2. Rilevanza dell'approccio probabilistico per il blackjack 2
2. Descrizione del Gioco e Sfide Principali 2
2.1. Regole fondamentali del blackjack 2
2.2. Aspetti probabilistici rilevanti e strategie di base 3
2.3. Sfide specifiche per un agente basato su calcolo probabilistico 4
3. Modello Probabilistico e Struttura dell'Agente 4
3.1. Concetti di valore atteso e distribuzioni di probabilità 4
3.2. Implementazione delle logiche decisionali per le mosse di gioco 6
3.3. Calcolo delle probabilità e decisioni ottimali 7
4. Simulazioni e Valutazione delle Prestazioni 8
4.1. Configurazione delle simulazioni e metodi di valutazione 8
4.2. Confronto con strategie base 8
4.3. Analisi dei risultati 9
5. Discussione e Conclusione 10
5.1. Efficacia dell'approccio probabilistico 10
5.2. Conclusioni sui risultati e prospettive di sviluppo futuro 11
6. Appendici e Riferimenti 11
6.1. Codice sorgente 11
6.2. Riferimenti bibliografici 11

1. Introduzione e Obiettivi

1.1. Scopo e obiettivi del progetto

In questo progetto, mi sono posto l'obiettivo di sviluppare un agente autonomo capace di giocare a blackjack utilizzando un approccio basato su calcoli probabilistici e di valore atteso, piuttosto che sull'apprendimento automatico o su tecniche di machine learning.
L'approccio probabilistico ci consente di analizzare il gioco tramite inferenze statistiche, sfruttando le probabilità di ciascun evento e il valore atteso delle possibili azioni, al fine di prendere decisioni ottimali senza necessità di allenare un modello. Questo tipo di agente, infatti, basa le proprie scelte sull'analisi delle distribuzioni di probabilità delle carte e sui risultati attesi delle possibili mosse (come "stare" o "chiedere carta"), con l'obiettivo di massimizzare le probabilità di vittoria e il guadagno atteso che ne consegue.

1.2. Rilevanza dell'approccio probabilistico per il blackjack

Il blackjack è un gioco di carte molto studiato in ambito statistico per via della sua struttura semplice e per la varietà di combinazioni possibili nelle mani di gioco, che ne fanno un interessante caso di studio per modelli decisionali. Diversamente da altri giochi d'azzardo puramente basati sulla fortuna, il blackjack incorpora una significativa componente di abilità e strategia, dato che i giocatori devono decidere se rischiare ulteriori carte o fermarsi in base alle carte già presenti in gioco. Questo fattore rende il blackjack particolarmente adatto all'applicazione di calcoli probabilistici, perché permette di prendere decisioni razionali.
Inoltre, l'approccio probabilistico in contrasto a quelli di apprendimento automatico garantisce l'explainability delle decisioni dell'agente, mantiene il codice strutturalmente semplice e richiede meno risorse sia computazionali che temporali.

2. Descrizione del Gioco e Sfide Principali

2.1. Regole fondamentali del blackjack

Il blackjack è un gioco particolarmente semplice che si basa su un numero limitato di regole.
In un tavolo da blackjack è sempre presente un dealer che gestisce il gioco e possono partecipare da 1 a 7 giocatori ognuno con il proprio capitale.
Una partita è composta dalle varie mani che si susseguono le quali sono indipendenti l'una dalle altre se non per il mazzo, il quale non viene rimischiato fino al raggiungimento di una taglia posta inizialmente, rendendo di fatto possibile conteggiare le carte già uscite nelle mani precedenti.
Il mazzo può essere composto, in base al casinò, da 2 a 6 mazzi di carte francesi, esclusi i jolly.
Ogni mano è a sua volta definita da varie fasi, una iniziale in cui ogni giocatore deve scegliere una puntata con la quale entrare, seguita dalla fase di gioco nella quale il dealer consegna scoperte due carte ad ogni giocatore mentre a sé stesso una carta scoperta ed una coperta.
Successivamente a turno i giocatori hanno la possibilità di effettuare delle decisioni:

- Se hanno due carte uguali per numero dividerle in due mani e quindi giocare come se fossero due giocatori diversi e nel farlo raddoppiare la puntata iniziale;
- Se hanno un punteggio iniziale tra 9 e 11 gli è concesso di raddoppiare la propria puntata ma a condizione di poter chiedere solamente una terza carta prima di terminare il proprio turno;

Fino al raggiungimento o superamento di 21 punti di chiedere una carta in più o se fermarsi e terminare il proprio turno.

Solo una volta che tutti i giocatori hanno a terminato il proprio turno il dealer scopre la sua seconda carta e poi inizia ad aggiungersi nuove carte fino al raggiungimento o superamento di 17 punti, indipendentemente dai punteggi dei giocatori.
Infine, un'ultima fase in cui vengono confrontati uno ad uno i punteggi dei giocatori con quello del dealer e in base ad essi assegnare o meno una vincita in rapporto alla puntata iniziale.

Per determinare le vincite si considerano i seguenti casi:

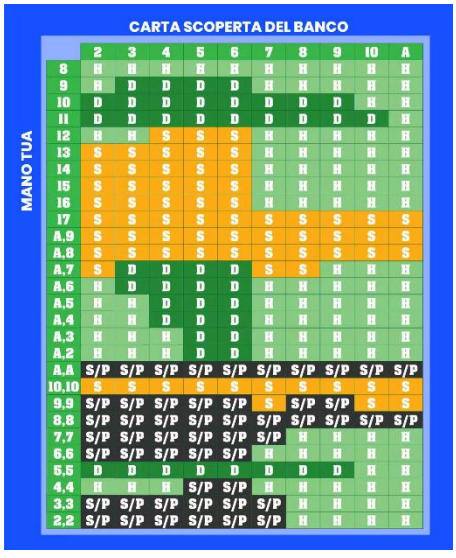
- Se il giocatore sballa, ossia supera il 21, perde a prescindere dal punteggio dealer la sua puntata;
- Se il dealer sballa e il giocatore no, esso riceve due volte la sua puntata;
- Se il punteggio del giocatore è minore di quello del dealer, perde la propria puntata;
- Se il punteggio del giocatore e del dealer sono uguali, riceve indietro la sua puntata;
- Se il punteggio del giocatore è maggiore di quello dl dealer, riceve due volte la sua puntata;
- Se il giocatore effettua blackjack, ossia 21 punti con le prime due carte, ed il dealer no, riceve indietro due volte e mezzo la sua puntata;

2.2. Aspetti probabilistici rilevanti e strategie di base

Nel blackjack il mazzo non viene rimescolato fino al raggiungimento di una taglia posta all'incirca alla sua metà; perciò, è possibile contare tutte le carte già uscite nelle mani precedenti al fine di sapere le carte ancora rimanenti nel mazzo.
Questo rende possibile calcolare in ogni momento con esattezza la probabilità che una determinata carta sia la prossima nel mazzo, e di conseguenza in ogni fase del gioco il valore atteso delle possibili decisioni.

Nel tempo i giocatori hanno appreso come sfruttare a proprio vantaggio questa particolarità del gioco e sono state inventate alcune strategie di base, le più comuni sono:

- L'uso di una tabella decisionale
- Conteggio tramite il sistema Hi-Lo



Nel caso di uso della tabella decisionale si effettuerà la mossa consigliata in base alle proprie carte iniziali e a quella scoperta del dealer, considerando che le mosse sono state calcolate unicamente nel caso base di mazzo pieno e perciò non tengono conto dell'evoluzione della partita, le mosse definite sono:

- H: hit, ossia chiedi carta
- S: stand, ossia stai
- D: double, ossia raddoppia se concesso dal casino
- S/P: split, ossia dividi se consesso dal casino

In figura una tabella di blackjack

Nel caso invece del sistema Hi-Lo l'obiettivo è quello di tenere a mente un punteggio andando per ogni carta che esce sommare o sottrarre in base a queste semplici regole:

- Aggiungere 1 se la carta è dal 2 al 6
- Sottrarre 1 se la carta è un 10, una figura o un asso
- Non fare nulla se la carta è tra il 7 ed il 9
- Ricominciare da 0 quando viene mischiato il mazzo

In base al valore di questo punteggio il giocatore dovrà modificare la sua strategia andando ad aumentare o diminuire la tendenza al rischio sia nelle puntate che nelle decisioni di gioco.
Infatti, se il conteggio è negativo significa che il mazzo è composto principalmente da carte piccole e le probabilità sono a favore del dealer, mentre se il conteggio è positivo più è alto e più sono presenti assi, figure e 10 nel mazzo e perciò le probabilità sono a favore del giocatore.

2.3. Sfide specifiche per un agente basato su calcolo probabilistico

Un agente basato sul calcolo delle probabilità che gioca a blackjack deve essere in grado di tenere traccia di tutte le carte uscite e di determinare per ogni possibile stato la mossa migliore da effettuare per massimizzare la probabilità di vincita.
Inoltre, deve saper interpretare la bontà del mazzo con le sue carte rimanenti per scegliere la puntata iniziale più opportuna al fine di massimizzare anche il guadagno atteso di ogni mano.

I requisiti per un agente intelligente si possono quindi ricapitolare come segue:

- Saper contare le carte uscite per determinare con accuratezza quelle rimanenti nel mazzo;
- Saper calcolare per ogni combinazione di carte possedute e carta scoperta del dealer la migliore decisione da prendere, tenendo conto delle carte nel mazzo;
- Saper calcolare, considerando le carte rimaste nel mazzo, la puntata migliore;

Per fare ciò si possono utilizzare vari approcci, quello che ritengo più efficace e che ho scelto di implementare verrà illustrato nei punti 2 e 3 del prossimo capitolo.

3. Modello Probabilistico e Struttura dell'Agente

3.1. Concetti di valore atteso e distribuzioni di probabilità

Nel blackjack, il valore atteso e le distribuzioni di probabilità rappresentano le basi fondamentali su cui l'agente può prendere decisioni ottimali. Questi concetti permettono di trasformare le incertezze del gioco in un modello decisionale razionale, calcolando la probabilità degli eventi e stimando le mosse più vantaggiose a lungo termine.

Valore Atteso (Expected Value)

Il valore atteso rappresenta il guadagno medio che ci si può attendere da una determinata mossa nel lungo periodo. In pratica, il valore atteso è la somma dei possibili risultati, ciascuno ponderato per la probabilità che si verifichi.

In formula, il valore atteso E(X) per una determinata azione è:

$$E(X) = \sum_{i=1}^{n} p_i \times x_i$$

dove:

- p_i è la probabilità che esca l'esito i,
- x_i è il guadagno o la perdita associata all'esito i,
- $E(X)$ è il valore atteso,
- X è l'azione da svolgere,
- n è il numero di possibili esiti.

Esempio di Valore Atteso:
Consideriamo un semplice gioco d'azzardo in cui si lancia un dado a sei facce e si guadagna un dollaro per ogni numero uscito. I possibili esiti e le relative probabilità sono:

- Guadagnare 1 dollaro se esce il numero 1 (probabilità $\frac{1}{6}$)
- Guadagnare 2 dollari se esce il numero 2 (probabilità $\frac{1}{6}$)
- Guadagnare 3 dollari se esce il numero 3 (probabilità $\frac{1}{6}$)
- Guadagnare 4 dollari se esce il numero 4 (probabilità $\frac{1}{6}$)
- Guadagnare 5 dollari se esce il numero 5 (probabilità $\frac{1}{6}$)
- Guadagnare 0 dollari se esce il numero 6 (probabilità $\frac{1}{6}$)

Il valore atteso può essere calcolato come segue:

E(X) = $(\frac{1}{6} \cdot 1) + (\frac{1}{6} \cdot 2) + (\frac{1}{6} \cdot 3) + (\frac{1}{6} \cdot 4) + (\frac{1}{6} \cdot 5) + (\frac{1}{6} \cdot 0) = \frac{15}{6} = 2.5$

Ciò significa che, in media, un giocatore può aspettarsi di guadagnare 2,5 dollari per ogni lancio del dado.

Distribuzione di probabilità

Le distribuzioni di probabilità descrivono come le probabilità sono distribuite su tutti i possibili esiti di un esperimento casuale. Una distribuzione di probabilità fornisce una visione complessiva delle probabilità associate a ciascun possibile valore che una variabile casuale può assumere.

Esempio di Distribuzione di Probabilità:
Consideriamo di nuovo il dado a sei facce.
La distribuzione di probabilità dei risultati di un singolo lancio può essere rappresentata come:

- Risultato 1: probabilità $\frac{1}{6}$
- Risultato 2: probabilità $\frac{1}{6}$
- Risultato 3: probabilità $\frac{1}{6}$
- Risultato 4: probabilità $\frac{1}{6}$
- Risultato 5: probabilità $\frac{1}{6}$
- Risultato 6: probabilità $\frac{1}{6}$

Oppure nella forma:

{1: $\frac{1}{6}$, 2: $\frac{1}{6}$, 3: $\frac{1}{6}$, 4: $\frac{1}{6}$, 5: $\frac{1}{6}$, 6: $\frac{1}{6}$}

3.2. Implementazione delle logiche decisionali per le mosse di gioco

Per implementare un agente come descritto al punto 3 del secondo capitolo l'approccio che ho scelto per prendere la decisione statisticamente migliore è stato quello di rappresentare sia la mano dell'agente che quella del dealer come distribuzioni di probabilità sotto forma di dizionario, dove la chiave è il valore risultante delle carte in mano mentre il valore associato alla chiave è la probabilità che si abbia quella mano.
Per poi calcolare queste due distribuzioni facendo un'inferenza per ogni carta nel mazzo, andando a considerare ogni possibile combinazione di esse tra mano dell'agente e del dealer.
E infine, confrontare le due distribuzioni per calcolare tre variabili:
probabilità di vittoria, probabilità di pareggio e probabilità di sconfitta.
Usando una semplice logica come quella mostrata in figura.
Dove vado a controllare ogni coppia valore mano dealer e valore mano agente e in base ai casi spiegati al primo punto del secondo capitolo vado ad aggiornare le variabili vittoria, sconfitta e pareggio tenendo conto della probabilità di trovarsi in quel determinato stato, ossia la probabilità che la mano del dealer sia quella moltiplicata per la probabilità che la mano dell'agente sia quella.

```
probWinPasso = 0
probDrawPasso = 0
probLosePasso = 0
for m in dictValMyHand:
    for d in dictValDealerHand:
        if m == 22:
            probLosePasso += dictValMyHand[m]*dictValDealerHand[d]
        elif d == 22:
            probWinPasso += dictValMyHand[m]*dictValDealerHand[d]
        else:
            if m > d:
                probWinPasso += dictValMyHand[m]*dictValDealerHand[d]
            elif m == d:
                probDrawPasso += dictValMyHand[m]*dictValDealerHand[d]
            else:
                probLosePasso += dictValMyHand[m]*dictValDealerHand[d]
```

Ripetendo questa procedura anche nel caso in cui si chiede carta, ossia non calcolando la mano corrente contro la distribuzione di probabilità della mano del dealer ma verificando ogni possibile nuova mano dell'agente che chiede carta contro la corrispettiva distribuzione di probabilità della mano del dealer, ci ritroviamo con 6 variabili, 3 nel caso in cui si passa e 3 nel caso in cui si chiede carta. Confrontando quali probabilità sono più favorevoli l'agente è in grado di scegliere quale mossa effettuare.

Per il calcolo effettivo della distribuzione di probabilità si rimanda al prossimo punto di questo capitolo dove verrà spiegata nel dettaglio l'inferenza effettuata.

Per quanto riguarda invece la capacità dell'agente di determinare una puntata ottimale per massimizzare la vincita attesa si rimanda al sistema Hi-Lo visto in precedenza, infatti ho implementato una funzione che emula quello che è il calcolo del Running Count per determinare la quantità di assi, figure o 10 nel mazzo, rivalutato come True Count in rapporto al numero di mazzi ed infine usato per determinare la puntata come equazione lineare di una Bet Units scelta arbitrariamente come la puntata minima.
A seguire la funzione dell'agente in Python:

```python
def chooseBet(self):
    RunningCount = 0
    for carta in self.carteUscite:
        if carta[0] == '1':
            RunningCount -= 1
        elif carta[0] == '2' or carta[0] == '3' or carta[0] == '4' or carta[0] == '5' or carta[0] == '6':
            RunningCount += 1
    numMazziRimanenti = len(self.mazzo)/52
    TrueCount = int(RunningCount/numMazziRimanenti)
    if TrueCount < 0:
        return 2
    if TrueCount == 0:
        return self.BetUnits
    else:
        return int((TrueCount+1)*self.BetUnits)
```

3.3. Calcolo delle probabilità e decisioni ottimali

Come introdotto nel punto precedente di questo capitolo, il calcolo delle probabilità avviene per avere delle distribuzioni di probabilità corrette necessarie per prendere la decisione migliore.
In particolare, l'approccio che ho implementato è stato quello di usare una funzione ricorsiva la quale, venendo richiamata per ogni combinazione di carte possibili in una mano ed effettuando un ciclo in cui considera tutte le nuove carte ancora nel mazzo, aggiorna la distribuzione di probabilità originale diminuendo la probabilità di avere il vecchio valore della mano e aumentando la probabilità di avere il nuovo valore della mano dell'esatta probabilità che ci si trovi in quello stato e che esca la carta necessaria ad effettuare quel cambiamento.
Realizzando infine, al raggiungimento del caso base (che per il dealer è di avere almeno 17), una distribuzione di probabilità definitiva che rappresenta esattamente tutti i possibili esiti di quella mano tenendo conto delle carte già uscite e facilmente confrontabile con la rispettiva distribuzione della mano dell'agente per determinare le probabilità di vittoria, sconfitta e pareggio.

A seguire in figura la funzione ricorsiva usata dall'agente per calcolare la distribuzione di probabilità della mano del dealer, chiamata conoscendo la sua carta scoperta e le carte uscite:

```python
def InferenzaProbabilità(self,i,dictMazzo,lenMazzo,dictProbHand,flagAsso): # funzione ricorsiva per calcolare la distribuzione di probabilità del dealer
    if i >= 17: # caso base
        return dictProbHand
    prob = dictProbHand[i]  # questa ricorsione ha nel calcolo finale peso "prop"
    for e in dictMazzo:
        if dictMazzo[e] == 0:
            continue
        flag = False    # flag asso per il ciclo
        flag2 = flagAsso    # flag asso per la chiamata ricorsiva
        if len(e) == 3:     # se la carta è un 10 o una figura
            if i+10 <= 21:  # caso in cui non si sballa
                dictProbHand[i+10] = dictProbHand[i+10] + prob*dictMazzo[e]/lenMazzo    # la probabilità di avere i+10 punti aumenta (dove i era il punteggio della chiamata ricorsiva precedente)
                i2 = i+10
            elif flagAsso: # caso in cui c'è un asso che può essere considerato 1
                flag2 = False
                dictProbHand[i-10+10] = dictProbHand[i-10+10] + prob*dictMazzo[e]/lenMazzo
                i2 = i-10+10
            else:   # caso in cui si sballa
                dictProbHand[22] = dictProbHand[22] + prob*dictMazzo[e]/lenMazzo
                i2 = 22
        elif e[0] == '1':   # se la carta è un asso
            if i+11 <= 21:  # caso in cui l'asso verrà considerato 11
                dictProbHand[i+11] = dictProbHand[i+11] + prob*dictMazzo[e]/lenMazzo
                i2 = i+11
                flag = True
            elif i+1 <= 21: # caso in cui l'asso verrà considerato 1
                dictProbHand[i+1] = dictProbHand[i+1] + prob*dictMazzo[e]/lenMazzo
                i2 = i+1
            else:
                dictProbHand[22] = dictProbHand[22] + prob*dictMazzo[e]/lenMazzo
                i2 = 22
        else:   # se la carta è da 2 a 9
            if i+int(e[0]) <= 21:   # caso in cui non si sballa
                dictProbHand[i+int(e[0])] = dictProbHand[i+int(e[0])] + prob*dictMazzo[e]/lenMazzo
                i2 = i+int(e[0])
            elif flagAsso:  # caso in cui c'è un asso che può essere considerato 1
                flag2 = False
                dictProbHand[i-10+int(e[0])] = dictProbHand[i-10+int(e[0])] + prob*dictMazzo[e]/lenMazzo
                i2 = i-10+int(e[0])
            else:   # caso in cui si sballa
                dictProbHand[22] = dictProbHand[22] + prob*dictMazzo[e]/lenMazzo
                i2 = 22
        dictProbHand[i] = dictProbHand[i] - prob*dictMazzo[e]/lenMazzo  # la probabilità di avere i punti diminuisce di quanto sono aumentate le probabilità di avere punteggi maggiori
        newMazzo = dictMazzo.copy()
        newMazzo[e] -= 1
        if flag:
            dictProbHand = self.InferenzaProbabilità(i2,newMazzo,lenMazzo-1,dictProbHand,True)  # chiamata ricorsiva
        else:
            dictProbHand = self.InferenzaProbabilità(i2,newMazzo,lenMazzo-1,dictProbHand,flag2)
    return dictProbHand
```

4. Simulazioni e Valutazione delle Prestazioni

4.1. Configurazione delle simulazioni e metodi di valutazione

Per valutare il corretto funzionamento dell'agente, oltre che nell'ambiente di gioco classico nel quale si è liberi di formare un tavolo con sia giocatori umani che IA, è stato implementato un ambiente di gioco volto a simulare velocemente diverse partite in parallelo.
Questo ambiente sfrutta il multithreading della macchina per creare istanze diverse di una partita in cui l'agente gioca un numero limitato di mani fino al termine della simulazione, e per ogni istanza raccoglie le statistiche necessarie a fornire una media precisa della prestazione dell'agente.

In particolare, per ogni partita simulata si salvano i capitali finali dei giocatori del tavolo in modo da poterli rapportare ai capitali iniziale ed avere una stima empirica delle prestazioni.
Per avere delle stime significative il numero di simulazioni deve essere abbastanza elevato così da poter ridurre al minimo l'interferenza di casi particolari o casuali.

Per realizzare un quadro generale sull'andamento dell'agente sviluppato sono state scelte due casistiche: gioco individuale e gioco condiviso.
Ossia, simulazioni di partite in cui l'agente è l'unico giocatore partecipante al tavolo e simulazioni di partite in cui ci sono vari agenti che giocano nello stesso tavolo.
Inoltre, per avere una valutazione più realistica possibile delle prestazioni è fondamentale effettuare simulazioni di lunghezza diversa per numero di mani, passando da partite lunghe solamente 100 mani a partite lunghe 2000 mani.

4.2. Confronto con strategie base

Una volta aver raccolto abbastanza dati da poter definire l'andamento delle prestazioni dell'agente è utile avere uno strumento di confronto per capire quanto esse siano buone.
Vista la difficoltà nel trovare un agente IA riconosciuto come migliore nel giocare a blackjack invece di un paragone per difetto è stato scelto di effettuare un confronto con un agente Naive.

L'agente naive è stato realizzato similarmente all'agente intelligente come sottoclasse di giocatore ed in grado di partecipare ad una partita di blackjack prendendo in totale autonomia le decisioni durante tutta la partita. Per guidare questo agente invece di un approccio probabilistico o di apprendimento è stato scelto l'uso di una strategia base, per simulare perfettamente un giocatore inesperto. Infatti, il nostro agente naive si limiterà in ogni suo turno a seguire alla lettera la tabella decisionale mostrata al secondo punto del secondo capitolo, quindi senza considerare il cambiamento del mazzo o l'evoluzione della partita, ma solamente lo stato corrente.
Per quanto riguarda le scommesse iniziali è stato invece scelto arbitrariamente una puntata pari allo 0.1% del capitale iniziale, e quindi non dinamico in base al conteggio delle carte.

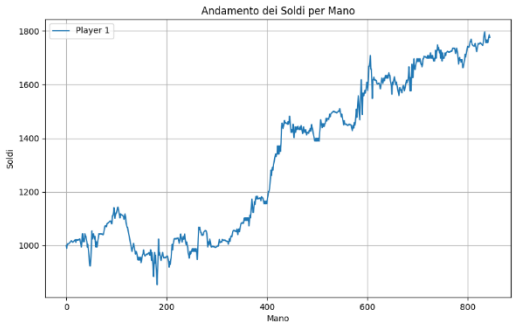
Una volta realizzato l'agente naive sono state effettuate esattamente le stesse simulazioni svolte per l'agente intelligente in modo da poter confrontare 1:1 le prestazioni di entrambe le strategie.

4.3. Analisi dei risultati

Tutte le casistiche sono state analizzate effettuando 100 simulazioni in parallelo, poi raccolte in un unico risultato tramite media aritmetica dei risultati delle singole simulazioni.

Per ogni casistica sono stati effettuati 4 test:

- Partita da 100 mani;
- Partita da 500 mani;
- Partita da 1000 mani;
- Partita da 2000 mani;



esempio andamento partita agente IA

Casi analizzati:

- Un agente IA;
- Un agente Naive;
- Un agente IA ed uno Naive;
- Quattro agenti IA;
- Quattro agenti Naive;
- Tre agenti IA e tre Naive;

Dopo svariate ore di simulazioni sono stati raccolti tutti i risultati e mostrati qui di seguito:

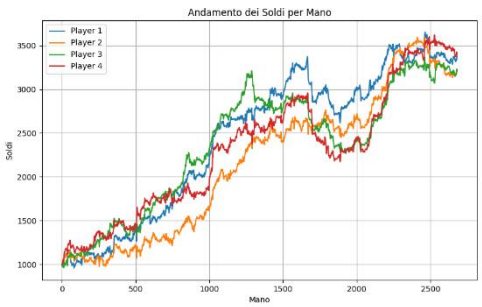
Le simulazioni che vedono coinvolto un solo agente IA hanno portato i seguenti risultati:
1.093x dopo 100 mani, 1.149x dopo 500, 1.778x dopo 1000 e 2.264x dopo 2000 mani.

Le simulazioni che vedono coinvolto solo un agente Naive hanno portato questi risultati:
0.916x dopo 100 mani, 0.686x dopo 500, 0.281x dopo 1000 e 0.026x dopo 2000 mani.

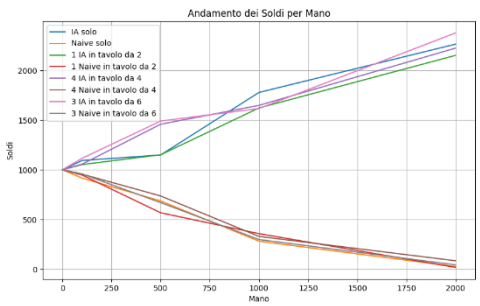
Mostrano quindi una tendenza positiva per quanto riguarda l'agente intelligente e una negativa per l'agente naive. Per quanto riguarda le simulazioni multi-agente invece abbiamo:

Le simulazioni con 1 agente IA ed 1 agente Naive:
dopo 100 mani 1.052x per l'IA e 0.947x per il Naive;
dopo 500 mani 1.151x per l'IA e 0.567x per il Naive;
dopo 1000 mani 1.622x per l'IA e 0.357x per il Naive;
dopo 2000 mani 2.151x per l'IA e 0.017x per il Naive;

Le simulazioni con 4 agenti IA:
dopo 100 mani: 1.036x 1.028x 1.069x 1.092x e quindi 1.056x di media;
dopo 500 mani: 1.500x 1.423x 1.527x 1.423x e quindi 1.458x di media;
dopo 1000 mani: 1.627x 1.701x 1.617x 1.647x e quindi 1.648x di media;
dopo 2000 mani: 2.375x 2.023x 2.305x 2.194x e quindi 2.224x di media;



esempio andamento partita con 4 agenti IA



andamento generali casi analizzati

Le simulazioni con 4 agenti Naive:
dopo 100 mani: 0.960x 0.962x 0.949x 0.954x e quindi 0.956x di media;
dopo 500 mani: 0.700x 0.707x 0.743x 0.792x e quindi 0.736x di media;
dopo 1000 mani: 0.309x 0.330x 0.346x 0.329x e quindi 0.329x di media;
dopo 2000 mani: 0.107x 0.030x 0.112x 0.087x e quindi 0.084x di media;

Infine, le simulazioni con 3 agenti IA e 3 agenti Naive:
dopo 100 mani: 1.115x di media per l'IA e 0.958x di media per il Naive;
dopo 500 mani: 1.490x di media per l'IA e 0.669x di media per il Naive;
dopo 1000 mani: 1.616x di media per l'IA e 0.298x di media per il Naive;
dopo 2000 mani: 2.377x di media per l'IA e 0.043x di media per il Naive;

I dati qui raccolti si possono raffigurare in una tabella come questa sottostante

mani\agenti	IA	Naive	IA + Naive	4 IA	4 Naive	3 IA + 3 Naive
100	1.093x	0.916x	1.052x , 0.947x	1.056x	0.956x	1.115x , 0.958x
500	1.149x	0.686x	1.151x , 0.567x	1.458x	0.736x	1.490x , 0.669x
1000	1.778x	0.281x	1.622x , 0.357x	1.648x	0.329x	1.616x , 0.298x
2000	2.264x	0.026x	2.151x , 0.017x	2.224x	0.084x	2.377x , 0.043x

Analizzando i risultati empirici ricavati dalle simulazioni possiamo affermare un evidente distacco prestazionale tra l'agente intelligente sviluppato e quello naive preso come riferimento.
Inoltre, si nota una differenza non significativa tra le prestazioni dell'agente in una partita solitaria rispetto alle stesse in una partita multi-agente, evincendo la capacità dell'agente di calcolare la migliore decisione possibile indipendentemente dalle condizioni esterne e adattarsi a qualsiasi ambiente di gioco.

L'unico vantaggio di usare un approccio Naive rispetto ad uno più ragionato è nel tempo di esecuzione delle simulazioni, e quindi in generale nella velocità con cui vengono effettuate le decisioni.
Infatti, mentre per effettuare 100 simulazioni di 1000 mani con uno o più agenti IA si impiegano dalle 4 alle 16 ore circa, per lo stesso numero di simulazioni e mani con solo agenti Naive servono solamente dai 2 ai 5 secondi circa. La differenza è molto evidente nella simulazione di un gran numero di mani, però quando si considera un ambiente più realistico come lo svolgimento di una partita reale entrambi gli approcci hanno un tempo di ragionamento inferiore a quello umano, rendendo di fatto questa differenza non significativa.

5. Discussione e Conclusione

5.1. Efficacia dell'approccio probabilistico

L'approccio probabilistico adottato per sviluppare l'agente ha dimostrato di essere efficace nell'affrontare il gioco del blackjack senza dover ricorrere a tecniche di machine learning o reti neurali. Basandosi sul calcolo del valore atteso e sulle distribuzioni di probabilità, l'agente è riuscito a gestire le proprie decisioni in modo razionale, sfruttando le informazioni disponibili per scegliere le mosse che, in media, ottimizzano le probabilità di successo. Inoltre, questo approccio ha il vantaggio di essere facilmente interpretabile e matematicamente giustificabile, il che permette di comprendere meglio i meccanismi alla base delle decisioni prese dall'agente.

5.2. Conclusioni sui risultati e prospettive di sviluppo futuro

In conclusione, l'implementazione di questo approccio ha dimostrato una buona efficacia per lo sviluppo di un agente autonomo che gioca a blackjack. L'agente ha potuto effettuare scelte ragionevoli, riducendo le probabilità di sconfitta rispetto a decisioni casuali e avvicinandosi a strategie ottimali.
I risultati statistici per quanto empirici e non dimostrati matematicamente sono un buon indice per affermare l'efficacia di questa implementazione.

Alcune possibili integrazioni che vedrei bene in futuro sono l'aggiunta di un terzo agente differente da quello naive e da quello probabilistico, il quale usa un approccio di machine learning, per poi confrontare a sua volta i risultati e verificare se è possibile ottenere risultati ancora migliori.
Altre interessanti migliorie a questa implementazione, anche se sforano dal campo di interesse del corso per il quale è stato realizzato il progetto, è il completamento di un'interfaccia grafica dinamica per rappresentare la partita in corso e rendere più semplice ed immediato per l'utente la comprensione del gioco e delle carte sul tavolo.

Ringraziamenti

Ringrazio comunque il Professor Mancini ed il Dottor Esposito, nonché tutti i membri del corso di Intelligenza Artificiale per il percorso di studio svolto il quale ha portato oltre che alla realizzazione di questo progetto anche ad un ulteriore mio interessamento verso la materia studiata, invogliandomi ancora di più ad impegnarmi in futuro (si spera non lontano) in un percorso di Laurea Magistrale sull'argomento dell'IA.

6. Appendici e Riferimenti

6.1. Codice sorgente

Repository GitHub

6.2. Riferimenti bibliografici

2.2.
Strategia di base: le tabelle del blackjack
Contare le carte nel Blackjack: come funziona il conteggio
3.1.
Valore atteso, slide del corso
Distribuzione di probabilità. slide corso