POLITECNICO DI MILANO

Corso di Laurea in
Ingegneria dell'automazione

Dipartimento di Elettronica e Informazione

Apprendimento off-line nel Poker (Batch Learning)

AI & R Lab
Laboratorio di Intelligenza Artificiale
e Robotica del Politecnico di Milano

Relatore: Prof. Marcello Restelli

Tesi di Laurea di:

Rafael Domingues Santos Vilella
Matr. 720320

Anno Accademico 2010- 2011.

Sommario

Il Poker è un gioco di elevata complessità, il vincitore è il prodotto di fortuna e strategia. Lo scopo di questo progetto è determinare strategie (politiche) efficace, a partire di uno specifico database di giocate. Sarà realizzato il machine learning col apprendimento per rinforzo, in particolare con iterazione del tipo Fitted Q-iteration e con l'utilizzo del regressore Extremly Randomized Trees.

Parole chiave: Apprendimento per rinforzo, Fitted Q-iteration, Extremly randomized trees, Computer poker competition, Poker Limit Heads Up Hold'em.

Abstract

Poker is a very complex game, the winner is established by lucky and strategy. The aim of this project is to determinate an efficient strategy (policy), from a given database of plays. It will be done the machine learning with Reinforcement Learning, with Fitted Q-Iterations utilizing a regressor Extremely Randomized Trees.

Keywords: Reinforcement Learning, Fitted Q-iteration, Extremely randomized trees, Computer poker competition, Poker Limit Heads Up Hold'em.

Indice Generale

Elenco delle Figure

Elenco di Abbreviazione

BB- Big Blind
SB- Small Blind
EXTRA- Extremely Randomized Trees
FQI- Fitted Q-iteration

Glossario

Bankroll- Le risorse totale disponibile di un giocatore per scommettere.
Pot- Quantità totale di monete scommesse nel tavolo.
Hero- Bots creati per l'algoritmo del progetto.
Mano- Ogni partita giocata, cioè, quello periodo in cui ogni giocatore tiene le stesse carte nella mano.
Giocata- Ogni azione durante una mano.
Showdown- Fine della mano, dove uno dei giocatori prende il pot.

Capitolo 1

Introduzione

Il Poker è un gioco di elevata complessità, il vincitore è il prodotto di fortuna e strategia. Lo scopo di questo progetto è determinare strategie (politiche) efficace, a partire di uno specifico database di giocate. Sarà realizzato il [8]machine learning col [1]apprendimento per rinforzo, in particolare con iterazione del tipo [2]Fitted Q-iteration e con l'utilizzo del regressore [2]Extremly Randomized Trees.

Per quanto riguarda la strategia, esistono diverse possibilità, una di queste sarebbe per esempio quella di equilibrio, prescritta dalla teoria dei giochi, che garantisce nella media un ritorno minimo uguale a zero di fronte al più forte avversario possibile. Tale strategia, anche se vantaggiosa, non è capace di adattarsi di fronte a giocatore diversi; se è stato identificato un giocatore debole, è possibile, giocando opportunamente vincere più velocemente il suo bankroll.

L'idea generale del progetto, non è identificare la politica di equilibrio, ma quella che meglio si adatta a un determinato ambiente, cioè a una base di dati con giocatori già stabiliti, utilizzando il batch learning.

Figura 1.1. Esempio di tavolo di gioco in un ambiente virtuale.

1.1 Descrizione del lavoro

L'idea generale è determinare i possibili stati del Heads Up Limit Hold'em e attribuire a ognuno il suo valore, cioè il suo ritorno medio e poter cosi determinare le migliore azione possibile e ottenere alla fine la politica.

Inizialmente si può pensare che basterebbero i concetti di Reinforcement Learning. Il problema è, però, molto più complesso. Il perché è semplice, il numero di stati per questo problema è circa 10^{14} e i database sono scarsi, nel senso che non presentano tutti gli stati possibili.

Allora, è necessario semplificare il problema e/o sviluppare strumenti capaci di determinare l'azione corretta in tutti i casi, quelli visti poche volte o anche quelli mai raggiunti.

Un'ulteriore complicazione è la stocasticità del problema, non basta analizzare soltanto una volta lo stato, le carte sono estratte casualmente, allora una valutazione giusta richiede più di uno campione.

Lo strumento utilizzato sarà il Fitted Q-Iteration, che usando un database di giocate valuterà l'utilità associata ai diversi stati tramite gli extremly-randomized trees.

Dopo di generare le politiche, queste devono essere validate, quindi sono stati sviluppati bots, che poi, giocano contro lo stesso giocatore con cui è stato generato il database.

1.2 Struttura degli argomenti

La struttura organizzativa del progetto sarà la seguente:

I prossimi due capitoli definiscono i concetti teorici di base del problema, uno spiega la tecnica di risoluzione del problema che è il Reinforcement Learning, con alcuni semplici esempi di impiego, l'altro è lo strumento responsabile per generare la politica che tratta gli stati per regione e attribuisce i valori d'utilità: il Fitted Q-iteration (FQI) con gli extremly randomized trees (extra-trees).

Gli atri capitoli costituiscono la parte pratica, in cui è definito il problema, sono creati i bots e in seguito vengono validati.

Capitolo 2

Reinforcement Learning

L'apprendimento per rinforzo è un insieme di tecniche di apprendimento molto simili a quelle usate dagli animali e dagli esseri umani, basate su un approccio a tentativi-errore. L'idea base è l'interazione con l'ambiente, dove a ogni azione corrisponde una transizione di stato e l'ottenimento di una ricompensa. Esistono diversi algoritmi di scelta delle azioni che permettono di bilanciare lo sfruttamento dell'esperienza acquisita con la necessità di esplorare nuove possibilità.

L'idea base è l'iterazione con l'ambiente, ogni azione porta a differenti stati, benefici o no. Ci sono diversi algoritmi di scelta delle azione che permettono di attingere un oggettivo più velocemente.

I termini principali che vengono usati in questo tipo di approccio sono: politica,funzione ricompensa e funzione di valore. Con ambiente invece indichiamo il sistema, o se più comodo, un modello di questo, con cui l'agente interagisce.

Per chiarire i termini, consideriamo il seguente esempio:

Un guidatore per andare da casa all'università deve scegliere tra diverse strade possibili, i criteri posso essere vari, come il tempo speso e/o il consumo di benzina. L'insieme delle possibili strade costituisce il suo ambiente. Ogni via presa è una azione, che determina un tempo di percorrenza diverso. Se il tempo è il criterio che il guidatore vuole ottimizzare, allora la ricompensa potrebbe essere l'opposto del tempo richiesto.

In alcuni casi potrebbe non essere una buona azione scegliere la via più veloce, perché la successiva potrebbe essere molto lenta e quindi sarebbe stato meglio scegliere due vie a velocità normale; a questa idea viene associato il concetto di valore. La politica è ciò che determina il comportamento dell'agente nei diversi stati. Obiettivo dell'apprendimento per rinforzo è trovare la politica con il massimo valore.

Valore è un concetto molto importante, che considera gli effetti futuri di una politica mente la ricompensa porta soltanto a un risultato istantaneo. Una via molto lenta che porta a vie veloci, avrà una ricompensa scarsa, ma il suo valore probabilmente sarà elevato se queste vie a loro volta ci condurranno ad altre vie veloci.

A ogni politica è possibile associare un valore valutando le ricompense che permette di ottenere lungo un certo orizzonte temporale.

Una possibile rappresentazione dell'ambiente:

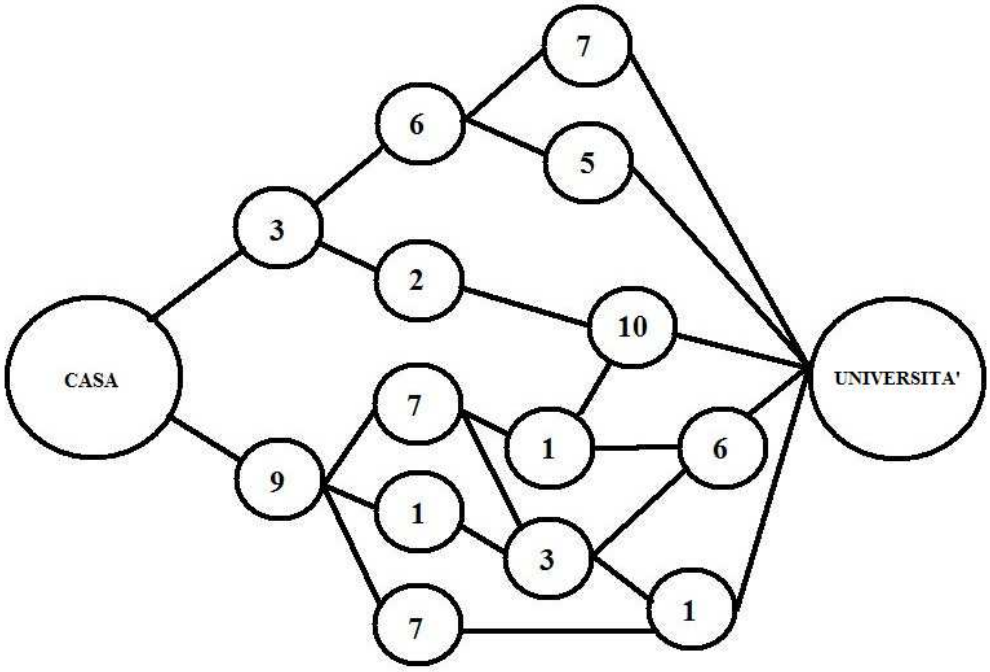

Figura 2.1. Modello del guidatore casa università.

In Figura 2.1 è visualizzato l'ambiente dell'esempio del guidatore. Ogni circolo rappresenta un locale, casa, università o le vie, i numeri sono i tempi necessari a percorre ogni via. Un buon esercizio per iniziare a capire i concetti che saranno spiegati più avanti è provare a trovare qual è il miglior camino.

L'approccio può essere ancora più complesso, come in qualunque sistema reale, più informazione si considera, migliori risultati si possono ottenere. Possiamo considerare informazioni addizionali relative al tempo speso in ogni via, quali: l'ora del giorno, il clima, gli incidenti, etc. Oppure, a causa di cambiamenti nel proprio sistema, una via lenta per problemi di pavimentazione può diventare veloce se il problema è stato risolto. D'altra parte, se il guidatore seguirà sempre lo stesso cammino non sarà in grado di accorgersi di tali cambiamenti.

In alcuni casi, nel sistema non esiste un inizio o un fine, soltanto cammini ciclici o anche infiniti, però l'obbiettivo è sempre lo stesso, massimizzare i valori, cioè, l'intensità media delle ricompense.

2.1 Tecniche di esplorazione e base matematica.

In alcuni modelli, l'azione non definisce sempre il prossimo stato, e neanche la sua ricompensa è sempre deterministica. In tali problemi è necessario ripetere più volte le stesse azioni per capire la dinamica del sistema.

Il poker è un problema esattamente di questo tipo. La ricompensa non è deterministica, poiché dipende delle carte che saranno girate, così come il prossimo stato, che dipende dalle azioni dell'altro giocatore. Dato che in questo progetto è stato usato un database di partite, abbiamo sfruttato algoritmi di batch-learning che non richiedono di definire meccanismi di esplorazione.

Se l'apprendimento fosse di tipo on-line, le tecniche di esplorazione avrebbero l'obiettivo di capire la logica degli avversari o se stanno utilizzando una strategia tempo variante.

Alcuni esempi di esplorazione sono utili per capire la logica del apprendimento per rinforzo e possono dare spunto per un futuro sviluppo della tesi.

L'esempio classico 'Armed-Bandit' esemplifica il concetto: (ricompensa non deterministica)

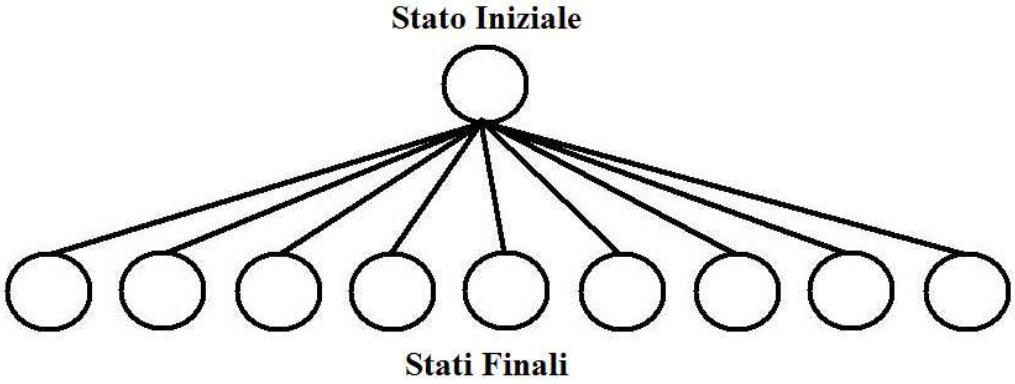

Figura 2.2. Problema Armed-Bandit

Soltanto una scelta deve essere fatta, il valore degli stati non è conosciuto, c'è del rumore, il valore delle ricompense può variare. Dopo una prima esplorazione generale nell'ambiente è possibile avere una prima stima dei valori, che per un'unica scelta è lo stesso della ricompensa.

Sia dato un unico stato terminale: in una prima volta si ottiene una ricompensa di 2.3, nella seconda 1.8 e nella terza 1.9. Allora, si può stimare che il valore dello stato è mediamente 2.

La domanda è: come procedere per massimizzare la ricompensa media? La risposta non è semplice. Si può, per esempio scegliere sempre la

stessa azione, quella considerato di maggior valore, però il rumore può trarci in inganno: si crede di aver individuato l'azione migliore, quando in realtà no lo è.

Quindi, una prima immediata conclusione è: si deve sempre aggiornare il valore delle azioni e degli stati, ogni volta che vengono esplorati.

Scegliere sempre l'azione con maggiore valore è conosciuto come metodo greedy. Un'altra opzione è la strategia ε-greedy, nel quale esiste una probabilità ε di scegliere un'azione sub-ottima per permettere di stimarne valori, cambiando la scelta quando si vede che lo stato greedy non è più il migliore.

Per questo esempio, si suppone di avere 10 stati, a ognuno è stato attribuito un valore fisso sconosciuto (una funzione random che ha generato numeri con media zero e varianza uno), e si vuole stimare questi valori. Il rumore introdotto nel sistema è dello stesso tipo, cioè, ogni volta che si va in uno stato si aggiunge una gaussiana di media zero e varianza uno. (i codici degli esempi di questo capitolo solo disponibile nell'Appendice A).

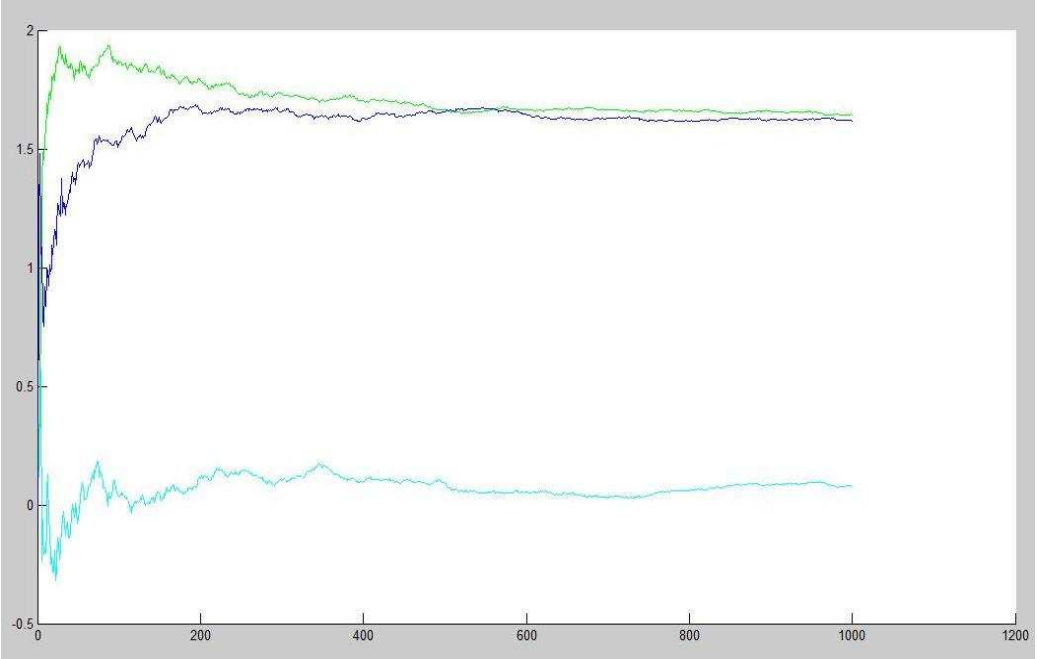

Figura 2.3. Ricompensa media di esplorazione.

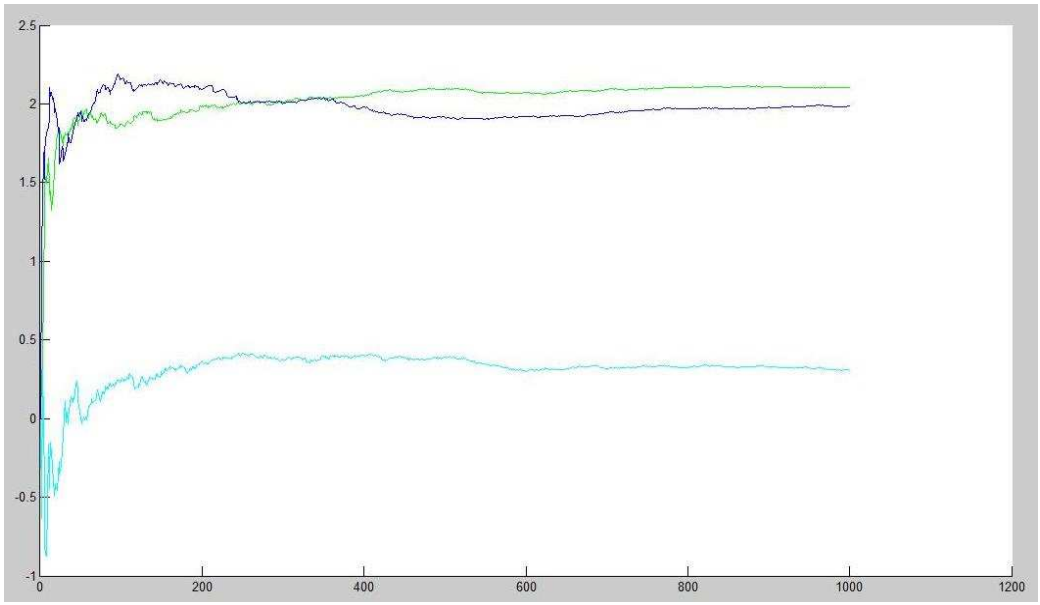

Figura 2.4. Ricompensa media di esplorazione.

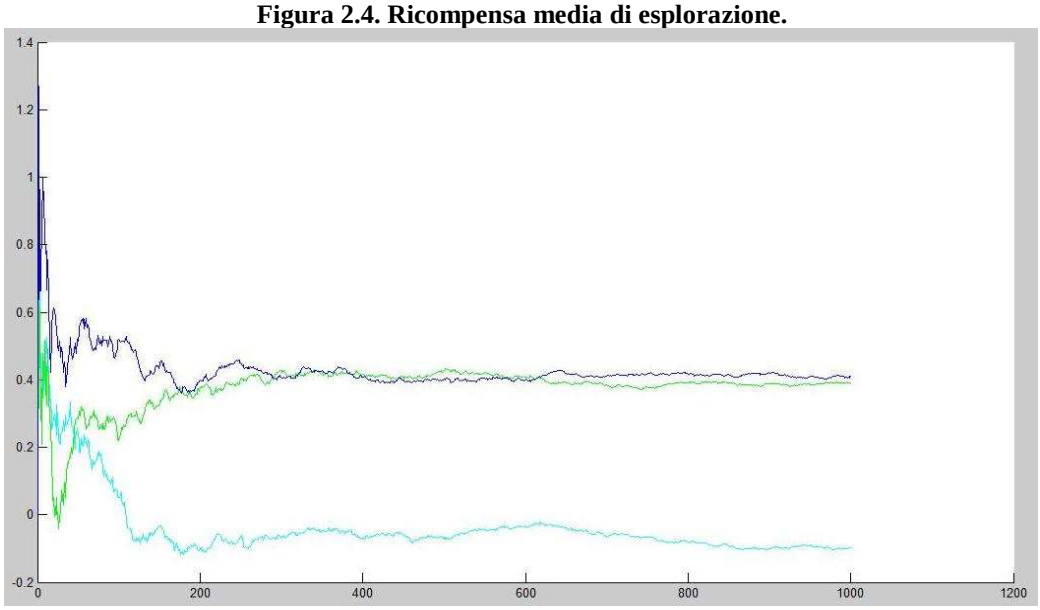

Figura 2.5. Ricompensa media di esplorazione.

L'asse delle ascisse rappresenta la quantità di giocate e quello verticale la ricompensa media. La linea verde è il metodo ε-greedy, con ε = 0.01, quella azzurra con ε = 0.1 e la blu corrisponde alla scelta random.

Non esiste un valore di ε che sia meglio in assoluto (il metodo random è stato messo soltanto per avere un attributo in più di confronto). Se ε è piccolo, la velocità con cui si scopre lo stato di maggior valore può essere bassa, però una volta trovato le oscillazione sono minori, le ricompense tendono a

mantenersi elevate, tra l'altro, per un valore di ε elevato, velocemente si individuano i buoni stati, però l'oscillazione è grande, e alla fine genera un ricompensa media minore.

Un fattore importante in qualunque esplorazione è il tasso di apprendimento, cioè, il grado di importanza della nuova ricompensa in relazione alle ricompense precedenti. Questa dipende molto dal sistema considerato, quelli più dinamici (le proprietà sono tempo varianti) devono a priori avere tassi maggiori di quelli più statici. Come sempre esiste un trade-off, maggior è il valore di questo parametro, maggior è l'effetto del rumore e la politica può diventare instabile e non efficace. Per il codice appena presentato, il tasso di apprendimento considerato è stato quello più intuitivo, una media pesata delle medie prima con la nuova ricompensa.

Di seguito si definiscono alcune basi matematiche, che permettono di chiarire questi concetti e facilitare l'introduzione di argomenti più astratti.

Definizioni:

Q= valore del stato, r= ricompensa, t= tempo, k= numero di esplorazione,
a= scelta, azione.

Formulando il problema del armed-bandit, il valore medio del stato in funzione del tempo può essere visto come la media delle k ricompense prima:

$$Q_t(a) = \frac{r_1 + r_2 + \cdots + r_{k_a}}{k_a}.$$

Se è considerata anche la azione futura per aggiornare il sistema:

$$\begin{aligned} Q_{k+1} &= \frac{1}{k+1}\sum_{i=1}^{k+1} r_i \\ &= \frac{1}{k+1}\left(r_{k+1} + \sum_{i=1}^{n} r_i\right) \\ &= \frac{1}{k+1}\Big(r_{k+1} + kQ_k + Q_k - Q_k\Big) \\ &= \frac{1}{k+1}\Big(r_{k+1} + (k+1)Q_k - Q_k\Big) \\ &= Q_k + \frac{1}{k+1}\Big[r_{k+1} - Q_k\Big], \end{aligned}$$

In soma, per questa linea di ragionamento, la nuova ricompensa è ponderata con quelle prima per trovare il nuovo valore dello stato, tutte sempre con la stessa importanza. In sistemi dinamici è meglio considerare come più importante gli ultimi dati, poiché possono ripresentare la situazione attuale, cosi invece di avere lo stesso peso attribuiti a tutti i dati, si può avere il fattore α:

$$Q_{k+1} = Q_k + \alpha\Big[r_{k+1} - Q_k\Big]$$

Che, come già, abbordato, è il fattore di apprendimento.

Esistono ancora diversi metodi di esplorazione, tutti hanno i suoi attribuiti scelti basati nelle caratteristiche del sistema scelto, precisione desiderata e/o capacità computazionale e la abilità del programmatore in modellare il sistema di una forma adeguata.

Un altro metodo interessante e importante è il min-max.

Min-Max

Nelle esplorazione iniziale col metodo ε -greedy, si può notare che alcuni stati sempre portano a ricompense migliori delle altre. Per esempio, dopo aver fatto quattro visito in ognuno dei tre stati A, B, C si sono ottenuti i seguenti valori di ricompensa:

A: 21, 18, 17 13
B: 25,18,10,21
C: 1,2,0,5

Si può dedurre che non ha molto senso esplorare gli stati A,B,C con la stessa frequenza, C può crescere, ma se si considera sistemi non estremamente dinamici, perché C possa raggiungere i valori di A,B, sarà necessario un tempo molto maggiore di quella relativo alla scelta fatta nel sistema.

Ci sono diverse tecniche per determinare la probabilità delle scelte, la più conosciuta è quella relativa all'entropia, che utilizza la formula di Gibbs.

La probabilità di una scelta 'a' dentro n è fornita per:

$$\frac{e^{Q_t(a)/\tau}}{\sum_{b=1}^{n} e^{Q_t(b)/\tau}}$$

τ è la temperatura del sistema, con un alto valore le scelte tendono al random, per bassi tendono al metodo greedy. Adesso, un paragone tra i tre metodi già presentati, nello stesso problema del armed-bandit. (codice disponibile nel appendice A)

Il soft-max è quello in Rosso:

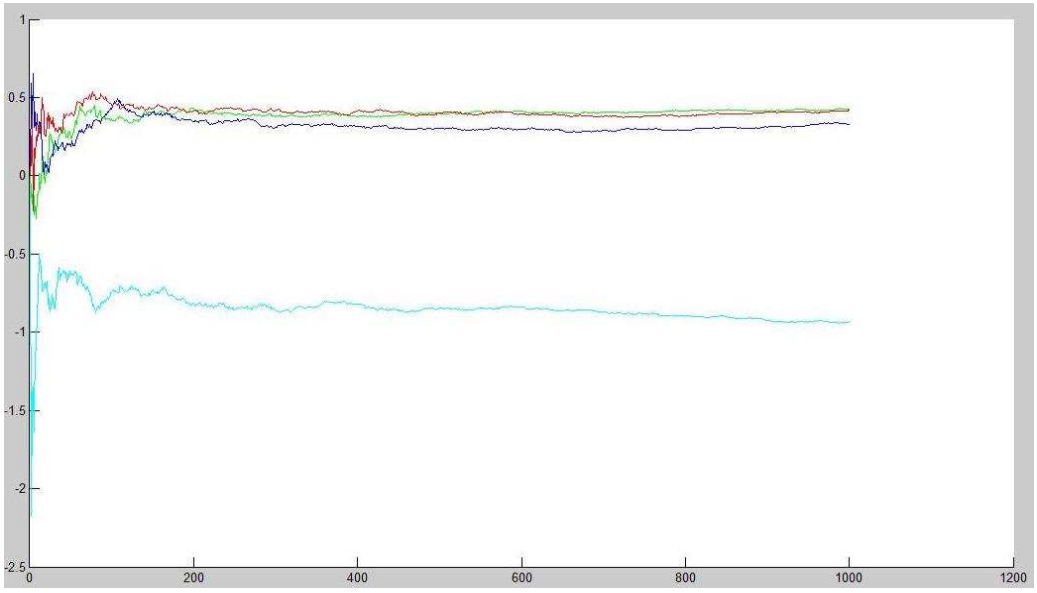

Figura 2.6. Ricompensa media di esplorazione.

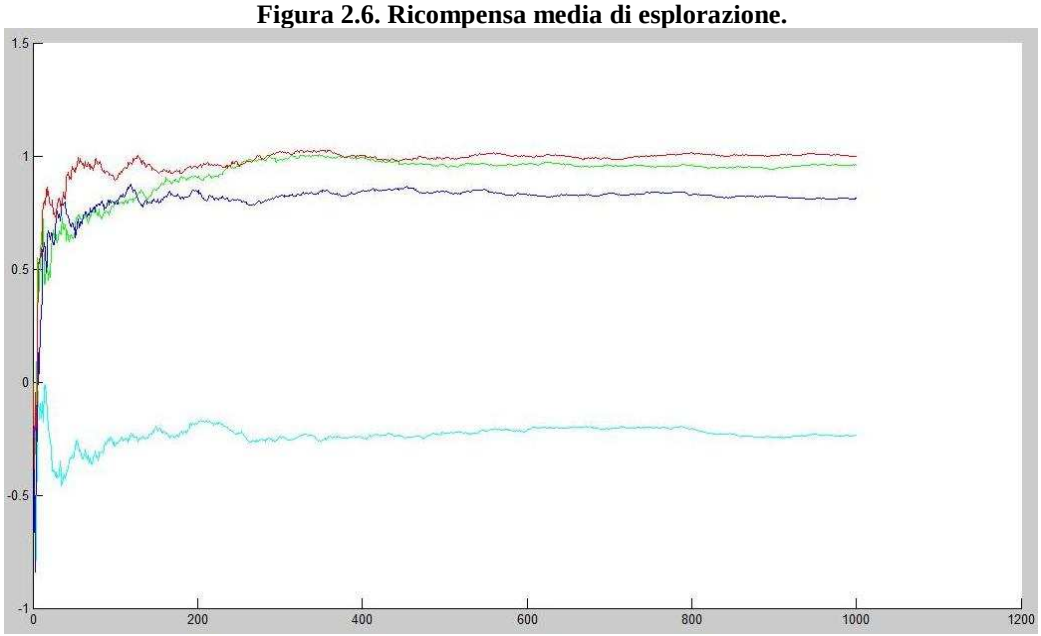

Figura 2.7. Ricompensa media di esplorazione.

La grande sfida per questo metodo è trovare la temperatura che se adatta al sistema, sia capace di fornire un'esplorazione efficace.

Esistono ancora diverse altre tecniche nel Reinforcement Learning. Le principale linee e l'idea base sono state presentate. Nel prossimo capitolo spiegheremo in che cosa consiste la tecnica di batch-learning Fitted Q-iteration che abbiamo usato in questo progetto.

2.2 Paragone col caso del poker

Dopo i concenti generali del Reinforcement Learnig, è necessario capire come questi saranno impiegati nel modello analizzato del progetto, il gioco Heads-up Hold'em.

Le definizioni, anche se semplici, sono fondamentali per capire la costruzione della logica per la definizione degli stati e la dinamica degli attributi.

Le azione possibili sono tre: fold, check/call o raise.

Il fold si può fare in qualunque momento della partita in cui si aspetta una risposta del giocatore. In alcuni si può scartare questa scelta, poiché il fold ha senso soltanto quando l'avversario in precedenza ha fatto un raise, cioè, fare un fold quando si può fare un check significa perdere le scommesse già fatte quando ancora esiste una possibilità di vincita.

Il check/call, che da adesso in poi sarà chiamato soltanto call, come il fold, può essere sempre scelto. In questo caso, il discorso è diverso, in una prima analisi, dopo un raise dell'avversario, si deve analizzare se ancora si può vincere il pot o no. Se non è possibile, l'azione giusta è il fold, poiché il call implicherebbe di aggiungere monete nel tavolo o ancora peggio sarebbe il raise (più monete perse).

Il raise, che per il caso limit, significa mettere uno big blind in più dell'avversario nel tavolo (più precisamente: uno nel pre-flop e flop, due nel turn e river). Non può essere sempre fatto, esistono vincoli nel gioco, nel pre-flop il massimo numero di rounds di scommesse è quattro e negli altri stadi è cinque.

Il significato dell'azione in una prima analisi è, il giocatore ha una buona mano e vuole aumentare il pot, per avere un maggior margine di guadagno. Se le azioni fossero basate soltanto sulla forza della mano(da adesso in poi chiamata probabilità di vincita), il modello sarebbe abbastanza semplice, ma dipende anche da tanti altri fattori. Per illustrare, si suppone un tavolo con due giocatore A e B. Il giocatore A tiene una mano debole, con una probabilità di vincita bassa, lui sa che il giocatore B è tight, cioè va fino in fondo (lo

showdown) solo quando ha una mano molto forte, allora può essere efficace fare il raise, poiché esiste la probabilità che l'avversario faccia fold. Un altro esempio, nello stesso tavolo, il giocatore B si accorge che il giocatore A già ha capito che lui è tight, allora lui poi iniziare a fare raise, anche quando lui non ha una buona mano, per ingannare il giocatore A.

Si conclude che, le azione dipendono da diversi fattori e loropossono per sé alterare l'ambiente della partita.

Il valore di ogni stato dipende della ricompensa degli stati terminali, cioè, quando uno dei giocatori prende tutto il pot. Allora, un modello possibile è: non esistono ricompense intermedie, soltanto alla fine e così possono essere aggiornati i valori di tutti gli stati. E' un caso particolare, spesso, ogni azione porta una ricompensa istantanea.

Questo modello è stato il primo a essere utilizzato nel progetto. Anche se concettualmente giusto, richiede un maggior sforzo al FQI per apprendere la politica del gioco, cioè, più iterazioni per stimare i valori degli stati. Un altro modello delle ricompense è, ogni azione non terminale anche porta a una ricompensa; per esempio, un raise porta a una ricompensa negativa(uguale al numero al valore della scommessa appena fatta), anche un call, il check o il fold a una ricompensa di valore nullo.

L'ambiente nel caso del batch learning è determinato dalla base di dati disponibile e perciò dalle giocate che la hanno generata.

Capitolo 3

Batch Learning

Considera la seguente rappresentazione:

$$x_{t+1} = f(x_t, u_t, w_t) \quad t = 0, 1, \cdots$$

La funzione f definisce quale sarà il prossimo stato del sistema, dato lo stato attuale x_t, la azione u_t e la interferenza di un possibile disturbo w_t. Questa formulazione attribuita a periodo determinati di tempo è quella discreta, che è il caso del poker, che non presenta continuità. Cioè, per esempio, nella sequenza di azione call/raise/call/fold, non ha senso attribuire valori intermedi/continui a questi.

Ogni transizione temporale ci porta a una ricompensa r.

$$r_t = r(x_t, u_t, w_t)$$

La politica è μ, che nel batch learning è stazionaria. J^{μ}_{inf} è il ritorno sperato utilizzando la politica μ in uno orizzonte di tempo infinito e perciò ha il suo valore:

$$J^{\mu}_{\infty}(x) = \lim_{N\to\infty} \underset{t=0,1,\cdots,N-1}{\underset{w_t}{E}} [\sum_{t=0}^{N-1} \gamma^t r(x_t, \mu(x_t), w_t) | x_0 = x].$$

Dove γ* è il fattore di sconto, un valore fisso, sempre nel intervallo [0,1], che definisce il grado di importanza delle ricompense future sul valore dello stato attuale. Il valore utilizzato sarà 1, poiché l'orizzonte del problema nel poker è molto ben definito.

La equazione può essere riscritta come:

$$\mu^*(x) = \arg\max_{u\in U} Q(x, u)$$

μ* è la politica che garantisce ritorno medio massimo. Che dipende della costruzione della funzione Q, che dato lo stato attuale realizza sempre la azione che porta un altro stato di valore massimo tra quelli possibili.

Il database è definito come F, che nel nostro caso è la quadrupla: stato, azione, ricompensa e prossimo stato.

$$\mathcal{F} = \{(x_t^l, u_t^l, r_t^l, x_{t+1}^l), l = 1, \cdots, \#\mathcal{F}\}$$

Se nel nostro problema non ci fosse disturbi (la stocasticità delle carte) e tutti gli stati possibili fossero visitati al meno una volta, il problema sarebbe completamente risolvibile di modo esatto (equazione di Bellman). Cioè, la equazione Q sarebbe precisamente definita. Ma non è quello che succede, è necessario una soluzione alternativa, ce ne sono diversi, quella qui proposta è il FQI con il regressore EXTRA.

3.1 Fitted Q-Iteration

La determinazione della politica ideale data la funzione-Q, come vista nel capitolo in precedenza è molto semplice. Il problema per se è costituito della determinazione della funzione-Q.

Nel caso del poker limit heads up il numero di stati possibile è enorme (circa 10^{14}), allora alcune semplificazione di stati se ragionevole devono essere impiegati. Un esempio: invece di avere diversi attributi per le possibile combinazione di carte nel gioco, si può avere un'unica che è la probabilità di vittoria (1001 valore per esempio se ha una risoluzione di 0.1%), però sempre si deve analizzare il quanto la semplificazione onera il modello, un altro fattore è decidere quale attributi hanno importanza per la determinazione della giocata, per arbitare queste decisione esistono diverse tecniche, una di queste è la feature straction.

Anche con queste semplificazione il numero di stati è ancora grande e non stante alcuni di loro non sono stati mai esplorati per il dataset, allora ci deve essere un metodo capace di scegliere la migliore azione in questo stato, di modo a capire la continuità tra i valore involviti. Una tecnica capace è il FQI, che aggiusta, un 'fitted' iterando i valori della funzione Q.

3.1.1 Il metodo FQI

La funzione-Q è determinata per iterazione, nella prima volta del algoritmo è determinata la funzione Q_0, fino alla N-esima volta, in cui è trovato Q_n, il criterio di fermata varia, quello che sarà utilizzato per noi sarà la profondità delle giocate, cioè il numero massimo di giocate fino ad arrivare alla decisione di quale giocatore prende il pot, che sono 11.

Il perché del ragionamento è semplice, con almeno 11 iterazione si permette che abbia la correzione del valore di tutti gli stati. Per chiarire, con due iterazioni soltanto, gli stati immediatamente vicini sono aggiustati, con due ogni stato riesce a vedere due dopo e cosi via. Con 11 tutti i camini di stati riescono a vedere tutti gli altri, con tutti i valori aggiornati correttamente.

La descrizione del metodo è la seguente:

Inputs: a set of four-tuples $\mathcal{F}$ and a regression algorithm.
Initialization:
Set N to 0 .
Let $\hat{Q}_N$ be a function equal to zero everywhere on $X \times U$.
Iterations:
Repeat until stopping conditions are reached

- $N \leftarrow N+1$.

- Build the training set $\mathcal{TS} = \{(i^l, o^l), l = 1, \cdots, \#\mathcal{F}\}$ based on the the function $\hat{Q}_{N-1}$ and on the full set of four-tuples $\mathcal{F}$:

$$i^l = (x_t^l, u_t^l), \tag{12}$$
$$o^l = r_t^l + \gamma \max_{u \in U} \hat{Q}_{N-1}(x_{t+1}^l, u). \tag{13}$$

- Use the regression algorithm to induce from $\mathcal{TS}$ the function $\hat{Q}_N(x,u)$.

Figura 3.1. L'algoritmo FQI

3.2 Metodi che utilizzano le arbore come regressori

Il metodo FQI presenta dentro di se il metodo di regressione, se si ragiona che per la determinazione della funzione-Q è necessario un FQI*(il FQI senza il regressore) e una regressione, si capisce che il metodo FQI* è sempre uguale e di una struttura semplice, lui soltanto crea una mappa di inputs e outputs, basato nel dataset e nell'iterazione prima.

Allora, due caratteristiche interessante si possono notare:

Le iterazioni sono interamente indipendenti una dell'altra e per quello si possono usare regressori diversi e/o un altro database. L'efficienza di questa logica è discutibile, pero nel mondo teorico è consistente.

Il regressore non ha bisogno di essere un albero di regressione, ce ne sono altre opzione come le rete neurale. I vantaggi de questa scelta sono sposti nel prossimo topico.

.

3.2.1 Regressore con alberi

L'idea generale di questo tipo di regressore è dividere l'universo di stati in diversi insieme, di tale modo che, ogni uno presenta una determinata similarità.

In realtà per gli alberi, le insieme considerati sono i chiamati inputs del sistema, generati delle coppie (x^l_t,u^l_t), cioè stato-azione. Ogni uno porta a uno output, che è il valore associato al corrispondente stato, come già visto in precedenza abbiamo:

$$\begin{aligned} i^l &= (x^l_t, u^l_t)\,, \\ o^l &= r^l_t + \gamma \max_{u \in U} \hat{Q}_{N-1}(x^l_{t+1}, u)\,. \end{aligned}$$

L'oggettivo finale è trovare una funzione f(i)= Q(i), che dato uno input qualunque è capace di determinare il valore di suo stato.

La procedura generale di calcolo è: si prendi ogni insieme e si fa la media degli outputs e a tutti gli inputs è associata lo stesso output medio. In una prima analisi, può sembrare una approssimazione grossolana, ma in realtà non lo è. Questo processo è ripetuto diverse volte per lo stesso dataset, ogni volta con una divisione diversa(i criteri di divisione possono cambiare, ma non è strettamente necessario), cosi presenta diversi vantaggi: La funzione f(i) diventa più continua e meno soggetta alla presenza di possibile rumore, il metodo è capace di generalizzare qualunque regressione, non ha bisogno di vincoli parametrici come succede per esempio nelle rete neurale o feature vectors.

Gli inputs che non sono presenti nel dataset anche ricevono uno valore di ouput, che è lo stesso di quello associato al suo insieme di appartenenza, cioè presentano uno stessa similarità a quelli del dataset, allora possono ricevere la stessa previsione.

Alla fine la funzione-Q può essere definita come:

$$\hat{Q}_N(x,u) \quad = \quad \sum_{l=1}^{\#\mathcal{F}} k_{\mathcal{TS}}((x_t^l,u_t^l),(x,u))[r_t^l + \gamma \max_{u' \in U} \hat{Q}_{N-1}(x_{t+1}^l,u')], \quad \forall N > 0$$

Dove K_{TS} è la funzione che determina il grado di appartenenza di un determinato input:

$$k_{\mathcal{TS}}(i^l,i) = \frac{I_{S(i)}(i^l)}{\sum_{(a,b)\in\mathcal{TS}} I_{S(i)}(a)}$$

S(i) è uno insieme di inputs, $I_{S(i)}(i^l)$ ritorna uno se l'input in questione appartiene allo insieme o zero se no. Questa formula, che alla prima vista sembra complessa è molto semplice, se per esempio nella regione di i^l abbiamo 5 inputs diversi, il valore di k per questo i^l sarà: 1/5=0.2. Si deve fare molta attenzione che questa formula non considera gli stati non che non appartengono al dataset, pero come si vede nella definizione $k(i^l,i)$ questo valore dipende di tutti gli inputs, quelli del dataset o no, ma è nel seguente senso: la funzione k per se definisce tutti gli le insieme di tutti gli inputs, cioè, definisce le regione di dove gli inputs non esplorati si trovano; facendo cosi, loro possono ricevere lo output medio degli stati del dataset come il suo valore. In soma:

$$f(i) = \sum_{l=1}^{\#\mathcal{TS}} k_{\mathcal{TS}}(i^l,i) * o^l$$

Manca una spiegazione, quale sono i criteri che realizzano la divisione degli inputs e cosi la determinazione degli insieme. Ci sono diverse forme, tecniche per farlo, ogni uno definisce uno tipo albero di regressione diverso. Quella utilizzata nel progetto è il EXTRA.

3.2.2 Extremely Randomized Trees

Il metodo fa genere M alberi di regressione, ogni una utilizza tutto il dataset, ogni nodo è definito con un procedura parzialmente random. Sono generate randomicamente K direzione possibile di taglio per creare il nodo, poi c'è il criterio di selezione che è la parte deterministica del processo, si valuta lo score di ogni taglio, cioè, la sua qualità in dividere una regione in inputs simili. I nodi sono creati fin che si arriva al numero minimo di inputs in un foglio, N_{min}.

In soma, il EXTRA è definito per tre parametri M,K,N_{min}.

Lo score è definito come:

$$\text{Score}([i_j < t], \mathcal{TS}) = \frac{var(o|\mathcal{TS}) - \frac{\#\mathcal{TS}_l}{\#\mathcal{TS}} var(o|\mathcal{TS}_l) - \frac{\#\mathcal{TS}_r}{\#\mathcal{TS}} var(o|\mathcal{TS}_r)}{var(o|\mathcal{TS})}$$

i_j è uno input valutato sul suo attributo j, t è il valore definito randomicamente e TS è il traning set. TS_l corrisponde a quelli inputs del data set con i_j<t e quindi TS_r il complementare. Allora questa formula rappresenta una differenza della varianza della regione prima e dopo del taglio, se lo score vale uno la riduzione è stata la meglio possibile, cioè il taglio porta a due regione con varianza nulla.
Alla fine l'algoritmo è:

Build_a_tree($\mathcal{TS}$)
Input: a training set $\mathcal{TS}$
Output: a tree T;

- If

  (i) $\#\mathcal{TS} < n_{min}$, or

  (ii) all input variables are constant in $\mathcal{TS}$, or

  (iii) the output variable is constant over the $\mathcal{TS}$,

  return a leaf labeled by the average value $\frac{1}{\#\mathcal{TS}} \sum_l o^l$.

- Otherwise:

  1. Let $[i_j < t_j] =$ Find_a_test($\mathcal{TS}$).
  2. Split $\mathcal{TS}$ into $\mathcal{TS}_l$ and $\mathcal{TS}_r$ according to the test $[i_j < t]$.
  3. Build $T_l =$ Build_a_tree($\mathcal{TS}_l$) and $T_r =$ Build_a_tree($\mathcal{TS}_r$) from these subsets;
  4. Create a node with the test $[i_j < t_j]$, attach T_l and T_r as left and right subtrees of this node and return the resulting tree.

Find_a_test($\mathcal{TS}$)
Input: a training set $\mathcal{TS}$
Output: a test $[i_j < t_j]$:

1. Select K inputs,$\{i_1, ..., i_K\}$, at random, without replacement, among all (non constant) input variables.
2. For k going from 1 to K:
   (a) Compute the maximal and minimal value of i_k in $\mathcal{TS}$, denoted respectively $i_{k,min}^{\mathcal{TS}}$ and $i_{k,max}^{\mathcal{TS}}$.
   (b) Draw a discretization threshold t_k uniformly in $]i_{k,min}^{\mathcal{TS}}, i_{k,max}^{\mathcal{TS}}]$
   (c) Compute the score $S_k =$ Score$([i_k < t_k], \mathcal{TS})$
3. Return a test $[i_j < t_j]$ such that $S_j = \max_{k=1,...,K} S_k$.

Figura 3.2. L'algoritmo EXTRA

3.2.2.1 Esempio del Metodo EXTRA

Si suppone di avere gli stati determinati per soltanto due attributi $x=<i_1,i_2>$, i parametri della regressione K=2 e N_{min}= 2. Con la seguente mappa di inputs:

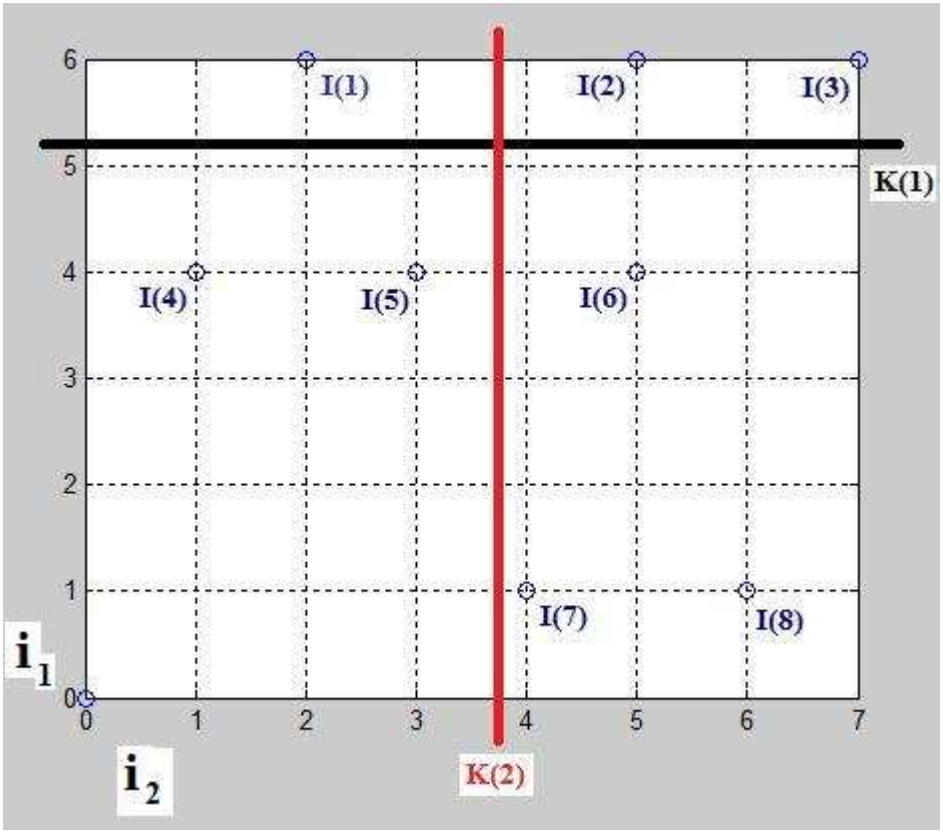

Figura 3.3. Inputs esempio per il metodo EXTRA

Ogni input rappresenta la coppia stato-azione, la azione corrispondente a ogni input è: Azione [I(1) I(2) I(3) I(4) I(5) I(6) I(7) I(8)]=
[call raise raise raise call call call raise]

Con i seguenti outputs:

Output	[I(1)	I(2)	I(3)	I(4)	I(5)	I(6)	I(7)	I(8)]=
	[3	6	?	9	?	4	2	1]

Come $N>N_{min}$ e gli outputs e inputs non sono costanti, si deve realizzare due corti (K=2) e determinare quale esegue la meglio divisione dello stato, per mezzo dello score.

Gli inputs I(3) e I(5), non appartengono al TS, pero sono possibili statiazione e devono ricevere un valore stimato di output.

La varianza di tutto il TS è: Var(o|TS)= 7.14.

Per K(1): $Var(o|TS(i_1<5.2))=$ Var[I(4) I(6) I(7) I(8)]=
Var[9 4 2 1]= 9.5
$Var(o|TS(i_1\geq 5.2))=$ Var[I(1) I(2)]=
Var[3 6]= 2.25

$$Score([i1<5.2],TS)=\frac{7.14-(\frac{4}{6})*9.5-(\frac{2}{6})*2.25}{7.14}=0.0079$$

Per K(2): $Var(o|TS(i_2<3.8))=$ Var[I(1) I(4)]=
Var[3 9]= 9
$Var(o|TS(i_2\geq 3.8))=$ Var[I(2) I(6) I(7) I(8)]=
Var[6 4 2 1]= 3.69

$$Score([i1<5.2],TS)=\frac{7.14-(\frac{4}{6})*3.69-(\frac{2}{6})*9}{7.14}=0.2353$$

Allora, il meglio taglio è il secondo, che permette un maggior riduzione della varianza. Attenzione, la varianza qui calcolata è quella basata su tutta la popolazione, non quella che considera soltanto alcune campioni.

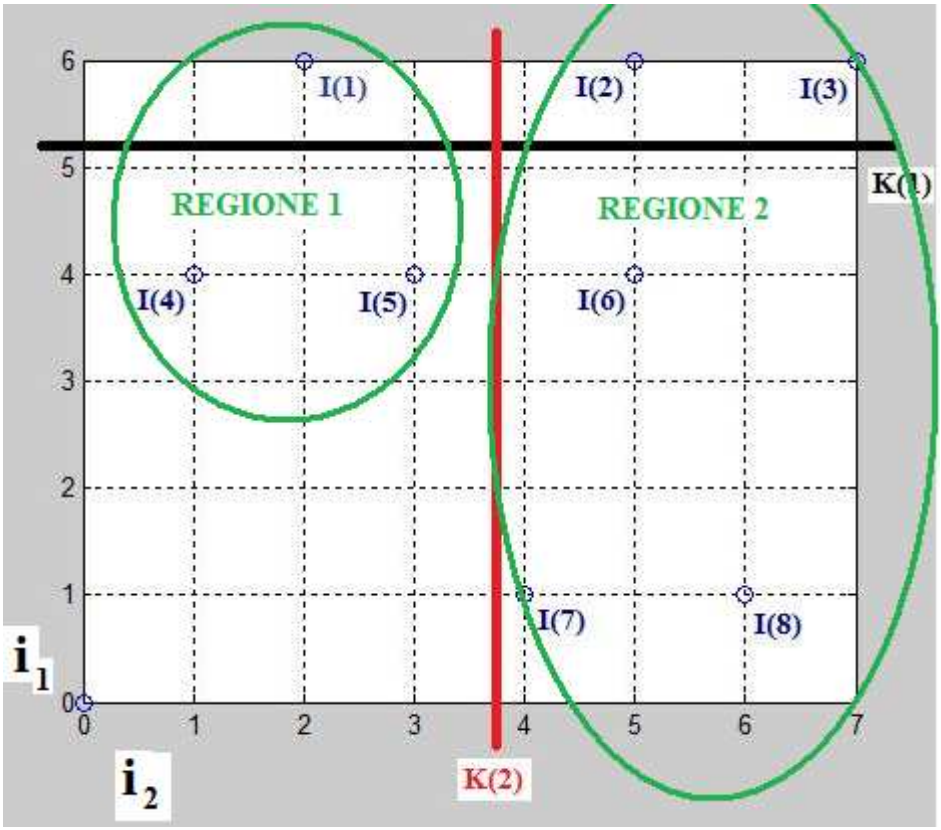

Figura 3.4. Scelta della direzione di taglio e determinazione delle regione

Il taglio divide il sistema in due regione, quello che si deve fare adesso è ugualizzare tutti gli outputs di ogni una. Cioè:

Regione 1

$Output_{nuovo}[I(1)]=Output_{nuovo}[I(4)]=Output_{nuovo}[I(5)]=$

$$\frac{Output[I(1)]+Output[I(4)]}{2}=6$$

Come si vede, il regressore già ha fatto una stimativa dello I(5); cioè, nello stato <4,3> con una azione del tipo call, si aspetta un output(il valore della azione) uguale a 6.

Regione 2

$Output_{nuovo}[I(2)]=Output_{nuovo}[I(3)]=Output_{nuovo}[I(6)]=Output_{nuovo}[I(7)]=Output_{nuovo}[I(8)]=$

$$\frac{Output[I(2)]+Output[I(6)]+Output[I(7)]+Output[I(8)]}{4}=3.25$$

Per I(3); cioè, nello stato <6,6> con una azione del tipo raise, si aspetta un output(il valore della azione) uguale a 6.

Per la regione uno la regressione è finita, poiché il suo numero di elementi corrispondente al TS è uguale a Nmin, già nella seconda ancora si deve ripetere l'algoritmo.

Ci possono essere ancora diversi tante alberi, d'accordo con il suo valore scelto M, quindi tanti altri tipi di tagli possono essere fatti e alla fine l'output di ogni stato-azione è la media di tutti i fogli calcolati con questo algoritmo.

Capitolo 4

Modellando il poker

Il primo punto a capirsi sono i vincoli del sistema, le regole.

Poi sono stati determinati tutti gli attributi che definiscono univocamente il sistema:

0-1: Prima carta nella mano
2-3: Seconda carta della mano
4-9: Carte Floop
10-11: Carta Turn
12-13: Carta River
14-17: Scommesse Pre-Floop
18-22: Scommesse Floop
23-27: Scommesse Turn
28-31: Scommesse River

Figura 4.1. Attributi elementari

Ogni carta utilizza due attributi, uno per il suo numero e l'altro per il suo segno. Come nel Floop sono girate 3 carte, ci sono 6 variabile per questa situazione.

Le scommesse sono rappresentate del seguente modo, ogni attributo rappresenta la giocata di ogni uno giocatore, che è un valore trinario, 0 per fold, 1 per check/call e 2 per raise. Esempio:

Scomesse Pre-Floop: 1221. Che può essere tradotto come:

Small blind, che gioca prima, fa il check, poi il Big Blind fa un raise, lo SB risponde con un altro raise e il BB fa il call.

Le azione che non sono ancora realizzate nelle scommesse ricevono il valore di zero. Per esempio scommessa Pre-Floop: 1000, il SB ha fatto call e il BB deve ancora giocare. Un dettaglio, può sembrare che cosi non si sa quando l'altro giocatore ha fatto un fold, però in realtà si sa, perché ogni giocatore ha la sua volta di azione, se è arrivato 1000 al BB, lui deve giocare, se è arrivato 1000 al SB, lui non deve giocare e sa che il BB ha fatto un fold.

Si possono creare anche attributi ausiliari, con l'oggettivo di esprimere di un modo più semplice le informazione contenute precedentemente, che dopo saranno combinati per definire lo stato. Tale combinazione di attributi ridondanti con non ridondanti deve essere analizzato con cautela e se necessario la realizzazione di tecniche come la feature extraction.

Gli stati ausiliari sono: (creazione arbitraria, basata in classificazione spesso usata in libri di tecniche di poker, potevano essere diverse)

32: Prima carta mano
33: Seconda carta mano
34-36: Carte Floop
37: Carta Turn
38: Carta River
39: Stage
40: Raises
41: RaisesSB
42: Raise Prima
43: Turno
44: PV
45: Pot
46: Scomesse Pre-Floop
47: Scomesse Floop
48: Scomesse Turn
49: Scomesse River

Figura 4.2. Attributi aggiunti

Gli attributi delle carte dicono lo stesso di due attributi in un unico numero che va da 1 a 52, cioè, tutte le carte possibile.

Lo stage è:

0- La azione a essere eseguita è nel Pre-Floop
1- Nel Floop
2- Nel Turn
3- Nel River

Raises- il numero totale di Raises di entrambi giocatori.

RaisesSB- il numero di raises del SB. (queste due informazioni permettono di calcolare il valore di RaisesBB)

Raise prima- se 0, non è stato eseguito un raise immediatamente prima di questo stato, se 1 si. Cioè, se vale zero il minimo a fare è il check, se uno il minimo a fare è il fold.

Turno- 0:deve giocare il BB; 1: deve giocare lo SB.

PV- Probabilità di vittoria, supponendo che il giocatore avversario abbia una mano random e anche tutte le prossime carte che saranno girate.

Pot- quantità totale di monete già scommesse nel tavolo.

Scommesse- adesso, ogni stage di scommessa è rappresenta con un unico numero.(vedere appendice B per più dettagli)

Il dataset (giocate già realizzate da dove sarà estratta la analisi- batch learning) presenta il seguente formato:

```
GGValuta|HyperboreanLimit-Eqm:0:rc/crc/cc/rc:8hTd|As9c/KsJcJh/Qc/8d:10|-10
HyperboreanLimit-Eqm|GGValuta:1:rc/crf:5d6s|6dKd/ThTdJs:-4|4
GGValuta|HyperboreanLimit-Eqm:2:rrc/rc/cc/rc:JhQh|5dAc/4c4s9c/7h/8c:-12|12
HyperboreanLimit-Eqm|GGValuta:3:rrc/rrc/crf:QdTd|2h2d/Js2s3h/Jc:-10|10
GGValuta|HyperboreanLimit-Eqm:4:rrc/rc/cc/crc:Ts3s|AsKs/Td2dJd/5d/Ah:-12|12
```

Figura 4.3. Esempio di database.

In questo esempio ci sono 5 mani giocate, tra i giocatori GGValuta e HyperboreanLimit, il 'r' significa raise, 'c' call, 'f' fold. Dopo delle scommesse sono presentate le carte dei giocatori sempre nella stessa ordine, prima BB, poi SB, separate per '|'. Dopo le carte del tavolo, separate per '/' e alla fine la ricompensa presa di ogni giocatore, positiva al vincitore e nell'ordine BB|SB.

Nella mano zero, come esempio, lo SB con un Asso e un Nove fa un raise iniziale, il BB con un Otto e un Dieci fa il call. Nel floop è uscito un Re e due Jay, il BB check poi lo SB fa un raise e il BB fa il call. Cosi via, fino a arrivare al showdown, dove il BB prende 10 monete del SB.

4.1. Procedura per il FQI

Gli attributi scelti per la procedura del FQI sono:(già nella forma vettoriale impiegata nella programmazione, appendice B)

Stati[0]=stage;
Stati[1]=raises;
Stati[2]=raisesSB;
Stati[3]=raiseanterior;
Stati[4]=turno;
Stati[5]=PV;
Stati[6]=pot;
Stati[7]=betPre;
Stati[8]=betFloop;
Stati[9]=betTurn;
Stati[10]=betRiver;
Stati[11]=azione;

Figura 4.4. Figura per il FQI.

Il dataset deve essere trasformato in questa informazione di stato, si nota che l'elemento 11 è la azione che è una delle informazione necessari per il FQI, che è lo stato più azione, per costituire l'input.

Il programma che realizza il FQI, si utilizza della seguente struttura:

Stato attuale+azione+Prossimo stato+ricompensa.

Questa struttura è fatta mediante il programma contenuto nel appendice B.

Allora è necessario trasformare tutto il dataset in questo formato, prendiamo la seguente mano come esempio:

PLAYER1|PLAYER2:0:cc/rc/cc/crrrc:8sAs|Ac2h/8h4h2s/5c/6s:80|-80

Che sarà trasformata in tutti gli inputs: (stato-azione)

1:PLAYER1|PLAYER2:0::8sAs|Ac2h/8h4h2s/5c/6s:80|-80:c$
0:PLAYER1|PLAYER2:0:c:8sAs|Ac2h/8h4h2s/5c/6s:80|-80:c$
0:PLAYER1|PLAYER2:0:cc/:8sAs|Ac2h/8h4h2s/5c/6s:80|-80:r$
1:PLAYER1|PLAYER2:0:cc/r:8sAs|Ac2h/8h4h2s/5c/6s:80|-80:c$
0:PLAYER1|PLAYER2:0:cc/rc/:8sAs|Ac2h/8h4h2s/5c/6s:80|-80:c$
1:PLAYER1|PLAYER2:0:cc/rc/c:8sAs|Ac2h/8h4h2s/5c/6s:80|-80:c$
0:PLAYER1|PLAYER2:0:cc/rc/cc/:8sAs|Ac2h/8h4h2s/5c/6s:80|-80:c$
1:PLAYER1|PLAYER2:0:cc/rc/cc/c:8sAs|Ac2h/8h4h2s/5c/6s:80|-80:r$
0:PLAYER1|PLAYER2:0:cc/rc/cc/cr:8sAs|Ac2h/8h4h2s/5c/6s:80|-80:r$
0:PLAYER1|PLAYER2:0:cc/rc/cc/crr:8sAs|Ac2h/8h4h2s/5c/6s:80|-80:r$
0:PLAYER1|PLAYER2:0:cc/rc/cc/crrr:8sAs|Ac2h/8h4h2s/5c/6s:80|-80:c$
2:PLAYER1|PLAYER2:0:cc/rc/cc/crrrc:8sAs|Ac2h/8h4h2s/5c/6s:80|-80

Figura 4.5. Database trasformato in inputs.

Il primo numero di ogni riga dice di chi è il turno, zero il BB, uno il SB. La parte finale :c$,r$,f$ dice la azione che è stata fatta nel determinato stato.

Che convertito nei numeri degli attributi, per esempio, per la prima riga è:

0	0	0	0	1	590	15	2	0
0	0	**1**	1	1	0	1	1	610
30	5	4	0	0	-5			

Figura 4.6 Struttura dei dati numerici

Fino a uno(in azzurro) è lo stato attuale e uno è la azione scelta, dopo è lo stato arrivato allo stesso giocatore dopo di questa azione (e quindi anche dopo una possibile giocata del avversario) e -5(verde) è la ricompensa per

la scelta attuale. In questo caso il blind vale dieci e per fare il call nella prima giocata è necessario mettere uno small blind, cioè 5 monete.

La logica di ricompensa di ogni giocata è della seguente forma: ogni volta di scommessa in cui si porta più monete nel tavolo la ricompensa è negativa, di valore uguale a la sua quantità. Quindi tutte le ricompense saranno o zero, quando si fa un check, o negativa, quando fa un call/raise, o positiva soltanto alla fine della mano in cui uno dei giocatore prende il pot. Due vincoli importanti: lo small blind in sua prima giocata per non lasciare la mano deve fare il call e avere una ricompensa di -0.5blinds o anche un raise, -1.5blinds. Il BB per iniziare a giocare non ha bisogno di mettere monete, quindi può iniziare con una ricompensa uguale a zero, ma poi quando prende le monete del pot si deve sottrarre il blind iniziale che lui ha messo.

Esistono momenti giusti nella partita in cui si può fare il fold, cioè, immediatamente dopo il raise del avversario o la prima giocata del SB. Sempre che nel dataset è stato realizzato una scommessa fuori di queste condizione la giocata è considerata per la formulazione del FQI, non c'è nessun problema in considerarla, poiché il proprio FQI identifica che non è una buona giocata e l'azione sarà scartata.

La trasformazione di stati in numeri quantificati è la parte più delicata del processo, poiché non deve contenere nessun errore e per quello si deve avere una programmazione molto precisa, tra l'altro esiste il problema di tempo di calcolo computazionale. La determinazione del PV può essere molto onerosa, circa 0.5sec per ogni uno dei casi, se per esempio sono giocate 10000 mani, in ogni una si fa la valutazione del PV circa 8 volte (per entrambe giocatore), allora il tempo totale di calcolo sarà circa 11 ore. Un'altra opzione è l'utilizzo di un altro database che contenga già valore calcolati previamente della PV, che sarà quello maggiormente utilizzato.

Allora si può realizzare il FQI.

Capitolo 5

Validazione della metodologia

In linee generale per validare il metodo è stata realizzata la seguente procedura: prima sono stati creati due database di giocate, uno con due giocatori random e l'altro con un giocatore random contro un bot già esistente. Gli database sono utilizzati per l'algoritmo FQI dove sono stati creati nuovi giocatore, bots del progetto, che saranno chiamati di *Heros*, per non far confusione con altri bots. Gli Heros sono alla fine confrontati contro un giocatore random e il bot.

5.1 Primo caso: giocatore random

Il bot random creato non è assolutamente random, cioè, le azione fold/call/raise non hanno la stessa probabilità, ma invece segueno: 10% fold, 50% call e 40% raise. Questi valore in prima ordine imitano i volumi di giocate realizzate per essere umani e anche si permette che ogni tanto si arrive al showdown; importante perché cosi si può analizzare bene tutta la possibile evoluzione del gioco, se ci fosse 33.3% di probabilità di fold le mani giocate finirebbero tropo velocemente.

Per creare il Hero adatto a giocatore contro il bot random si devono prima capire diverse informazione, come quante giocate di dataset devono essere analisate e tutti i parametri del metodo: M-numero di alberi, K-numero di tagli e Nmin.

Si deve anche analizzare qual' è un buon valore iniziale per iniziare le prove:

Quanto a M non c'è segreto, maggiore meglio è, si construiscono più alberi, cosi ci sarà un risultato più consistente, poiché il processo di costruzione degli alberi è parzialmente random e cosi si permette di fare diversi modo di taglio, o sia, diverse spiegazione degli stati. L'unica limitazione è il tempo computazionale che cresce linearmente con questo valore.

Il numero di mani del database è simile a M, maggiore informazione di come si gioca meglio è, ma dopo un certo punto l'informazione passano a soltanto ripetere, allora c'è una limitazione pratica.

K e Nmin, sono già valori più delicati, poiché con lo stesso volume di informazione possono creare azione del Hero di modo overfitted o

underfitted. Di modo generale K grande o Nmin piccolo si crea overfitting e K piccolo o Nmin grande si crea underfitting.

Il valore iniziale del database sarà 5000 giocate, con queste valore si crede che il contenuto di informazione sia già sufficiente. Nella computer poker competion[9], ogni partita è il risultato di 3000 giocate, se con questo valore di giocate i boots della competizione riescono mediamente a capire la logica del avversario, la logica random, che è banale, molto probabilmente sarà spiegata con un valore maggiore di giocate. Per verificare se questa ipotese è giusta, questa è confrontata con un altro hero creato con un database maggiore e non sono stati incontrati guadagni significativi.

Il numero M iniziale sarà 100, cioè ogni albero avrà 2% di importanza del risultato finale dell'azione, se si considera che si vuole avere una varianza di risultati minore di 5% di quella fatta con un numero infinito di alberi, tale numero è di ordine giusta.

Il K sarà quello consigliato per Ernst[2], che determina che il numero di K è uguale alla quantità di attribuiti, cioè su ogni analisi di taglio degli inputs, si farà uno e soltano uno taglio, su ogni attributo.

Nmin è quello più difficile di essere stimato, determinare la dimensione dei fogli del albero dipende della stocasticità del problema, che è il caso del poker, non deterministico, tra l'altro anche questa dimensione non è sempre uguale. Per esempio, forse nel pre floop è necessario una dimensione minore per spiegare ogni tipo di mano, giocata, etc; già nel turn la contribuzione di disturbi come le carte girate nel tavolo e la possibilità di blufs del avversario non permettono fogli cosi piccoli altrimenti si andrà in direzione a un overfitting. Il metodo proposto per Ernst, pero, che è quello utilizzato nel progetto, non considera variazione nel Nmin nella costruzione degli alberi, allora si deve trovare un valore che mediamente massimizza l'efficienza del Hero.(Si può anche concludere che il metodo può essere modificato, con una analisi della varianza interna a ogni foglio, per determinare se un prossimo taglio dividerebbe in due regione con proprietà diverse o no)

Con un test iniziale di 5-100-11-4 (miglia di giocate, numero di alberi, numero di tagli e Nmin) in 3000 giocate simulate tra hero e bot random si arriva a un pot totale del hero di 11925 monete e ovviamente a -11925 del bot random, cioè, il hero a preso questa quantità del bot random. Col blind di 10, si conclude che il hero ha vinto mediamente 11925/10/3000 blinds per mano, o sia 3975 blinds/mano.

Proseguendo con lo stesso ragionamento si fa un test con 5-100-11-10 e se ottiene un pot di 14865, si osserva un guadagno se paragonato a quello di prima. Con 5-100-11-20 si ottiene 23335. Si può concludere che con un Nmin

maggiore fin'ora migliore i risulta, poiché i fogli erano tropo piccoli per in problema e portavano a un probabile overfitting.

Con 5-100-11-30 23480 monete, con 5-100-40 22540. Che vuoi dire che una dimensione di foglio tra 20-40 non cambia i risultati di modo significativo, c'è una piccola varianza pero già è della ordine della varianza inerente al problema (l'analisi della varianza sarà anche fatta in questo capitolo).

Già con 5-100-11-100 -11150, che significa dire che con una dimensione cosi grande del foglio il hero non è capace di identificare un strategia vincente, è stato generato un modello underfitted insufficiente.

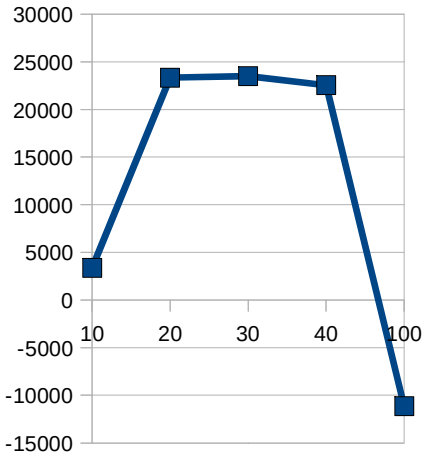

Figura 5.1 Grafico di guadagno medio del hero 5-100-11-Nmin contro il bot in funzione della variazione di Nmin.

Sono stati creati anche altri tipi di database, come per esempio uno con una feature straction arbitraria, dove sono stati selezionati gli atributi: [0]-stage, [4]-turno, [5]- Probabilità di vitoria, [6]- pot con la azione [11]. Il risultato sperato è uno simile uguale a quello di prima, poiché queste sono le informazione principale per giocare contro un giocatore random.

Ci sono stati i seguenti risultati: 5-100-11-20 -31645, 5-100-11-100 -25510, 5-100-11-400 16695, 5-100-11-800 20165, 5-100-11-1200 16955. E infatti per Nmin vicino a 800 c'è un risultato simile. È interessante notare che qui la dimensione del foglio è cambiata di ordine; la spiegazione è, prima c'erano tagli nei altri attributi che non erano fondamentali che pero facevano decrescere la dimensione del foglio, adesso che tutti i tagli sono più importante, una dimensione piccola porta al overfitting della soluzione. Si può anche concludere che, con una dimensione cosi grande il database di informazione è

molto maggiore del necessario, si possono arrivare a dimensione minore di foglio con minore dimensione di database senza pregidicare l'efficienza del hero.

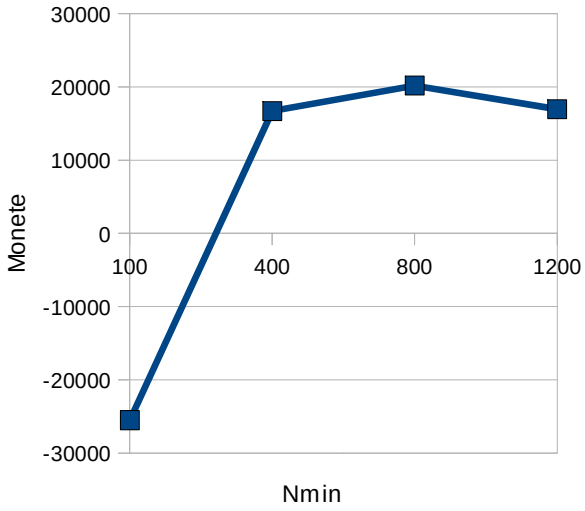

Figura 5.2 Grafico di guadagno medio del hero 5-100-11-Nmin con una feature straction forzata contro il bot random in funzione della variazione di Nmin.

È stato creato anche un altro database generato di 5 mila giocate random contro random come prima, più 5 mila giocate del hero contro il bot random, per creare un nuovo hero. Questa procedura è utilizzata in alcuni sperimenti per aggiungere efficienza al hero, cosi lui è capace di analizzare le giocate che lui sta, da vero, facendo contro il bot random e no soltanto le giocate simulate tra i bots random. Pero in questo caso non sono stati incontrati migliore significative, che vuoi dire che il hero di prima è stato capace di incontrare informazione sufficienti, senza aver bisogno di sua simulazione contro il bot random.

Ci sono due concetti che non sono stati esplorati fin'ora, uno è la costruzione esatta della probabilità di vittoria, i risultati qui presentati utilizzano per il calcolo del floop il metodo di monte carlo, che è stato poi migliorato, con la generazione di un database con i valori esatti per tutti le possibile combinazione del floop e ci sono incontrati risultati circa 25% superiori a quelli di prima. Il secondo concetto è il seed, cioè, in tutte le simulazione, in cui è stata creata la partita, le carte che sono girate sono pseudo-radom, cioè sono sempre le stesse carte per un stesso seed, allora si deve analizzare non soltanto il risultato per un unico seed, ma per un insieme.

Col calcolo esatto della probabilità di vittoria si ottiene, per esempio i risultati: 5-100-11-10 36825, 5-100-11-20 30550, 5-100-11-4 31175.

Esploriamo adesso i risultati per Nmin 10, col seed 33, per fare l'analisi della varianza dei risultati, cioè, fare lo stesso sperimento, hero contro il bot random nello stesso seed (non si spera un risultato uguale, anche se il hero giocherà della stessa maniera, il bot random non giocherà): 5-100-11-10 36825, 28630, 34595. Quindi la varianza del sistema è abbastanza considerevole.

Si può esplorare adesso la variazione del seed, anche per 5-100-11-10, col seed 34 si ottiene 38870, col seed 35 26880 e quello prima col seed 10 con 33300(media). Quindi, anche in questo caso, la varianza del sistema è abbastanza considerevole.

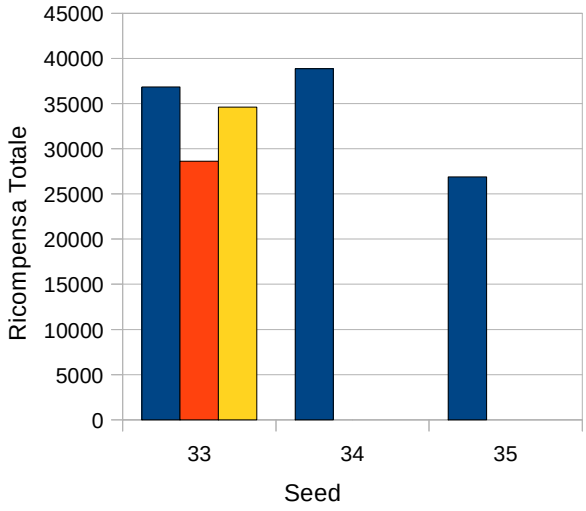

Figura 5.3 Grafico di guadagno medio del hero in funzione del seed, con 5-100-11-10.

Un'analisi dell'efficienza del hero con la variazione del numero di alberi(con seed 33) e anche della varianza dei risultati: 5-20-11-10 32110(primo sperimento) e 33410(secondo sperimento); 5-10-11-10 22060 e 19960, 5-1-11-10 3155 e 6195. O sia, la variazione di 100 alberi a 20 alberi non cambia i risultati di modo significativo, ma per valore come 10 e 1 le prestazione sono grandemente peggiorate. La varianza per un albero è enorme, visto che come la costruzione di ogni singolo albero è parzialmente random dipende in grande parte di come sono stati scelti i tagli random.

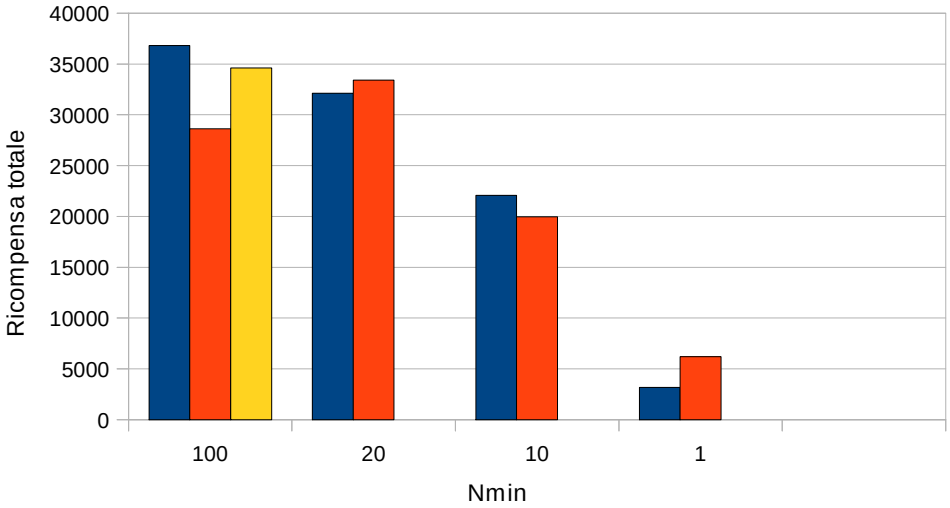

Figura 5.4 Grafico della prestazione del hero con variazione del numero di alberi M.

5.2 Secondo caso: Bot fornito del Poker Strategy

Adesso non si gioca più contro il giocatore random, ma contro un giocatore molto più furbo, che segue le strategie di gioco fornita per il sito: [10]pokerstrategy. Per creare il hero è stato utilizzato un database con 50000 mani, 50 alberi, 11 tagli come prima (sempre undici attributi) e Nmin che varia d'accordo col grafico. Sempre la ricompensa è calcolata da 3000 mani di gioco.

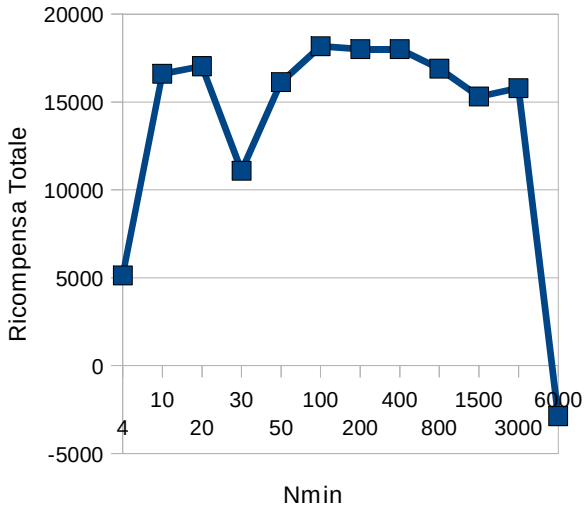

Figura 5.5 Grafico della ricompensa totale in funzione della variazione di Nmin contro il bot del Poker Strategy.

È interessante notare che l'algoritmo è capace di determinare politiche efficace per un'ampia variazione di Nmin da 10 a 3000 e con questa informazione già si può concludere che bastano poche informazione per capire il modo giusto di giocare, molto probabilmente pochi attributi di quelli messi per il hero sono importanti per la realizzazione del calcolo della politica efficace.

Allora, un'analisi importante è quella della importanza di ogni singolo attributo, perciò sarà costruito 11 heroes diversi, ogni uno senza utilizzarsi di uno dei tributi.

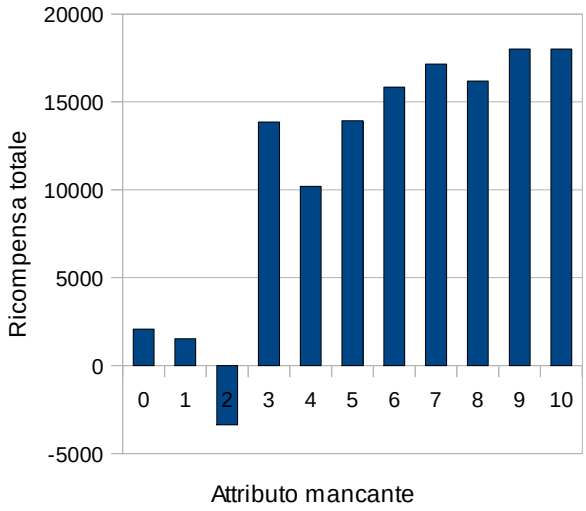

Figura 5.6 Grafico della ricompensa ottenuta in funzione del attributo mancante.

Ricordando che: Stati[0]=stage;
Stati[1]=raises;
Stati[2]=raisesSB;
Stati[3]=raiseanterior;
Stati[4]=turno;
Stati[5]=PV;
Stati[6]=pot;
Stati[7]=betPre;
Stati[8]=betFloop;
Stati[9]=betTurn;
Stati[10]=betRiver;
Stati[11]=azione;

Si nota che i primi 3 attributi sono fondamentali, stage, raises, raises SB e un po il turno.

Un fatto che può provocare stupefazione è la probabilità di vittoria con una bassa importanza, che può essere giustificato per al meno due motivi: con soltanto due giocatori, è meglio mantenersi nella mano anche con una probabilità di vittoria bassa; o l'informazione di raises del avversario è molto più importante per capire chi vincerà la partita.

Si osserva anche che l'ordine specifica di scommesse hanno una importanza molto bassa, si ricorda che ogni valore per un attributo di scommessa significa un ordine di scommessa diversa e maggiore numeri per questo non significano necessariamente più raises di uno o del altro giocatore, fatto che può creare difficoltà in un taglio giusto nel algoritmo per questo attributo. Una possibile soluzione sarebbe cambiare la rappresentazione di questi tributi. Pero, teoricamente con un database infinito, questa modifica non è necessaria, prima o poi ci sarebbero i tagli necessari per identificare queste aree diverse del attributo.

Capitolo 6

Conclusione

Il metodo FQI con alberi di regressione EXTRA si è dimostrato efficace per determinare una politica con un dataset del ambiente già fornito. La variazione di valori, come il numero di alberi e Nmin, per l'eseguizione del algoritmo e la costruzione del bot, porta a modificazione profonde nei risultati ottenuti e è fondamentale sua scelta opportuna. Nmin presenta una banda nella quale è valore incontrati sono giusti, se fuori banda porta a un under/over fitting.

In particolare per il caso di determinazione contro un giocatore random, ci sono stati presentati diverse formulazione e analisi di come si generano le prove, come variano i risultati in funzione del seed, del numero di alberi e di Nmin.

Nel confronto contro il bot del pokerstrategy si analizza l'importanza di ogni singolo attributo. La determinazione dello stato univoca dipende di diverse fattori, pero per la determinazione della politica efficace soltanto alcuni presentano una importanza rilevante, come lo stage, raises totali, raises di uno dei giocatori e di chi è il turno. La probabilità di vittoria contro il bot fornito del pokerstrategy non è stata considerata un attributo fondamentale, poiché la quantità di raises riusciva a spiegare meglio l'informazione di chi vince la mano.

Si suggerisce come lavoro futuro:

Una analisi più profonda di Nmin, di maniera che non sia costante, cercando di variare d'accordo con la varianza incontrata nel foglio;

Cambiare gli attributi osservati di modo opportuno, come per esempio cambiando la rappresentazione delle scommesse, con grandezze più continue nel senso di qui è il vantaggio, del small blind o del big blind;

Fare l'implementazione on-line del progetto, in un primo punto lui si utilizza del dataset già esistente e è capace di identificare modifiche nel ambiente di attuazione.

Bibliografia

[1] Andrew G. Barto (1998). Reinforcement Learning: An Introduction. MIT Press.

[2] D. Ernst, P. Geurts, and L. Wehenkel. Tree-Based Batch Mode Reinforcement Learning. Journal of Machine Learning Research 6 (2005) 503–556, April 2005. Springer-Verlag Heidelberg.

[3] http://home.dei.polimi.it/lazaric/?download=ThesisKnowledgeTransferInRL.pdf

[4] http://poker.cs.ualberta.ca/

[5] http://pokerstars.com/

[6] http://www.4kingpoker.com/article/58-poker_strategy_winning_players_style_tight_aggressive.html

[7] Agre, P. E. (1988).
The Dynamic Structure of Everyday Life.
PhD thesis, Massachusetts Institute of Technology, Cambridge, MA.
AI-TR 1085, MIT Artificial Intelligence Laboratory.

[8] Andreae, J. H. (1969b).
Learning machines--a unified view.
In Meetham, A. R. and Hudson, R. A., editors, Encyclopedia of Information, Linguistics, and Control. Pergamon, Oxford.

[9] http://www.computerpokercompetition.org/

[10] http://resources.pokerstrategy.com/Strategy/it/ps_fl_basic_handout_it.pdf

Appendice A

```
clear all
close all
clc
A= randn(1,10);
Q= A+randn(1,10);
tempQ= Q;
%Primo método- Scelta random
K= ones(1,10);
averagereward= zeros(1,1001);

for i=1:1000;
    play= randint(1,1,10)+1;
    K(play)= K(play)+1;
    reward= A(play)+randn;
    Q(play) = ((K(play)-1)*Q(play)+ reward)/K(play);
    averagereward(i+1)= (i*averagereward(i)+ reward)/(i+1);
end

%E-greedy
E= 0.01;
K= ones(1,10);
averagereward2= zeros(1,1001);
Q= tempQ;
for i=1:1000;
    if rand<E
        play= randint(1,1,10)+1;
    else [temp,ind]=max(Q);
        play = ind;
    end
    K(play)= K(play)+1;
    reward= A(play)+randn;
    Q(play) = ((K(play)-1)*Q(play)+ reward)/K(play);
```

```
    averagereward2(i+1)= (i*averagereward2(i)+ reward)/(i+1);
end
%E-greedy
E= 0.1;
K= ones(1,10);
averagereward3= zeros(1,1001);
Q= tempQ;
for i=1:1000;
    if rand<E
        play= randint(1,1,10)+1;
    else [temp,ind]=max(Q);
        play = ind;
    end
    K(play)= K(play)+1;
    reward= A(play)+randn;
           Q(play)   =   ((K(play)-1)*Q(play)+   reward)/K(play);
averagereward3(i+1)= (i*averagereward3(i)+ reward)/(i+1);
end
hold on
plot(averagereward,'c-')
plot(averagereward2,'g-')
plot(averagereward3,'b-')
```

```
%soft-max
T= 0.05; %temperatura del sistema
K= ones(1,10);
averagereward4= zeros(1,1001);
Q= tempQ;

for i=1:10;
```

```
    SM(i)= exp(Q(i)/T);
end
nSM= SM/sum(SM);


for i=1:1000;
    uniplay=1;
    gibbs= rand
    nSMprob(1)= nSM(1)
    for j=2:10;
        nSMprob(j)= nSMprob(j-1)+nSM(j);
    end
    for j=1:9;
        if gibbs<nSMprob(j)&& uniplay ==1
            play=j;
            uniplay=0;
        end
    end
    K(play)= K(play)+1;
    reward= A(play)+randn;
    Q(play) = ((K(play)-1)*Q(play)+ reward)/K(play);
    averagereward4(i+1)= (i*averagereward4(i)+ reward)/(i+1);
    for j=1:10;
    SM(j)= exp(Q(j)/T);
    end
    nSM= SM/sum(SM);
end
```

Appendice B

```
#include <iostream>
#include <fstream>
#include <string>
#include <stdio.h>
#include <string.h>
#include <vector>
#include <cstdlib>

using namespace std;

vector<int> calcstati(string str);


int calcrecompensa(int stage, int raiseanterior, int azione, int BB, int betPre, int ultimagiocata, string str, int turno){

int recompensa, found;
string temp;

if(ultimagiocata==0){
	if(raiseanterior==0&&azione==1) recompensa= 0;
	if(raiseanterior==0&&azione==2) recompensa= -BB;
	if(raiseanterior==1&&azione==1) recompensa= -BB;
	if(raiseanterior==1&&azione==2) recompensa= -2*BB;
	if(stage>1) recompensa= 2*recompensa;
	if(betPre==2&&azione==1) recompensa= -0.5*BB;
	if(betPre==2&&azione==2) recompensa= -1.5*BB;
                                                                        }
	else	{
		found= str.find_last_of("|");
		str= str.substr(0,found);
		found= str.find_last_of(":");
		str= str.substr(found+1);
		recompensa= 2*atoi(str.c_str())-BB;//due volte perche si perde prima per ogni azione, -BB perche non è stato strato il BB iniziale
		if(turno==1) recompensa= -recompensa+0.5*BB;//strazione iniziale è 1(SB) e non 2 come il BB
                                }
```

```
if(azione==0) recompensa= 0;

return recompensa;

                                        }


vector<int> calcstati(string str)
{
//string str ("0:GGValuta|HyperboreanLimit-Eqm:0:rc/crc/:8hTd|As9c/KsJcJh/Qc/8d:10|-10:c");
string cartasBB, cartasSB, cartasFloop,cartasTurn, cartasRiver, scomPre="0", scomFloop="0", scomTurn="0", scomRiver="0";
string temp, apostas, cartas;
vector <int> stati(12);  // stati is a vector of 11 integers
int found=-1;//trovatore di un carattere
int count1;
int raiseanterior=0;//verifica se la giocata prima è stato un raise
int raises=0; //totale fino alla giocata
int raisesSB=0; //raises del SB
int stage=0;
int turno;// 0-turno del BB 1-turno del SB
int BB=0;
int pot=0;
int betPre=0, betFloop=0, betTurn=0, betRiver=0; //dicono la stessa informazione che scomX, pero come numeri interi
int azione;
int PV=0;


BB= 10;//valore del big blind

found=str.find_first_of(":");  // get last position of ":" in str
str = str.substr (found+1);   // get from ":"+1 to the end

found=str.find_first_of(":");
str = str.substr (found+1);

//found=str.find("/",6);
found=str.find(":");
str = str.substr (found+1);
```

```
found=str.find(":");
apostas = str.substr (0,found);

found= str.find(":");
cartas = str.substr (found+1);


//Stage 0:PreFloop 1:Floop 2:Turn 3:River
temp= apostas;
for (count1=0;temp.find("/")!=-1;count1++){
        found= temp.find("/");
        if(count1==0) scomPre = temp.substr (0,found);
        if(count1==1) scomFloop = temp.substr (0,found);
        if(count1==2) scomTurn = temp.substr (0,found);
        temp = temp.substr (found+1); }
        if(count1==0) scomPre = temp.substr (0,found);
        if(count1==1) scomFloop = temp.substr (0,found);
        if(count1==2) scomTurn = temp.substr (0,found);
        if(count1==3) scomRiver = temp.substr (0,found);
stage= count1;


//cartasBB cartas SB
if(str.find(":|")!=-1)
        {
        found=str.find_first_of("|");
        cartasSB = str.substr (found+1,4);
        }
else
        {
        found=str.find_first_of(":");
        cartasBB = str.substr (found+1,4);
        }
if(str.find("|/")==-1)   {//caso in cui abbiamo entrambe carte
        found=str.find_first_of("|");
        cartasSB = str.substr (found+1,4);

                }

//raise prima
found= apostas.find_last_of("r");
raiseanterior= 0;
```

```
if(found== apostas.length()-1&&apostas!="")
        raiseanterior= 1;


//Raises
temp= apostas;
for (count1=0;temp.find_first_of("r")!=-1;count1++){
        found= temp.find_first_of("r");
        temp = temp.substr (found+1); }
        raises= count1;




//Carte Floop,Turn,River
if(stage>0){
        found= cartas.find("/");
        cartasFloop = cartas.substr (found+1);
        found= cartasFloop.find("/");
        cartasFloop = cartasFloop.substr (0,found);
                                                }
if(stage>1){
        found= cartas.find("/");
        cartasTurn = cartas.substr (found+1);
        found= cartasTurn.find("/");
        cartasTurn = cartasTurn.substr (found+1);
        found= cartasTurn.find("/");
        cartasTurn = cartasTurn.substr (0,found);
                                                }
if(stage>2){
        found= cartas.find("/");
        cartasRiver = cartas.substr (found+1);
        found= cartasRiver.find("/");
        cartasRiver = cartasRiver.substr (found+1);
        found= cartasRiver.find("/");
        cartasRiver = cartasRiver.substr (found+1);
        found= cartasRiver.find("/");
        cartasRiver = cartasRiver.substr (0,2);
                                                }
```

```
//turno
if(stage==0){
        if (scomPre.compare("") == 0) turno= 1;//if 0 they are equal
        if (scomPre.compare("c") == 0) turno= 0;
        if (scomPre.compare("r") == 0) turno= 0;
        if (scomPre.compare("cr") == 0) turno= 1;
        if (scomPre.compare("rr") == 0) turno= 1;
        if (scomPre.compare("crr") == 0) turno= 0;
        if (scomPre.compare("rrr") == 0) turno= 0;
        if (scomPre.compare("crrr") == 0) turno= 1;
                                                }
if(stage==1){
        if (scomFloop.compare("") == 0) turno= 0;
        if (scomFloop.compare("c") == 0) turno= 1;
        if (scomFloop.compare("r") == 0) turno= 1;
        if (scomFloop.compare("cr") == 0) turno= 0;
        if (scomFloop.compare("rr") == 0) turno= 0;
        if (scomFloop.compare("crr") == 0) turno= 1;
        if (scomFloop.compare("rrr") == 0) turno= 1;
        if (scomFloop.compare("crrr") == 0) turno= 0;
        if (scomFloop.compare("crrrr") == 0) turno= 1;
        if (scomFloop.compare("rrrr") == 0) turno= 0;
                                                }
if(stage==2){
        if (scomTurn.compare("") == 0) turno= 0;
        if (scomTurn.compare("c") == 0) turno= 1;
        if (scomTurn.compare("r") == 0) turno= 1;
        if (scomTurn.compare("cr") == 0) turno= 0;
        if (scomTurn.compare("rr") == 0) turno= 0;
        if (scomTurn.compare("crr") == 0) turno= 1;
        if (scomTurn.compare("rrr") == 0) turno= 1;
        if (scomTurn.compare("crrr") == 0) turno= 0;
        if (scomTurn.compare("crrrr") == 0) turno= 1;
        if (scomTurn.compare("rrrr") == 0) turno= 0;
                                                }
if(stage==3){
        if (scomRiver.compare("") == 0) turno= 0;
        if (scomRiver.compare("c") == 0) turno= 1;
        if (scomRiver.compare("r") == 0) turno= 1;
        if (scomRiver.compare("cr") == 0) turno= 0;
        if (scomRiver.compare("rr") == 0) turno= 0;
        if (scomRiver.compare("crr") == 0) turno= 1;
```

```
        if (scomRiver.compare("rrr") == 0) turno= 1;
        if (scomRiver.compare("crrr") == 0) turno= 0;
        if (scomRiver.compare("crrrr") == 0) turno= 1;
        if (scomRiver.compare("rrrr") == 0) turno= 0;
                                                }
```

```
pot= 2*BB;//valore sbagliato qui, ma è corretto poi, solo per simplificare la visualisazione
raisesSB=0;
        if (scomPre.compare("") == 0) {betPre=2; raisesSB= raisesSB+0; pot=pot-0.5*BB;}//
        if (scomPre.compare("f") == 0) {betPre=1; raisesSB= raisesSB+0; pot=pot-0.5*BB;}//attenzione a questo stato che è molto particolare
        if (scomPre.compare("c") == 0) {betPre=3; raisesSB= raisesSB+0; pot=pot+0;}//
        if (scomPre.compare("r") == 0) {betPre=4; raisesSB= raisesSB+1; pot=pot+1*BB;}//
        if (scomPre.compare("cc") == 0) {betPre=5; raisesSB= raisesSB+0; pot=pot+0;}
        if (scomPre.compare("cr") == 0) {betPre=6; raisesSB= raisesSB+0; pot=pot+1*BB;}//
        if (scomPre.compare("rf") == 0) {betPre=7; raisesSB= raisesSB+1; pot=pot+1*BB;}
        if (scomPre.compare("rc") == 0) {betPre=8; raisesSB= raisesSB+1; pot=pot+2*BB;}
        if (scomPre.compare("rr") == 0) {betPre=9; raisesSB= raisesSB+1; pot=pot+3*BB;}//
        if (scomPre.compare("crf") == 0) {betPre=10; raisesSB= raisesSB+0; pot=pot+1*BB;}
        if (scomPre.compare("crc") == 0) {betPre=11; raisesSB= raisesSB+0; pot=pot+2*BB;}
        if (scomPre.compare("crr") == 0) {betPre=12; raisesSB= raisesSB+1; pot=pot+3*BB;}//
        if (scomPre.compare("rrf") == 0) {betPre=13; raisesSB= raisesSB+1; pot=pot+3*BB;}
        if (scomPre.compare("rrc") == 0) {betPre=14; raisesSB= raisesSB+1; pot=pot+4*BB;}
        if (scomPre.compare("rrr") == 0) {betPre=15; raisesSB= raisesSB+2; pot=pot+5*BB;}//
```

```
        if (scomPre.compare("crrf") == 0) {betPre=16; raisesSB= raisesSB+1;
pot=pot+3*BB;}
        if (scomPre.compare("crrc") == 0) {betPre=17; raisesSB= raisesSB+1;
pot=pot+4*BB;}
        if (scomPre.compare("crrr") == 0) {betPre=18; raisesSB= raisesSB+1;
pot=pot+5*BB;}//
        if (scomPre.compare("rrrf") == 0) {betPre=19; raisesSB= raisesSB+2;
pot=pot+5*BB;}
        if (scomPre.compare("rrrc") == 0) {betPre=20; raisesSB= raisesSB+2;
pot=pot+6*BB;}
        if (scomPre.compare("crrrf") == 0) {betPre=21; raisesSB= raisesSB+1;
pot=pot+5*BB;}
        if (scomPre.compare("crrrc") == 0) {betPre=22; raisesSB= raisesSB+1;
pot=pot+6*BB;}

        if (scomFloop.compare("") == 0) {betFloop=2; raisesSB= raisesSB+0;
pot=pot+0*BB;}//
        if (scomFloop.compare("c") == 0) {betFloop=3; raisesSB= raisesSB+0;
pot=pot+0*BB;}//
        if (scomFloop.compare("r") == 0) {betFloop=4; raisesSB= raisesSB+0;
pot=pot+1*BB;}//
        if (scomFloop.compare("cc") == 0) {betFloop=5; raisesSB= raisesSB+0;
pot=pot+0*BB;}
        if (scomFloop.compare("cr") == 0) {betFloop=6; raisesSB= raisesSB+1;
pot=pot+1*BB;}//
        if (scomFloop.compare("rf") == 0) {betFloop=7; raisesSB= raisesSB+0;
pot=pot+1*BB;}
        if (scomFloop.compare("rc") == 0) {betFloop=8; raisesSB= raisesSB+0;
pot=pot+2*BB;}
        if (scomFloop.compare("rr") == 0) {betFloop=9; raisesSB= raisesSB+1;
pot=pot+3*BB;}//
        if (scomFloop.compare("crf") == 0) {betFloop=10; raisesSB=
raisesSB+1; pot=pot+1*BB;}
        if (scomFloop.compare("crc") == 0) {betFloop=11; raisesSB=
raisesSB+1; pot=pot+2*BB;}
        if (scomFloop.compare("crr") == 0) {betFloop=12; raisesSB=
raisesSB+1; pot=pot+3*BB;}//
        if (scomFloop.compare("rrf") == 0) {betFloop=13; raisesSB=
raisesSB+1; pot=pot+3*BB;}
        if (scomFloop.compare("rrc") == 0) {betFloop=14; raisesSB=
raisesSB+1; pot=pot+4*BB;}
```

```
        if (scomFloop.compare("rrr") == 0) {betFloop=15; raisesSB= raisesSB+1; pot=pot+5*BB;}//
        if (scomFloop.compare("crrf") == 0) {betFloop=16; raisesSB= raisesSB+1; pot=pot+3*BB;}
        if (scomFloop.compare("crrc") == 0) {betFloop=17; raisesSB= raisesSB+1; pot=pot+4*BB;}
        if (scomFloop.compare("crrr") == 0) {betFloop=18; raisesSB= raisesSB+2; pot=pot+5*BB;}//
        if (scomFloop.compare("rrrf") == 0) {betFloop=19; raisesSB= raisesSB+1; pot=pot+5*BB;}
        if (scomFloop.compare("rrrc") == 0) {betFloop=20; raisesSB= raisesSB+1; pot=pot+6*BB;}
        if (scomFloop.compare("crrrf") == 0) {betFloop=21; raisesSB= raisesSB+2; pot=pot+5*BB;}
        if (scomFloop.compare("crrrc") == 0) {betFloop=22; raisesSB= raisesSB+2; pot=pot+6*BB;}
        if (scomFloop.compare("crrrr") == 0) {betFloop=23; raisesSB= raisesSB+2; pot=pot+7*BB;}//
        if (scomFloop.compare("rrrr") == 0) {betFloop=24; raisesSB= raisesSB+2; pot=pot+7*BB;}//
        if (scomFloop.compare("crrrrf") == 0) {betFloop=25; raisesSB= raisesSB+2; pot=pot+7*BB;}
        if (scomFloop.compare("crrrrc") == 0) {betFloop=26; raisesSB= raisesSB+2; pot=pot+8*BB;}
        if (scomFloop.compare("rrrrf") == 0) {betFloop=27; raisesSB= raisesSB+2; pot=pot+7*BB;}
        if (scomFloop.compare("rrrrc") == 0) {betFloop=28; raisesSB= raisesSB+2; pot=pot+8*BB;}
//Turn & River Praticamente uguale al Floop, pero le scomesse sono doppiate
        if (scomTurn.compare("") == 0) {betTurn=2; raisesSB= raisesSB+0; pot=pot+0*BB*2;}//
        if (scomTurn.compare("c") == 0) {betTurn=3; raisesSB= raisesSB+0; pot=pot+0*BB*2;}//
        if (scomTurn.compare("r") == 0) {betTurn=4; raisesSB= raisesSB+0; pot=pot+1*BB*2;}//
        if (scomTurn.compare("cc") == 0) {betTurn=5; raisesSB= raisesSB+0; pot=pot+0*BB*2;}
        if (scomTurn.compare("cr") == 0) {betTurn=6; raisesSB= raisesSB+1; pot=pot+1*BB*2;}//
        if (scomTurn.compare("rf") == 0) {betTurn=7; raisesSB= raisesSB+0; pot=pot+1*BB*2;}
```

```
        if (scomTurn.compare("rc") == 0) {betTurn=8; raisesSB= raisesSB+0;
pot=pot+2*BB*2;}
        if (scomTurn.compare("rr") == 0) {betTurn=9; raisesSB= raisesSB+1;
pot=pot+3*BB*2;}//
        if (scomTurn.compare("crf") == 0) {betTurn=10; raisesSB= raisesSB+1;
pot=pot+1*BB*2;}
        if (scomTurn.compare("crc") == 0) {betTurn=11; raisesSB= raisesSB+1;
pot=pot+2*BB*2;}
        if (scomTurn.compare("crr") == 0) {betTurn=12; raisesSB= raisesSB+1;
pot=pot+3*BB*2;}//
        if (scomTurn.compare("rrf") == 0) {betTurn=13; raisesSB= raisesSB+1;
pot=pot+3*BB*2;}
        if (scomTurn.compare("rrc") == 0) {betTurn=14; raisesSB= raisesSB+1;
pot=pot+4*BB*2;}
        if (scomTurn.compare("rrr") == 0) {betTurn=15; raisesSB= raisesSB+1;
pot=pot+5*BB*2;}//
        if (scomTurn.compare("crrf") == 0) {betTurn=16; raisesSB=
raisesSB+1; pot=pot+3*BB*2;}
        if (scomTurn.compare("crrc") == 0) {betTurn=17; raisesSB=
raisesSB+1; pot=pot+4*BB*2;}
        if (scomTurn.compare("crrr") == 0) {betTurn=18; raisesSB=
raisesSB+2; pot=pot+5*BB*2;}//
        if (scomTurn.compare("rrrf") == 0) {betTurn=19; raisesSB= raisesSB+1;
pot=pot+5*BB*2;}
        if (scomTurn.compare("rrrc") == 0) {betTurn=20; raisesSB=
raisesSB+1; pot=pot+6*BB*2;}
        if (scomTurn.compare("crrrf") == 0) {betTurn=21; raisesSB=
raisesSB+2; pot=pot+5*BB*2;}
        if (scomTurn.compare("crrrc") == 0) {betTurn=22; raisesSB=
raisesSB+2; pot=pot+6*BB*2;}
        if (scomTurn.compare("crrrr") == 0) {betTurn=23; raisesSB=
raisesSB+2; pot=pot+7*BB*2;}//
        if (scomTurn.compare("rrrr") == 0) {betTurn=24; raisesSB= raisesSB+2;
pot=pot+7*BB*2;}//
        if (scomTurn.compare("crrrrf") == 0) {betTurn=25; raisesSB=
raisesSB+2; pot=pot+7*BB*2;}
        if (scomTurn.compare("crrrrc") == 0) {betTurn=26; raisesSB=
raisesSB+2; pot=pot+8*BB*2;}
        if (scomTurn.compare("rrrrf") == 0) {betTurn=27; raisesSB=
raisesSB+2; pot=pot+7*BB*2;}
        if (scomTurn.compare("rrrrc") == 0) {betTurn=28; raisesSB=
raisesSB+2; pot=pot+8*BB*2;}
```

```
        if (scomRiver.compare("") == 0) {betRiver=2; raisesSB= raisesSB+0;
pot=pot+0*BB*2;}//
        if (scomRiver.compare("c") == 0) {betRiver=3; raisesSB= raisesSB+0;
pot=pot+0*BB*2;}//
        if (scomRiver.compare("r") == 0) {betRiver=4; raisesSB= raisesSB+0;
pot=pot+1*BB*2;}//
        if (scomRiver.compare("cc") == 0) {betRiver=5; raisesSB= raisesSB+0;
pot=pot+0*BB*2;}
        if (scomRiver.compare("cr") == 0) {betRiver=6; raisesSB= raisesSB+1;
pot=pot+1*BB*2;}//
        if (scomRiver.compare("rf") == 0) {betRiver=7; raisesSB= raisesSB+0;
pot=pot+1*BB*2;}
        if (scomRiver.compare("rc") == 0) {betRiver=8; raisesSB= raisesSB+0;
pot=pot+2*BB*2;}
        if (scomRiver.compare("rr") == 0) {betRiver=9; raisesSB= raisesSB+1;
pot=pot+3*BB*2;}//
        if (scomRiver.compare("crf") == 0) {betRiver=10; raisesSB=
raisesSB+1; pot=pot+1*BB*2;}
        if (scomRiver.compare("crc") == 0) {betRiver=11; raisesSB=
raisesSB+1; pot=pot+2*BB*2;}
        if (scomRiver.compare("crr") == 0) {betRiver=12; raisesSB=
raisesSB+1; pot=pot+3*BB*2;}//
        if (scomRiver.compare("rrf") == 0) {betRiver=13; raisesSB=
raisesSB+1; pot=pot+3*BB*2;}
        if (scomRiver.compare("rrc") == 0) {betRiver=14; raisesSB=
raisesSB+1; pot=pot+4*BB*2;}
        if (scomRiver.compare("rrr") == 0) {betRiver=15; raisesSB=
raisesSB+1; pot=pot+5*BB*2;}//
        if (scomRiver.compare("crrf") == 0) {betRiver=16; raisesSB=
raisesSB+1; pot=pot+3*BB*2;}
        if (scomRiver.compare("crrc") == 0) {betRiver=17; raisesSB=
raisesSB+1; pot=pot+4*BB*2;}
        if (scomRiver.compare("crrr") == 0) {betRiver=18; raisesSB=
raisesSB+2; pot=pot+5*BB*2;}//
        if (scomRiver.compare("rrrf") == 0) {betRiver=19; raisesSB=
raisesSB+1; pot=pot+5*BB*2;}
        if (scomRiver.compare("rrrc") == 0) {betRiver=20; raisesSB=
raisesSB+1; pot=pot+6*BB*2;}
        if (scomRiver.compare("crrrf") == 0) {betRiver=21; raisesSB=
raisesSB+2; pot=pot+5*BB*2;}
```

```
        if (scomRiver.compare("crrrc") == 0) {betRiver=22; raisesSB=
raisesSB+2; pot=pot+6*BB*2;}
        if (scomRiver.compare("crrrr") == 0) {betRiver=23; raisesSB=
raisesSB+2; pot=pot+7*BB*2;}//
        if (scomRiver.compare("rrrr") == 0) {betRiver=24; raisesSB=
raisesSB+2; pot=pot+7*BB*2;}//
        if (scomRiver.compare("crrrrf") == 0) {betRiver=25; raisesSB=
raisesSB+2; pot=pot+7*BB*2;}
        if (scomRiver.compare("crrrrc") == 0) {betRiver=26; raisesSB=
raisesSB+2; pot=pot+8*BB*2;}
        if (scomRiver.compare("rrrrf") == 0) {betRiver=27; raisesSB=
raisesSB+2; pot=pot+7*BB*2;}
        if (scomRiver.compare("rrrrc") == 0) {betRiver=28; raisesSB=
raisesSB+2; pot=pot+8*BB*2;}


//azione per enviare informazione al server questo valore non serve, ma per
calcolare la FQI si
found = cartas.find_first_of(":");
temp = cartas.substr (found+1);
found = temp.find_first_of(":");
if(found!=-1)  {
temp = temp.substr (found+1);
if(temp.compare("f")==0) azione= 0;
if(temp.compare("c")==0) azione= 1;
if(temp.compare("r")==0) azione= 2;

if(azione==0) {
        if(stage==0&&betPre==0) azione=3;//cioè, è stato un fold quando non si
doveva fare
        if(stage==1&&betFloop==0) azione=3;
        if(stage==2&&betTurn==0) azione=3;
        if(stage==3&&betRiver==0) azione=3;
                                                }

                                                }
```

```
//definire qui quale degli stati calcolati saranno considerati per il FQI
stati[0]= stage;
stati[1]= raises;
stati[2]= raisesSB;
stati[3]= raiseanterior;
stati[4]= turno;
stati[5]= PV;
stati[6]= pot;
stati[7]= betPre;
stati[8]= betFloop;
stati[9]= betTurn;
stati[10]= betRiver;
stati[11]= azione;



return stati;
}



 int main () {

          int BB=10;
          int controlSB, controlBB, recompensa, ultimagiocata;
          vector <int> stati(12);
          vector <int> stati2(12);
          vector <string> lineavirtuale(40);
  string line;
          string temp, temp2, part1, part2, part3, ripartito, ripartitofut, azione,
scomPre, scomFloop, scomTurn, scomRiver, part0;
          int count1, found, found2, stage, turno, count2;
  ifstream myfile ("ROOM_90.dealer.summary");//da dove copiero

          fstream myfile2;
  myfile2.open ("example.txt");//dove scrivero


  if (myfile.is_open())
  {
    while (! myfile.eof() )
```

```
   {
    getline (myfile,line);

temp= line;
for (count1=0;temp.find("|")!=-1;count1++){
          found= temp.find("|");
          temp = temp.substr (found+1); }
          if(count1>2)

{//Inizia loop del if, abbiamo un string line con una linea

//myfile2 << line <<"\n";
found= min(line.find(":f"),min(line.find(":c"),line.find(":r")));
part1 = line.substr (0,found+1);

found2= min(line.find("f:"),min(line.find("c:"),line.find("r:")));
part2 = line.substr (found+1,found2-found);
part3 = line.substr(found2+1);

for(count1=0;count1<part2.length();count1++)       {
          ripartito= part2.substr(0,count1);
          ripartitofut=    part2.substr(0,count1+1);

          if(ripartitofut.find_last_of("f")==ripartitofut.length()-1)     azione= ":f";
          if(ripartitofut.find_last_of("c")==ripartitofut.length()-1)     azione= ":c";
          if(ripartitofut.find_last_of("r")==ripartitofut.length()-1)     azione=    ":r";


          if(ripartitofut.find_last_of("/")==ripartitofut.length()-1)     {
                    count1=count1+1;
                    ripartito= part2.substr(0,count1);
                    ripartitofut=    part2.substr(0,count1+1);
          if(ripartitofut.find_last_of("f")==ripartitofut.length()-1)     azione= ":f";
          if(ripartitofut.find_last_of("c")==ripartitofut.length()-1)     azione= ":c";
          if(ripartitofut.find_last_of("r")==ripartitofut.length()-1)     azione=    ":r";




          }

temp= ripartito;
```

```
for (count2=0;temp.find("/")!=-1;count2++){
        found= temp.find("/");
        if(count2==0) scomPre = temp.substr (0,found);
        if(count2==1) scomFloop = temp.substr (0,found);
        if(count2==2) scomTurn = temp.substr (0,found);
        temp = temp.substr (found+1);                                    }
if(count2==0) scomPre = temp.substr (0,found);
if(count2==1) scomFloop = temp.substr (0,found);
if(count2==2) scomTurn = temp.substr (0,found);
if(count2==3) scomRiver = temp.substr (0,found);
stage= count2;
//turno
if(stage==0){
        if (scomPre.compare("") == 0) turno= 1;//if 0 they are equal
        if (scomPre.compare("c") == 0) turno= 0;
        if (scomPre.compare("r") == 0) turno= 0;
        if (scomPre.compare("cr") == 0) turno= 1;
        if (scomPre.compare("rr") == 0) turno= 1;
        if (scomPre.compare("crr") == 0) turno= 0;
        if (scomPre.compare("rrr") == 0) turno= 0;
        if (scomPre.compare("crrr") == 0) turno= 1;
                                                }
if(stage==1){
        if (scomFloop.compare("") == 0) turno= 0;
        if (scomFloop.compare("c") == 0) turno= 1;
        if (scomFloop.compare("r") == 0) turno= 1;
        if (scomFloop.compare("cr") == 0) turno= 0;
        if (scomFloop.compare("rr") == 0) turno= 0;
        if (scomFloop.compare("crr") == 0) turno= 1;
        if (scomFloop.compare("rrr") == 0) turno= 1;
        if (scomFloop.compare("crrr") == 0) turno= 0;
        if (scomFloop.compare("crrrr") == 0) turno= 1;
        if (scomFloop.compare("rrrr") == 0) turno= 0;
                                                }
if(stage==2){
        if (scomTurn.compare("") == 0) turno= 0;
        if (scomTurn.compare("c") == 0) turno= 1;
        if (scomTurn.compare("r") == 0) turno= 1;
        if (scomTurn.compare("cr") == 0) turno= 0;
        if (scomTurn.compare("rr") == 0) turno= 0;
        if (scomTurn.compare("crr") == 0) turno= 1;
        if (scomTurn.compare("rrr") == 0) turno= 1;
```

```
        if (scomTurn.compare("crrr") == 0) turno= 0;
        if (scomTurn.compare("crrrr") == 0) turno= 1;
        if (scomTurn.compare("rrrr") == 0) turno= 0;
                                                }
if(stage==3){
        if (scomRiver.compare("") == 0) turno= 0;
        if (scomRiver.compare("c") == 0) turno= 1;
        if (scomRiver.compare("r") == 0) turno= 1;
        if (scomRiver.compare("cr") == 0) turno= 0;
        if (scomRiver.compare("rr") == 0) turno= 0;
        if (scomRiver.compare("crr") == 0) turno= 1;
        if (scomRiver.compare("rrr") == 0) turno= 1;
        if (scomRiver.compare("crrr") == 0) turno= 0;
        if (scomRiver.compare("crrrr") == 0) turno= 1;
        if (scomRiver.compare("rrrr") == 0) turno= 0;
                                                }

if (turno==0) part0="0:";
if (turno==1) part0="1:";


myfile2 << part0+part1+ripartito+part3+azione <<"$";


                        }

myfile2 << "2:" << line;
myfile2 <<"\n";

}//Fine loop del if
     //myfile2 << line <<"\n";;

   }
   myfile.close();
  }

        else cout << "Unable to open Dealer Summary";

myfile2.close();
myfile2.open ("example.txt");

ofstream myfile3;
```

```
myfile3.open ("data4FQI.txt");//dove scriverò
myfile3 << "12"<<"\t"<<"12"<<"\n";//indice per la procedura FQI

if (myfile2.is_open())
  {
   while (! myfile2.eof() )
   {
    getline (myfile2,line);
                        temp= line;//temp è quello che c'è tutto che manca
                        controlSB=0;//dice in che linea mettere la giocata del SB
                        controlBB=0;
                        for(count1=0;temp.find_first_of("$")!=-1;count1++)
                                                                {
                        found= temp.find_first_of("$");
                        temp2= temp.substr (0,found);
                                if(temp2.find_first_of("0")==0)                 {

                                        lineavirtuale[controlBB] = temp2;
                                        controlBB=controlBB+1;


        }
                                else
                                                                {
                                        lineavirtuale[20+controlSB] = temp2;
                                        controlSB=controlSB+1;
                                        if(temp2.find_first_of("1")==1)
        myfile3 << "ERROR" <<"\n";

                                                                }
                        temp= temp.substr(found+1);



                                                }

//mette i vettori numerici degli stati, già col stato presente e quello futuro.
                        for(count1=0;count1<controlBB;count1++)
                        {
                                if(count1+1<controlBB){
                                        if(count1>0) stati= stati2;//evita che lo
stesso stato sia calcolato due volte (piu veloce), stati2 è quello del futuro
```

```
                                        else
                                                stati=
calcstati(lineavirtuale[count1]);
                                stati2= calcstati(lineavirtuale[count1+1]);
                                ultimagiocata=0;
                                recompensa=     calcrecompensa(stati[0],
stati[3], stati[11], BB, stati[7], ultimagiocata, lineavirtuale[count1], stati[4]);


myfile3<<stati[0]<<"\t"<<stati[1]<<"\t"<<stati[2]<<"\t"<<stati[3]<<"\t"<<stati[
4]<<"\t"<<stati[5]<<"\t"<<stati[6]<<"\t"<<stati[7]<<"\t"<<stati[8]<<"\t"<<stati[
9]<<"\t"<<stati[10]<<"\t"<<stati[11]<<"\t"<<stati2[0]<<"\t"<<stati2[1]<<"\t"<<
stati2[2]<<"\t"<<stati2[3]<<"\t"<<stati2[4]<<"\t"<<stati2[5]<<"\t"<<stati2[6]<<
"\t"<<stati2[7]<<"\t"<<stati2[8]<<"\t"<<stati2[9]<<"\t"<<stati2[10]<<"\t"<<int(
recompensa)<<"\n";

                                }
                        else    {
                                if(count1>0) stati= stati2;
                                        else
                                                stati=
calcstati(lineavirtuale[count1]);
                                stati= calcstati(lineavirtuale[count1]);
                                if(stati[11]!=3)//Per non analisare giocate
sbagliate

                                {
                                        ultimagiocata=1;
                                        recompensa=
calcrecompensa(stati[0], stati[3], stati[11], BB, stati[7], ultimagiocata,
lineavirtuale[count1], stati[4]);


myfile3<<stati[0]<<"\t"<<stati[1]<<"\t"<<stati[2]<<"\t"<<stati[3]<<"\t"<<stati[
4]<<"\t"<<stati[5]<<"\t"<<stati[6]<<"\t"<<stati[7]<<"\t"<<stati[8]<<"\t"<<stati[
9]<<"\t"<<stati[10]<<"\t"<<stati[11]<<"\t"<<"123.456"<<"\t"<<"123.456"<<"\t
"<<"123.456"<<"\t"<<"123.456"<<"\t"<<"123.456"<<"\t"<<"123.456"<<"\t"<<
"123.456"<<"\t"<<"123.456"<<"\t"<<"123.456"<<"\t"<<"123.456"<<"\t"<<"12
3.456"<<"\t"<<int(recompensa)<<"\n";

                                }
                                        }
```

```
                                                    }

for(count1=0;count1<controlSB;count1++)                                    {
                                    if(count1+1<controlSB){
                                                if(count1>0) stati= stati2;
                                                        else
                                                                stati=
calcstati(lineavirtuale[count1+20]);
                                                stati2= calcstati(lineavirtuale[count1+21]);
                                                ultimagiocata=0;
                                                recompensa=         calcrecompensa(stati[0],
stati[3], stati[11], BB, stati[7], ultimagiocata, lineavirtuale[count1+20], stati[4]);


myfile3<<stati[0]<<"\t"<<stati[1]<<"\t"<<stati[2]<<"\t"<<stati[3]<<"\t"<<stati[
4]<<"\t"<<stati[5]<<"\t"<<stati[6]<<"\t"<<stati[7]<<"\t"<<stati[8]<<"\t"<<stati[
9]<<"\t"<<stati[10]<<"\t"<<stati[11]<<"\t"<<stati2[0]<<"\t"<<stati2[1]<<"\t"<<
stati2[2]<<"\t"<<stati2[3]<<"\t"<<stati2[4]<<"\t"<<stati2[5]<<"\t"<<stati2[6]<<
"\t"<<stati2[7]<<"\t"<<stati2[8]<<"\t"<<stati2[9]<<"\t"<<stati2[10]<<"\t"<<int(
recompensa)<<"\n";

                                                }
                                    else     {
                                                if(count1>0) stati= stati2;
                                                        else
                                                                stati=
calcstati(lineavirtuale[count1+20]);
                                                stati= calcstati(lineavirtuale[count1+20]);
                                                if(stati[11]!=3)//Per non analisare giocate
sbagliate

                        {
                                                        ultimagiocata=1;
                                                        recompensa=
calcrecompensa(stati[0], stati[3], stati[11], BB, stati[7], ultimagiocata,
lineavirtuale[count1+20], stati[4]);


myfile3<<stati[0]<<"\t"<<stati[1]<<"\t"<<stati[2]<<"\t"<<stati[3]<<"\t"<<stati[
4]<<"\t"<<stati[5]<<"\t"<<stati[6]<<"\t"<<stati[7]<<"\t"<<stati[8]<<"\t"<<stati[
9]<<"\t"<<stati[10]<<"\t"<<stati[11]<<"\t"<<"123.456"<<"\t"<<"123.456"<<"\t
"<<"123.456"<<"\t"<<"123.456"<<"\t"<<"123.456"<<"\t"<<"123.456"<<"\t"<<
```

```
"123.456"<<"\t"<<"123.456"<<"\t"<<"123.456"<<"\t"<<"123.456"<<"\t"<<"12
3.456"<<"\t"<<int(recompensa)<<"\n";

                }
                                                        }
                                                        }

                        //stati= calcstati(lineavirtuale[0]);
                        //myfile3 << lineavirtuale[0] << stati[0] << "\n";



//if(stati[11]>3)stati[11]=4;//quando non c'è azione, cioè nel showdown, il valore
numerico della azione è 4
//                      myfile3 << line << "\t" << stati[0] <<"\t"<< stati[1]
<<"\t"<< stati[2] <<"\t"<< stati[3] <<"\t"<< stati[4] <<"\t"<< stati[5] <<"\t"<<
stati[6]<<"\t"<< stati[7] <<"\t"<< stati[8] <<"\t"<< stati[9] <<"\t"<< stati[10]
<<"\t"<< stati[11] <<"\t" <<"\t"<<"\n";

                }
        }
        else cout << "Unable to open Data Parciale";



myfile2.close();
myfile3.close();
return 0;
}
```