POLITECNICO DI MILANO
Corso di Laurea MAGISTRALE in Ingegneria Informatica
Dipartimento di Elettronica e Informazione

APPLICAZIONE DI TECNICHE DI APPRENDIMENTO PER RINFORZO AL TRADING DEL MERCATO VALUTARIO

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

Relatore: Ing. Marcello Restelli
Correlatore: Ing. Matteo Pirotta

Tesi di Laurea di:
Luca Pirrone, matricola 801714
Alberto Ramazzotti, matricola 815963

Anno Accademico 2014-2015

Al Dreamer che è in noi
che spinge il nostro Sogno
ad altezze oltre il nostro intelletto
e ad abissi oltre le nostre emozioni
che ci chiama e ci comanda
per renderci liberi

Indice

Sommario

La nostra tesi ha come oggetto l'applicazione degli algoritmi di apprendimento per rinforzo al trading automatico sui mercati finanziari.
Più precisamente abbiamo impiegato gli algoritmi di apprendimento per rinforzo a un software che permette di fare trading automatico sul FOREX.
Lo scopo della tesi è dimostrare l'efficacia della tecnica sopra citata, per la ricerca dei valori ottimali dei parametri che vengono utilizzati all'interno di una strategia di trading automatico, in modo da massimizzarne il rendimento.
Per raggiungere tale scopo, abbiamo identificato i parametri che influiscono maggiormente sul rendimento della strategia, interfacciato il programma di trading automatico con gli algoritmi di apprendimento per rinforzo ed infine siamo passati ad eseguire i test.
Nello specifico abbiamo utilizzato gli algoritmi di apprendimento per rinforzo implementati in Matlab, per eseguire i test sulla piattaforma software Metatrader, necessari a trovare i parametri ottimali della strategia di trading. I risultati ottenuti dai test, hanno migliorato notevolmente il rendimento della strategia grazie all'uso degli algoritmi di apprendimento per rinforzo che hanno identificato parametri ottimi rispetto ai parametri precedentemente utilizzati, che sono stati trovati utilizzando l'esperienza nei mercati finanziari.

Ringraziamenti

Ringraziamo il nostro relatore ing. Marcello Restelli per averci dato la possibilità di utilizzare un nostro progetto per la creazione della tesi.
Ringraziamo inoltre l'ing. Matteo Pirotta per averci aiutato concretamente a raggiungere lo scopo della tesi.
Ringraziamo infine i nostri familiari per aver sostenuto i costi accademici e averci permesso di arrivare fin qui.

Capitolo 1

Introduzione

1.1 Inquadramento generale

L'argomento principale della nostra tesi è l'applicazione del reinforcement learning nel campo del trading automatico dei mercati finanziari.
Questo problema è interessante perché può portare sia ad ottenere policy ottimali per strategie note che scoprire strategie di mercato non ancora conosciute oggi. Tutto questo avrebbe il vantaggio di ottenere degli algoritmi di trading automatico che sono ottime fonti di investimento.
Nello specifico, abbiamo interfacciato gli algoritmi di apprendimento per rinforzo chiamati NES [7] e REPS [4] con un software che permette di fare trading automatico sul FOREX.
Lo scopo della tesi è massimizzare il rendimento di una strategia di trading automatico, attraverso la ricerca dei valori ottimali dei parametri utilizzati all'interno del software, per dimostrare l'efficacia dell' algoritmo sopra citato nel campo dei mercati finanziari. Per far questo, dopo aver identificato le variabili rilevanti della strategia, abbiamo interfacciato il programma di trading automatico con gli algoritmi di apprendimento per rinforzo utilizzando il software matlab, ed infine siamo passati ad eseguire i test facendo interagire matlab con la piattaforma di trading.

1.2 Breve descrizione del lavoro

Il Forex (Foreign exchange) è il mercato internazionale in cui avviene lo scambio delle valute. In questo mercato operano diverse entità tra cui banche internazionali, banche centrali, compagnie commerciali, intermediari e piccoli investitori. Il mercato Forex è il più liquido al mondo, i volumi contrattuali giornalieri del mercato Forex supera i 1.900 miliardi di dollari ed è composto da una enorme quantità di partecipanti.

Per interagire con questo mercato, abbiamo utilizzato la piattaforma software Metatrader 5, che permette di effettuare operazioni di compravendita in maniera diretta, semplice e veloce. La stessa piattaforma ci permette anche di utilizzare software chiamati Expert Advisor, scritti in un particolare linguaggio di programmazione (MQL), per effettuare trading in modo automatico. Lo scopo della nostra tesi è utilizzare gli algoritmi di reinforcement learning per trovare i parametri ottimi che massimizzano il rendimento di un Expert Advisor creato da noi stessi.

1.3 Struttura della tesi

La tesi è strutturata nel modo seguente:

- Nella sezione due si mostra lo stato dell'arte delle tematiche principali inerenti la nostra tesi;
- Nella sezione tre si illustra in modo dettagliato l'ambiente di sviluppo del software di trading che abbiamo utilizzato,
- Nella sezione quattro si descrive la strategia di trading da noi creata, sulla quale abbiamo applicato gli algoritmi di apprendimento per rinforzo;
- Nella sezione cinque vengono presentati e analizzati gli algoritmi di apprendimento per rinforzo utilizzati per raggiungere l'obiettivo finale;
- Nella sezione sei si espone come sono stati effettuati i test, quali sono stati i risultati ottenuti;
- Nella sezione sette si riassumono gli obbiettivi e le motivazioni della tesi, vengono descritti gli approcci utilizzati, infine si espongono quali possono essere le future direzioni di ricerca;

Per concludere si riporta l'elenco della sitografia e bibliografia utilizzate nello sviluppo della tesi.

Capitolo 2

Stato dell'arte

2.1 Introduzione

In questo capitolo vengono affrontate le principale tematiche della tesi. Iniziamo con una spiegazione generale nell'ambito dei mercati valutari, in particolare vengono definiti i termini tecnici necessari per poter affrontare la successiva argomentazione. Successivamente passiamo al trading automatico, dove spieghiamo in che cosa consiste e quali sono i vantaggi che derivano da esso. Infine, facciamo un riassunto generale sull'apprendimento per rinforzo, più precisamente si definiscono le caratteristiche principali, i vantaggi e gli algoritmi più utilizzati al giorno d'oggi.

2.2 Forex

Il Forex è il mercato internazionale in cui avviene lo scambio delle valute. Il suo nome deriva da FOReign EXchange che significa cambio estero. In tale mercato viene stabilito il valore di una valuta nei confronti di un'altra valuta, ovvero il tasso di cambio. Nel mercato del Forex, operano una pluralità di soggetti, tra cui banche centrali, compagnie commerciali, intermediari finanziari e investitori di vario genere.
Nella figura 2.1 possiamo vedere un esempio di grafico del mercato FOREX.

Diversamente dal mercato azionario, non c'è una cassa di compensazione o una borsa dove fisicamente vengono condotte le transazioni, ma quest'ultime vengono condotte dall'enorme quantità di partecipanti al mercato distribuiti geograficamente.
Il Foreign Exchange Market [5], inizialmente non era aperto ai piccoli investitori, in quanto, per potervi partecipare bisognava disporre di una grande quantità di capitale. Oggi invece, grazie alla leva finanziaria, anche i piccoli investitori

possono interagire in questo mercato. Il concetto di leva, fa riferimento alla possibilità da parte di un trader, di prendere in prestito denaro dalla società di brokeraggio e usarla per aprire una posizione nel mercato. Il valore massimo che può assumere la leva finanziaria, dipende dal broker che offre il servizio di trading. Attualmente, la leva massima offerta dai broker in circolazione, si aggira intorno a 100. Questo vuol dire che, se un trader dispone di un capitale pari a 1000 euro, egli, grazie alla leva finanziaria è come se possedesse 100 volte tanto, ovvero 100000 euro. In questo modo, anche il trader che dispone di un capitale limitato può fare trading sul forex.
La caratteristica principale di questo mercato, è l'enorme quantità dei volumi scambiati giornalmente, che supera di gran lunga qualsiasi altro mercato. Grazie a questa proprietà, è sempre possibile trovare per ogni compratore un venditore e quindi un trader può sempre aprire e chiudere posizioni in qualsiasi momento. Questo mercato è aperto 24 ore al giorno e 5 giorni a settimana, chiude solamente sabato e domenica.
Quando si apre una posizione nel mercato delle valute, non si compra o vende una valuta, ma bensì il cambio di una valuta rispetto a un'altra. Per ogni cambio è possibile assumere posizioni Long (comprare) e Short (vendere) come in qualsiasi altro mercato finanziario.
L'operatività sul Forex si basa sull'assunzione che quando si compra una determinato cambio, ad esempio EURO/DOLLARO, si prevede che la valuta "certa", ossia quella che è al numeratore, incrementi il suo valore rispetto alla valuta "incerta"ovvero quella che è al denominatore. Con una posizione di vendita, invece, si assume l'aspettativa contraria. Immaginiamo, per esempio, una situazione in cui ci si attende che il dollaro USA perda di valore nei confronti dell'euro. Un Forex trader, in una tale situazione vende dollari americani ed acquista euro,

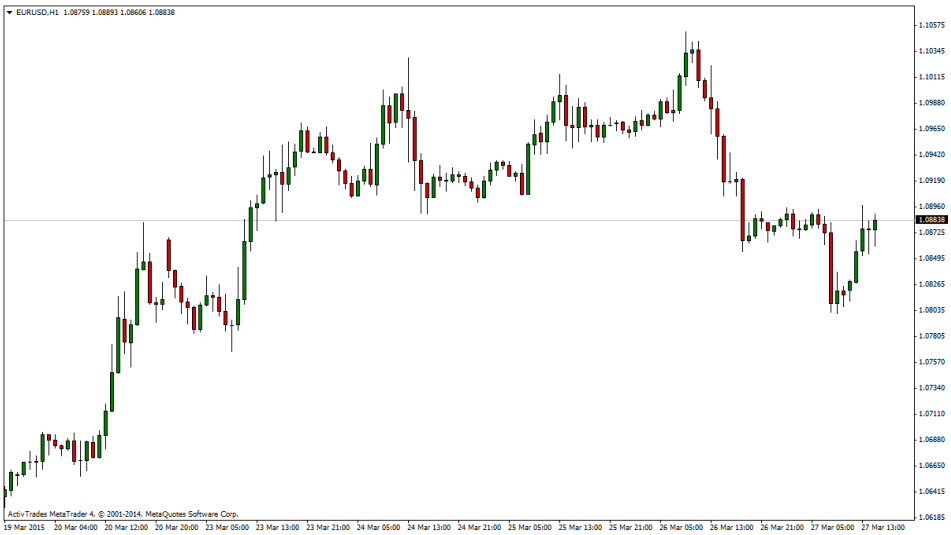
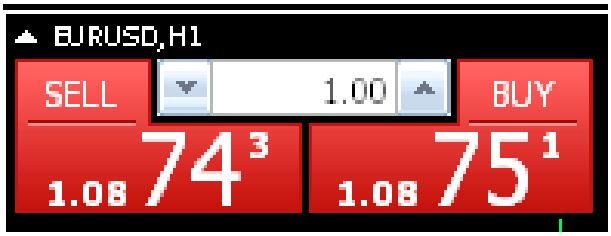


Figura 2.1: *Grafico EURUSD*

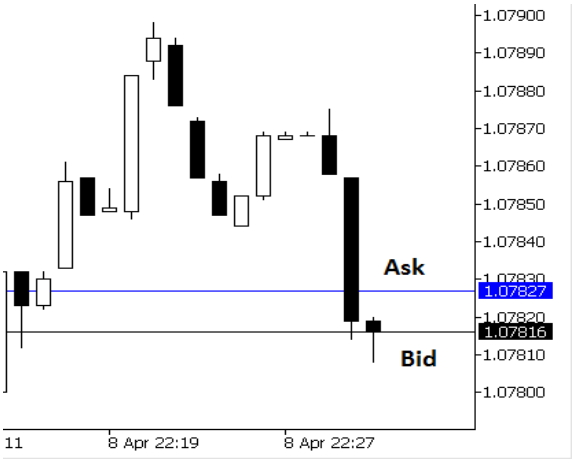
Figura 2.2: *Finestra buy-sell e quantità*



ovvero compra il cambio EURO/DOLLARO. Se l'euro aumenta di valore nei confronti del dollaro, il potere di acquisto degli euro nei confronti del dollaro aumenta, dunque il trader può ora acquistare dollari con i propri euro, ottenendo più dollari dell'ammontare iniziale, riportando un profitto.
La comune espressione "comprare basso e vendere alto", si applica anche alla negoziazione delle valute. Un trader sul mercato forex acquista valute che sono sottovalutate e vende valute che sono sopravvalutate, proprio come un operatore di borsa.
Per misurare le dimensioni di una transazione nel Forex, viene utilizzata un'unità di misura chiamata *lotto*, che corrisponde esattamente a 100000 unità della valuta certa. Le quantità minima comprabile in una singola operazione finanziaria è di 1000 unità, che corrisponde a 0.01 lotti.

La figura 2.2 rappresenta la finestra dove è possibile scegliere l'operazione di mercato da eseguire con la relativa quantità.
Per valutare le differenze di prezzo nei grafici del mercato valutario [6], si utilizza una unità di misura denominata *pip*. Quest'ultimo è l'unità in cui si valutano i profitti e le perdite per la maggior parte delle coppie di valute e sono espressi con la quarta cifra decimale, ad eccezione delle coppie in yen giapponesi in cui i pips si identificano con la seconda cifra decimale. Ad esempio, se l'Eur/Usd sale da 1,4010 a 1,4015, l'Eur/Usd è salito 5 pips.
Come possiamo vedere nella figura 2.3 il valore di una quotazione nel mercato del Forex viene espresso tramite 2 prezzi: il Bid e l'Ask. Il Bid (offerta) è il prezzo che viene utilizzato per la vendita di un determinato cambio, viceversa l'ask(domanda) è il prezzo che viene utilizzato per l'acquisto. Entrambi i prezzi sono definiti in termini di valuta "certa".
A differenza degli altri mercati, non si pagano commissioni eccetto il costo dell'eseguito che è rappresentato da una maggiorazione dello spread che rappresenta la differenza tra il prezzo di domanda e il prezzo di offerta.
Lo spread è un fattore importante da tenere in considerazione quando si fa trading sul Forex. Infatti, soprattutto nell'operatività di breve-termine, incide notevolmente sulle performance di trading in quanto il profitto viene generato su piccole variazioni di prezzo. Inoltre, è importante tenere conto del fatto che

Figura 2.3: *Differenza tra Ask e Bid*



il broker può decidere di aumentare il valore dello spread, a sua discrezione. Generalmente, gli aumenti di spread avvengono quando ci si aspettano grandi variazioni di prezzo nel mercato, e questo può avvenire ad esempio, appena prima di una news economica.
Quando si apre una posizione a mercato è possibile posizionare uno *stop loss* e un *take profit*. Il primo identifica il livello di prezzo dove verrà chiusa l'operazione nel caso in cui il mercato non vada nella direzione prevista, mentre il secondo indica il prezzo a cui si decide di uscire dal mercato ottenendo un profitto.

2.3 Trading Automatico

Grazie al rapido sviluppo della tecnologia informatica avvenuto negli ultimi decenni, nel campo dei mercati finanziari è nata la possibilità di fare trading in modo automatico [3]. Questa novità, permette di operare sul mercato in maniera totalmente automatizzata, grazie alla programmazione di appositi software.
Tale evoluzione ha permesso l'implementazione di strategie di trading, consentendo di operare sul mercato senza dover necessariamente essere presenti davanti al computer.
Una strategia di trading, indica il processo decisionale secondo cui un trader decide che azioni intraprendere sul mercato. Essa comprende una serie di fattori che tengono in considerazione diversi aspetti inerenti al mercato e alla gestione

del capitale.
I vantaggi dell'utilizzo di un sistema automatico sono molteplici:

- Il processo decisionale è autonomo, in modo tale da eliminare ogni componente interpretativa;
- Si limita il rischio perché viene sempre calcolato sistematicamente;
- Nulla è lasciato alla discrezionalità, perché vengono sempre seguite le indicazioni che il sistema fornisce;
- Si ha la possibilità di analizzare e di operare contemporaneamente su un numero elevato di strumenti finanziari 24 ore su 24;
- L'utilizzo di un trading system automatico, velocizza il processo di testing della strategia.

Inoltre il trader può scegliere se utilizzare un trading system come supporto decisionale o come sistema per l'immediata operatività su ogni segnale generato dal programma. Infatti, grazie all'elevato numero di funzioni offerte dall'ambiente di sviluppo, è possibile creare degli indicatori. Quest'ultimi, implementano al loro interno la strategia di trading ma non operano autonomamente sul mercato, essi si limitano a fornire, attraverso indicazioni grafiche o testuali, i segnali prodotti dalla strategia di trading.
L'implementazione delle strategie di trading ha portato dunque grandi vantaggi ai traders che le utilizzano. Se in passato un trader, per prendere decisioni operative, analizzava in prima persona tutte le componenti decisionali, spendendoci del tempo, ora questo non è più necessario perché è sufficiente sviluppare un software che implementi tale processo decisionale.
I software che permettono di fare trading sul forex in modo automatico, vengono chiamati Expert Advisor. Quest'ultimi hanno la possibilità di ricevere real-time le quotazioni del mercato, e di utilizzare i valori di prezzo registrati nel passato. Inoltre, grazie all'utilizzo di apposite funzioni è possibile interagire con il mercato allo scopo di eseguire richieste automatiche di compravendita.
Gli Expert Advisor dunque, se opportunamente programmati, possono a tutti gli effetti sostituire una persona. Un sistema automatico infatti, può gestire autonomamente il capitale del trader impostando a priori il rischio da utilizzare per ogni trade, scegliere a priori la policy da attuare per prendere le decisioni di entrata a mercato e di gestione delle posizioni aperte.
Tutti questi aspetti sono di fondamentale importanza perché migliorano le prestazioni dei traders, in quanto:

- permettono al trader di evitare di prendere decisioni errate dettate dalla frenesia e dall'emotività;
- non commettono errori di calcolo;
- garantiscono una maggiore velocità di esecuzione;

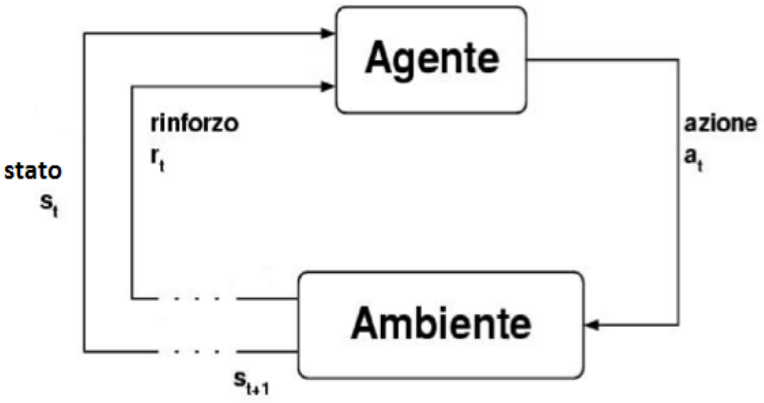
2.4 Apprendimento per Rinforzo

Il Reinforcement Learning (RL) o apprendimento per rinforzo [2], è un insieme di tecniche di apprendimento automatico che hanno l'obbiettivo di realizzare sistemi atti ad imparare, ed allo stesso tempo di adeguarsi ai cambiamenti dell'ambiente nella quale si trovano, grazie all'utilizzo di una ricompensa che gli viene assegnata per ogni azione da loro compiuta. Questo premio che viene dato in base a quanto sono apprezzate le loro prestazioni, viene chiamato rinforzo.
Tali sistemi software che elaborano informazioni generalmente in maniera automatica sono chiamati agenti intelligenti. Gli agenti decidono di compiere un'azione, selezionandole all'interno di un insieme continuo o discreto. Questa scelta viene effettuata in base a un determinato stato del sistema ed è presa a seconda dell'algoritmo di apprendimento utilizzato. Per modificare continuamente lo stato e massimizzare la sua ricompensa a lungo termine, l'agente seleziona l'azione da svolgere grazie al monitoraggio continuo dell'ambiente e a una memoria di dati.
Una delle sfide che si presentano nell'apprendimento per rinforzo (e non in altri tipi di apprendimento) è la ricerca di un trade-off tra l'esplorazione e l'azione. Per ottenere un'ampia ricompensa, e quindi per ottimizzare il rinforzo, l'agente deve preferire le azioni che nel passato hanno prodotto un buon reward. Per scoprire tali azioni, esso deve provare azioni che non ha mai selezionato prima. L'agente deve quindi sfruttare ciò che già conosce sia per ottenere il massimo reward, ma anche per esplorare e selezionare le migliori azioni per il futuro. Per fare ciò deve provare una varietà di azioni e progressivamente favorire quelle che sembrano essere migliori. Si procede dunque in modo stocastico, dove ogni azione deve essere provata molte volte per ottenere alla fine una stima affidabile sul reward previsto.
Tutti gli agenti che apprendono per rinforzo hanno obiettivi espliciti, come si può vedere nella figura 2.4 essi sono in grado di rilevare gli aspetti del loro ambiente, e sono capaci di scegliere le azioni per influenzare gli ambienti in cui si trovano. Inoltre, solitamente è assunto che all'inizio l'agente deve operare nonostante ci sia una significativa incertezza sull'ambiente in cui opera.

Gli attori del RL sono i seguenti [2]:

- *Policy*, definisce il modo in cui l'agente deve comportarsi in un dato momento. In altre parole, una policy è un mapping tra stati percepiti dall'ambiente e azioni da intraprendere quando l'agente si trova in quei determinati stati. In alcuni casi la policy può essere una semplice funzione o una tabella di ricerca, mentre in altri può comportare una vasta computazione come un processo di ricerca. La policy è il cuore di un agente di apprendimento per rinforzo nel senso che essa è sufficiente per determinare il comportamento che l'agente deve tenere. Nello specifico, una policy π è un mapping da ciascun stato, s $\in$ S, e azione, a $\in$ A(s), alla probabilità π(s, a) di prendere l'azione a nello stato s.

Figura 2.4: *Interazione agente - ambiente*



- *Reward function*, definisce l'obiettivo in un problema di apprendimento per rinforzo. In altre parole, viene mappato ogni stato percepito dall'ambiente con il proprio reward, che indica quanto si desidera essere in quello stato. L'unico obiettivo dell'agente di apprendimento per rinforzo è quello di massimizzare il reward totale che riceve nel lungo periodo. La Reward function definisce quindi quali sono gli eventi più o meno buoni per l'agente.

- *Value function*, a differenza della reward function che indica ciò che è buono in un momento immediato, una value function specifica ciò che è buono nel lungo periodo. In altre parole, il value di uno stato è la quantità totale di reward che un agente può aspettarsi di accumulare per il futuro, a partire da quello stato. Mentre i reward determinano l'immediata desiderabilità intrinseca degli stati dell'ambiente, i value indicano la desiderabilità di lungo termine degli stati tenendo conto degli stati che potrebbero seguire, e le ricompense disponibili in questi stati. Le value function sono specificate rispetto alla policy scelta. Nello specifico, la value function dello stato s utilizzando la policy π , $V^{\pi}(s)$, è il reward atteso partendo da s e ed eseguendo π. I reward sono in un certo senso primari, mentre i value, come la predizione dei rewards, sono secondari. Tuttavia, nel momento della valutazione e della decisione dell'azione da intraprendere, i value acquisiscono maggior rilevanza. Nell'apprendimento l'agente cerca azioni che determinano gli stati con più alto value, non con più alto reward, perché queste azioni ottengono la maggior quantità di reward nel lungo periodo.
  I reward sono fondamentalmente dati direttamente dall'ambiente, mentre

i value devono essere continuamente stimati dalle sequenze di osservazioni di un agente durante tutto il suo periodo di vita. Infatti, la componente più importante degli algoritmi di apprendimento per rinforzo è un metodo per stimare in modo efficace i value.

- Il *modello dell'ambiente*, è una rappresentazione interna all'agente che simula il comportamento dell'ambiente. Ad esempio, dato uno stato e un'azione, il modello prevede quale potrebbe essere il risultante stato successivo e il prossimo reward. I modelli vengono utilizzati per la pianificazione, ossia il modo di decidere una certa linea di azione considerando le possibili situazioni future prima che siano realmente accadute.

Nell'apprendimento per rinforzo, lo scopo dell'agente è formalizzato grazie a un segnale di reward che passa dall'ambiente all'agente. L'obiettivo dell'agente è quello di massimizzare la quantità totale di reward che riceve, questo non significa massimizzare il reward immediato, ma il reward ottenuto nel lungo periodo. L'utilizzo di un segnale di reward per raggiungere l'obiettivo è una delle principali caratteristiche dell'apprendimento per rinforzo. Nonostante questo metodo utilizzato per il raggiungimento degli obiettivi possa apparire a prima vista una limitazione, ha dimostrato invece di essere molto efficace.

Ricapitolando, l'obbiettivo dell'agente è apprendere una policy, π: S $\rightarrow$ A, in modo da selezionare la sua successiva azione a_t fondata sullo stato attualmente osservato s_t; cioé $\pi(s_t) = a_t$. Per la scelta di tale azione si individua la policy che massimizza i reward accumulati dall'agente nel tempo. Il valore cumulativo $V^\pi(s_t)$ che si ottiene perseguendo un'arbitraria policy π partendo da uno stato iniziale arbitrario s_t, viene definito come segue:

$$V^\pi(s_t) \equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... \equiv \sum_{i=0}^{\infty} \gamma^i r_{t+i}.$$

La sequenza di ricompense r_{t+i} è generata cominciando dallo stato s_t ed usando ripetutamente la policy π per selezionare le azioni da intraprendere.
La costante $\gamma(0 \leq \gamma < 1)$ viene chiamata fattore di sconto e rappresenta l'interesse dell'agente rispetto alle ricompense che può ricevere in futuro: Se si utilizza un γ vicino ad 1 significa che le ricompense future saranno rilevanti per l'agente, mentre se γ è vicino a 0 allora le ricompense future saranno poco rilevanti per l'agente. L'obbiettivo finale di apprendimento dell'agente è che apprenda una policy π che massimizzi $V^\pi(s)$ per tutti gli stati s. Una volta ottenuta, essa sarà la policy ottima e sarà denotata come π^*.

Esistono due tipi di interazione agente-ambiente nell'apprendimento per rinforzo, nel primo si ha una sequenza di episodi separati (episodic task), mentre nell'altro si ha un'interazione continua (continuing tasks).
Nell'ambito dell'apprendimento per rinforzo, l'agente effettua la sua decisione

in funzione di un segnale ricevuto dall'ambiente chiamato segnale di stato dell'ambiente. Idealmente, quello che si vorrebbe ottenere, è un segnale di stato che riassume le esperienze del passato in modo compatto, in modo tale che tutti le informazioni rilevanti vengano mantenute.
Un segnale di stato che riesce a mantenere tutte le informazioni pertinenti si dice essere di Markov, o avere la proprietà di Markov. La proprietà di Markov è importante nell'apprendimento per rinforzo perché le decisioni ed i values vengono considerati un unica funzione dello stato attuale.

I Markov Decision Process (MDP) [2] finiti permettono di modellizzare un processo decisionale sequenziale in un contesto stocastico. Il modello MDP è associato ad un decisore, che ha l'opportunità d'influenzare il comportamento del sistema stocastico che evolve nel tempo. Il suo obiettivo è scegliere una sequenza di azioni che portano il sistema a comportarsi in maniera ottima rispetto a criteri di prestazioni prestabiliti.
Dato che il sistema è in evoluzione, il decisore non può prevedere a priori lo stato futuro, ma per le ipotesi che abbiamo fatto è in grado di rilevare lo stato del sistema. Risolvere un modello MDP, significa trovare una policy ottima.
Per risolvere questo tipo di processi ci sono diverse tecniche [2]:

- Linear programming
- Dynamic programming
- Policy search

I metodi di RL si sono sviluppati principalmente a partire da quelli di Dynamic programming e anche in combinazione con approcci policy search. Il termine dynamic programming (DP) si riferisce a una collezione di algoritmi che possono essere usati per calcolare policy ottime dato un modello perfetto dell'ambiente formalizzato come un MDP. I classici algoritmi DP hanno un campo limitato di utilizzo in problemi reali sia per via della loro assunzione di interagire con un modello perfetto, sia perché sono computazionalmente onerosi. Tuttavia essi sono importanti nella teoria del RL, in quanto la maggior parte degli algoritmi utilizzati nella pratica si fondano su estensioni di algoritmi di programmazione dinamica. Infatti, tutti questi metodi possono essere visti come tentativi di raggiungere il più possibile lo stesso effetto della DP, solo con meno computazione e senza assunzione di modello perfetto dell'ambiente.
L'idea chiave della DP, e del RL in generale, è l'utilizzo di value function per organizzare e strutturare la ricerca di buone policy.
I principali algoritmi utilizzati per risolvere gli MDP sono:

- Policy Evaluation, consiste nel calcolare la value function V^π per ogni stato di una policy arbitraria π;
- Policy Improvement, cerca di ottimizzare i parametri di una policy π in modo da massimizzare la value function V^π;

- Policy Iteration, una volta che una policy π è stata migliorata utilizzando V^{π} per ottenere una policy perfezionata, π', possiamo computare $V^{\pi'}$ ed affinarla ulteriormente per ottenere una π'' ancor migliore. Possiamo così ottenere una sequenza di policy e value function monotone crescenti.

- Value Iteration, questo algoritmo consiste nel troncare la policy iteration in modo da non perdere la convergenza garantita dalla stessa policy iteration;

I primi due passi, Policy Evaluation e Policy Improvement, vengono iterativamente ripetuti in Policy Iteration.
Gli approcci RL possono essere distinti in tre categorie principali:

- value-based: sono metodi che utilizzano la value function per apprendere, mentre le policy sono implicite;

- policy-based: sono metodi che utilizzano le policy per apprendere, e non usano la value function;

- actor-critic: sono metodi che hanno una memoria per rappresentare esplicitamente la policy che non dipende dalla value function. La struttura della policy è conosciuta come l'actor, perché è usata per selezionare le azioni, mentre la value function stimata è la parte critic, perché critica le azioni effettuate dall'attore;

Capitolo 3

La Piattaforma Software di Trading

Una piattaforma software di trading è un programma che permette di eseguire operazioni di trading on-line. Nel corso degli anni, ne sono state sviluppate molte, ognuna con differenti caratteristiche, per meglio adattarsi alle esigenze degli investitori. Tuttavia, uno dei software più popolari e utilizzato al mondo, è la "Metatrader". Questo è dovuto al fatto che tale piattaforma, oltre a essere gratis e di facile utilizzo, è anche il programma più utilizzato dai broker per fornire i propri servizi di trading.
Esistono due differenti versioni del software Metatrader: la versione 4 e la versione 5. Entrambe le versioni offrono le funzionalità base per poter effettuare compravendite sui mercati finanziari. La differenza principale che intercorre tra le due versioni che ci ha spinto a utilizzare la versione 5 (figura 3.1), è la presenza di un tester della strategia più avanzato rispetto alla versione 4.

Come vediamo dalla figura 3.2 il programma Metatrader 5 offre numerose funzionalità, tutte raggiungibili attraverso la barra dei menù, in particolare:

- aprire un nuovo grafico per ogni strumento finanziario a disposizione;
- inserire una vasta gamma di indicatori tecnici utilizzati dai trader, come informazione ulteriore per poter predire i movimenti dei prezzi;
- visualizzare il "Tester Strategia";
- aprire l'Editor per la creazione di Expert Advisor;

Analizziamo ora due funzionalità offerte dalla piattaforma che ci hanno permesso di raggiungere gli obiettivi della nostra tesi: l'Editor per la scrittura di codice MQL5 (figura 3.4) e il Tester Strategia (figura 3.3).
L'Editor comprende una serie di funzioni che permettono al programmatore di creare gli Expert Advisor. Si tratta di una comune piattaforma di sviluppo software, che comprende l'editor di testo utilizzato per scrivere il codice sorgente

Figura 3.1: *Piattaforma Metatrader 5*

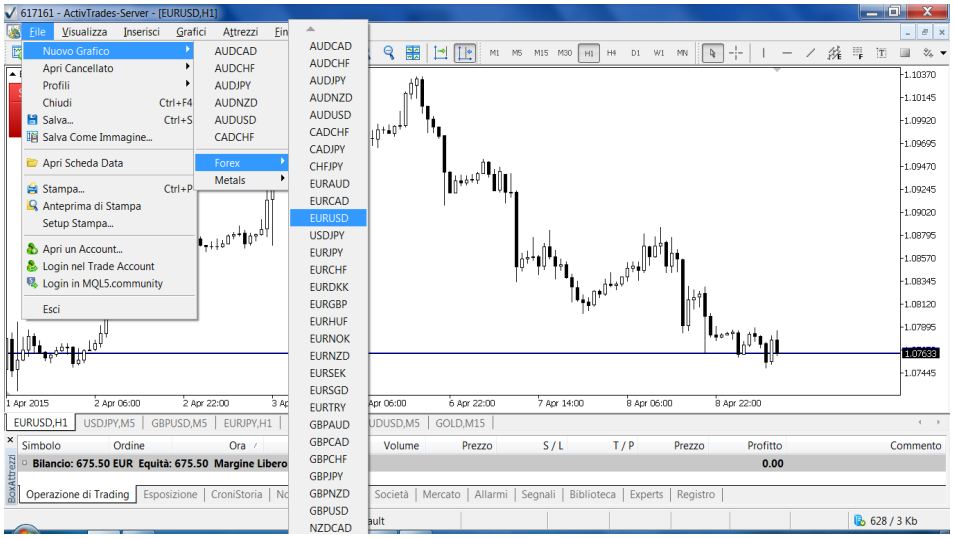
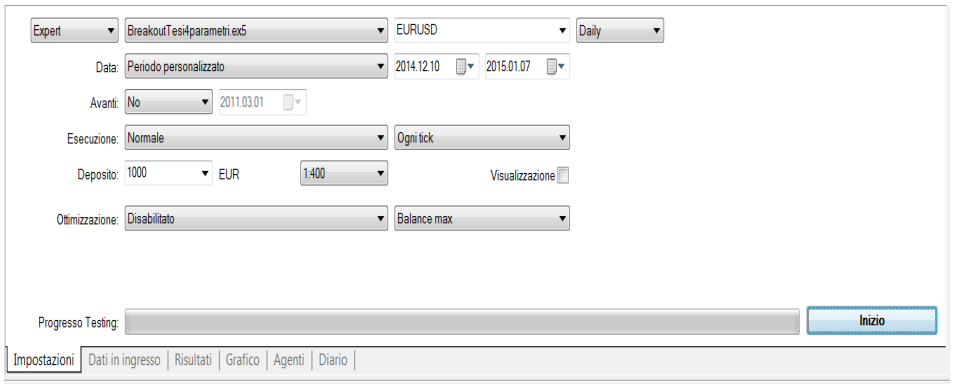


Figura 3.2: *Esempio di apertura grafico EURUSD*

Figura 3.3: *Tester Strategia*



del programma, ed il compilatore per la compilazione di tale codice.
Il linguaggio di programmazione utilizzato per sviluppare il nostro Expert Advisor è chiamato MQL5. Si tratta di un linguaggio di programmazione simile a "C", che dispone di una serie di funzioni "speciali"già implementate le quali permettono di gestire l'interazione con il mercato.
Di norma, gli Expert Advisor (EA), eseguono il loro algoritmo ad ogni singolo cambiamento del mercato (tick). Per attuare questo tipo di comportamento, viene utilizzata un'apposita funzione definita "Ontick", che come dice la parola viene chiamata ad ogni tick del mercato. Il tick rappresenta l'ampiezza della

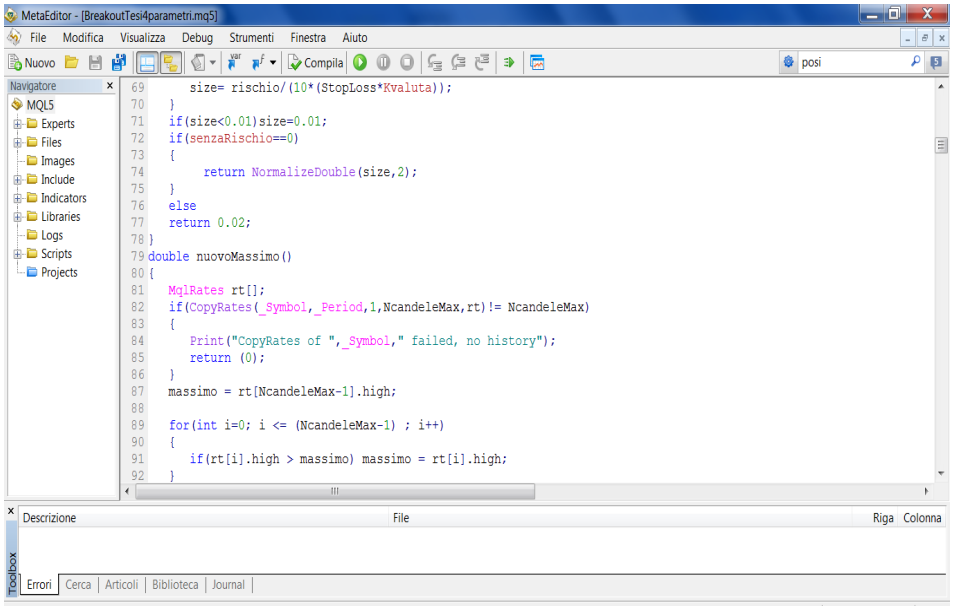


Figura 3.4: *Editor per la creazione di Expert Advisor*

minima variazione di prezzo registrabile sul mercato finanziario.

La seconda funzionalità offerta da Metatrader, che è stata di fondamentale importanza, è il Tester Strategia. Tale strumento dà la possibilità di testare il proprio Expert Advisor, simulandone l'esecuzione sui dati del passato. In tale sezione è possibile scegliere l'EA da testare, lo strumento finanziario su cui eseguire l'Expert, il bilancio iniziale da cui partire, il periodo e la precisione con cui effettuare il test (in base alla frequenza con cui viene aggiornato il prezzo). Dopo aver scelto le impostazioni desiderate, è possibile determinare i parametri di inputs dell'EA nella relativa scheda ed avviare la simulazione. Durante il funzionamento, nella scheda "grafico", si ha la possibilità di visualizzare l'andamento del profitto durante tutto il periodo selezionato precedentemente. Infine, una volta terminato il test, viene creata automaticamente una scheda chiamata "risultati"(figura 3.5) dove si possono visualizzare tutti i dati statistici della strategia appena testata.

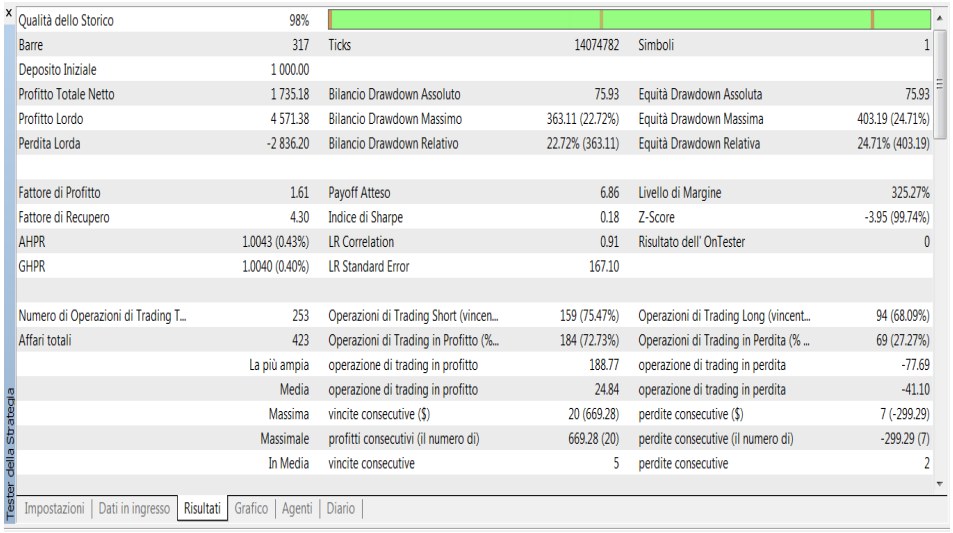


Figura 3.5: *Esempio dei risultati statistici di un test*

Capitolo 4

Il Modello di Policy utilizzato

L'Expert Advisor da noi creato è un software per il trading automatico che è in grado di sostituire completamente il processo decisionale del trader. Come già detto nel capitolo 2.2, le decisioni che un trader deve compiere coprono svariati aspetti:

- a quale livello di prezzo entrare e uscire dal mercato;
- quale rischio assumere (in termini di capitale), nel singolo trade, in modo tale che il capitale subisca una perdita che non superi un certo valore percentuale;
- calcolare la size di entrata adeguata al rischio determinato in precedenza;
- aprire la posizione desiderata, tramite gli appositi comandi offerti dalla piattaforma;
- gestire il trade aperto a mercato;

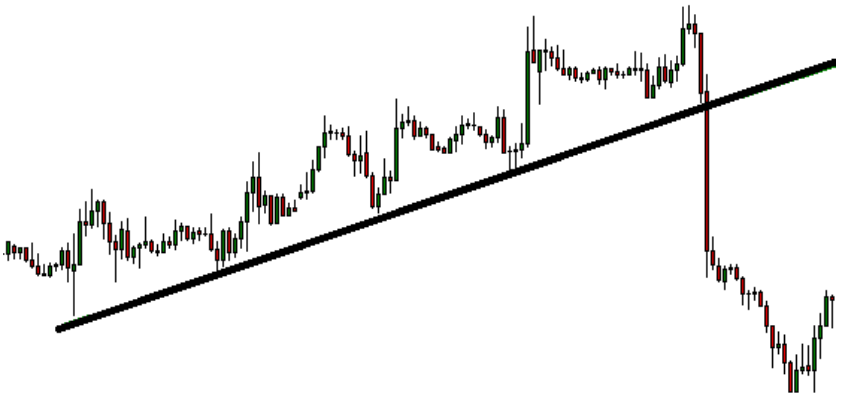
Dunque, l'Expert Advisor provvede a determinare i livelli di prezzo a cui entrare, calcolare automaticamente la quantità di lotti da comprare a seconda del rischio valutato dal trader, eseguire le operazioni di buy e sell a seconda di ciò che suggerisce la strategia, ed infine gestire il trade aperto.

La strategia che è stata implementata, appartiene a una particolare tipologia, conosciuta nel mondo del trading con l'appellativo di "Breakout Strategy". Si tratta di un metodo di trading basato su particolari comportamenti del prezzo. Nello specifico, si entra a mercato alla rottura di particolari pattern grafici, puntando sul fatto che il prezzo continuerà a muoversi con decisione nella direzione della rottura. Con la parola pattern, intendiamo particolari figure che

si vengono a creare sui grafici di prezzo. Nelle figure 4.1 e 4.2 possiamo vedere alcuni esempi dei pattern più noti :

Figura 4.1: *Esempio di breakout di una congestione*

Figura 4.2: *Esempio di breakout di una trendline*



Una congestione identifica una fase del mercato in cui esso si muove per lo più

lateralmente senza alcun trend ben definito. In questo pattern grafico, è intuitivo identificare i massimi e i minimi della congestione. Tali livelli sono importanti perchè rappresentano i livelli di prezzo a cui entrare a mercato con la Breakout Strategy. Una trendline, è una linea che unisce i minimi relativi formati dal prezzo, in un mercato rialzista. In presenza di un mercato ribassista, invece, una trendline è la linea che unisce i massimi relativi.
Prima di procedere con la descrizione della strategia è importante definire alcuni concetti.
Quando un expert advisor viene utilizzato, esso viene fatto eseguire su un particolare strumento finanziario (es. Euro/Dollaro) e su un particolare Time Frame(TF).
Il time frame nel mondo del trading, indica l'arco temporale che è possibile scegliere per campionare i dati rilevanti del mercato, in modo da poterli visualizzare sul grafico. Ai fini dell'analisi tecnica, le informazioni più rilevanti riguardo al movimento dei prezzi, sono l'apertura, la chiusura, il massimo e il minimo di un certo TF. Tutte queste informazioni vengono rappresentate sotto forma di un "grafico a candele giapponesi". Le candele di questo tipo di grafico sono costituite da un corpo detto "body"che indica l'intervallo di prezzo tra apertura e chiusura della candela. Le linee che escono dal corpo, ovvero le cosiddette "shadows", indicano il minimo e il massimo dei prezzi raggiunti durante la sessione. Quando il prezzo di chiusura è più alto rispetto a quello di apertura, il corpo della candela sarà di colore verde, al contrario quando il prezzo di chiusura è inferiore a quello di apertura, avremo una candela rossa. La colorazione verde/rosso è adottata come impostazione di default da molte piattaforme, mentre altre utilizzano quella bianca/nera, e consente una lettura più facile e intuitiva del grafico a candele.

Ora che abbiamo chiarito cosa si intende con TF e grafico a candele, passiamo ad analizzare la strategia da noi implementata. Essa si basa sul trading delle rotture di massimi e minimi creati dal prezzo. Il Time Frame utilizzato dall'EA è il Daily. Le informazioni che prendiamo in considerazione sono dunque i massimi e i minimi formati dal mercato nell'arco di un giorno. Più precisamente, i livelli di prezzo che teniamo in considerazione, non sono tutti i massimi e minimi giornalieri ma bensì i massimi e minimi creati in un arco temporale che comprende più giorni. Il numero di giorni, o equivalentemente, il numero di candele da utilizzare per determinare i massimi e i minimi, è uno dei parametri che abbiamo deciso di ottimizzare utilizzando l'algoritmo di reinforcement learning.

Al fine di raggiungere i nostri obbiettivi abbiamo inizialmente identificato i parametri che influiscono maggiormente sul rendimento della strategia. Per far questo, abbiamo creato vari files contenenti i dati storici del mercato, come il prezzo, lo spread e i valori degli indicatori tecnici maggiormente utilizzati, in seguito, li abbiamo analizzati con un software di data mining. Questo strumento utilizza l'analisi regressiva per stimare le relazioni tra le variabili in esame.

Nello specifico, aiuta a capire come una variabile dipendente cambia quando un'altra variabile indipendente varia, mentre tutte le altre variabili indipendenti rimangono fisse.

Procediamo ora ad analizzare più in dettaglio come funziona l'EA. La funzione principale del programma, quella che normalmente viene chiamata "main", è rappresentata dalla funzione denominata "Ontick". Tale funzione viene chiamata ogni volta che il mercato genera un nuovo tick, ed al suo interno, avvengono le chiamate a tutte le funzioni principali del programma.
Le funzioni principali sono :

- aggiungiNuovoMassimo(): questa funzione permette di identificare se l'ultima candela creata nel grafico è un massimo delle ultime N candele, dove N è un parametro che andremo ad ottimizzare con gli algoritmi di apprendimento per rinforzo. Se la candela in esame è un massimo, la funzione ritornerà il valore del prezzo che lo identifica, altrimenti ritornerà zero;
- aggiungiNuovoMinimo(): questa funzione permette di identificare se l'ultima candela creata nel grafico è un minimo delle ultime N candele, dove N è un parametro che andremo ad ottimizzare con gli algoritmi di apprendimento per rinforzo. Se la candela in esame è un minimo, la funzione ritornerà il valore del prezzo che lo identifica, altrimenti ritornerà zero;
- controllaSeEntrare(): ad ogni tick generato dal mercato questa funzione ha il compito di controllare se il prezzo supera i valori di massimi e minimi memorizzati in un array. Se il prezzo scende al di sotto di un valore minimo salvato nell'array, viene effettuata una richiesta automatica di vendita. Se invece il prezzo sale oltre il valore di un massimo salvato nell'array, viene effettuata una richiesta automatica di buy.
- gestisciTradeAperti(): questa funzione ha il compito di controllare se è presente una posizione aperta a mercato e in caso la trovi provvede a gestirla. La gestione di un trade consiste nel modificare il valore dello stop loss a seconda di una particolare funzione matematica da noi implementata. Nel trading, questa particolare gestione, viene chiamata "trailing stop". Questa indica uno stop loss dinamico che nel caso in cui il mercato si muove nella direzione prevista, si sposta automaticamente seguendo il prezzo, mentre nel caso contrario, rimane invariato.

Nella figura 4.3 possiamo vedere un esempio di una delle funzioni sopra citate:

```
79 double aggiungiNuovoMassimo()
80 {
81    MqlRates rt[];
82    if(CopyRates(_Symbol,_Period,1,NcandeleMax,rt)!= NcandeleMax)
83    {
84       Print("CopyRates of ",_Symbol," failed, no history");
85       return (0);
86    }
87    massimo = rt[NcandeleMax-1].high;
88 
89    for(int i=0; i <= (NcandeleMax-1) ; i++)
90    {
91       if(rt[i].high > massimo) massimo = rt[i].high;
92    }
93     if(rt[NcandeleMax-1].high == massimo)
94     {
95      return (massimo);
96     }
97    else return (0);
98 }
```

Figura 4.3: *Esempio funzione aggiungiNuovoMassimo*

Riportiamo qui di seguito, sotto forma di pseudo codice, l'algoritmo da noi creato:

1. **if** isMax() **then**
   addLevelToArray(max,level);
   end if
2. **if** isMin() **then**
   addLevelToArray(min,level);
   end if
3. foreach element of array
   if array.type = min **then**
   if bid < array.level **then**
   entrySell()
   end if
   else
   if array.type = max **then**
   if bid > array.level **then**
   entryBuy()
   end if
   end if
   end if

4. **if** PositionOpen() **then**
   if (maxCandela - openPrice) < maxBoundCandela **then**
   newStopLoss = openPrice + percentualeOttimale;
   else
   if (maxCandela - openPrice) ≥ maxBoundCandela **then**
   newStopLoss = openPrice + maxBoundCandela;
   end if
   end if

   modifyOpenPosition(StopLoss = newStopLoss);
   end if

Note:

- array è un vettore di dati composti da 2 campi : tipo e livello (type e level). Type identifica se il livello presente nell'array è un massimo o un minimo.
- addLevelToArray(type,level) è una funzione che aggiunge i valori di prezzo nell'array specificandone il tipo (massimo o minimo).
- nel punto 4 è stata definita la trailingFunction() che è la funzione che calcola a quale livello spostare lo stoploss.
- percentualeOttimale è una funzione matematica che viene calcolata in base a maxBoundCandela;
- maxBoundCandela è un valore in termini di pips oltre il quale si chiude l'operazione in profitto.

Capitolo 5

Gli Algoritmi di Apprendimento per Rinforzo

In questo capitolo vengono presentati gli algoritmi di apprendimento per rinforzo che sono stati utilizzati per raggiungere lo scopo della tesi.

5.1 NES

NES (Natural Evolution Strategies)[7] racchiude una famiglia di algoritmi di ottimizzazione, che utilizzano il gradiente naturale per aggiornare una distribuzione di ricerca parametrizzata, in direzione di più alti rendimenti.
Molti problemi di ottimizzazione nel mondo reale, sono troppo difficili e complessi per essere modellati direttamente. Dunque, può essere meglio risolverli in una maniera "black box", che non richiede informazioni supplementari sulla funzione obiettivo. I problemi che rientrano in questa categoria sono numerosi, e vanno da applicazioni nel campo della salute fino ad applicazioni scientifiche. [7] Negli ultimi cinquant'anni, in questo campo, sono stati sviluppati e applicati numerosi algoritmi, fornendo in molti casi soluzioni che sono vicine all'ottimo e che riguardano problemi difficili.
L'idea centrale di NES [7] è quella di mantenere e aggiornare in modo iterativo una distribuzione di ricerca, dove i punti di quest'ultima, sono disegnati e successivamente valutati. Tuttavia, NES aggiorna la distribuzione di ricerca, in direzione di più alti rendimenti, utilizzando il gradiente naturale. [1]

La procedura generale usata da NES è la seguente: la distribuzione di ricerca parametrizzata, viene impiegata per produrre una serie di punti di ricerca, e la funzione di rendimento viene calcolata per ognuno di questi punti. I parametri della distribuzione consentono all'algoritmo di trovare iterativamente

la struttura della funzione di rendimento. A partire dai campioni, NES stima un gradiente di ricerca dei parametri verso più alti valori di rendimento. NES in seguito compie un passo di salita lungo il gradiente naturale, il quale è un metodo al secondo ordine che, a differenza del gradiente semplice, normalizza gli aggiornamenti tenendo sotto controllo l'incertezza. Questo step è cruciale, in quanto esso previene le oscillazioni, le convergenze premature e gli effetti indesiderati derivanti da una data parametrizzazione. L'intero processo itera finché non viene raggiunto il numero massimo di iterazioni impostato.
Tutti i membri della "famiglia NES", operano in base agli stessi principi. Essi differiscono nel tipo di distribuzione e nell'approssimazione del gradiente utilizzato. Un'ultima distinzione è presente tra le distribuzioni in cui è possibile computare il gradiente naturale e le distribuzioni generali, in cui il gradiente viene stimato dai campioni.
L'idea centrale delle strategie di evoluzione naturale è quello di utilizzare i gradienti di ricerca per aggiornare i parametri della distribuzione di ricerca. Definiamo il gradiente di ricerca come un semplice gradiente di bontà sul campionamento previsto. La distribuzione di ricerca può essere vista come una distribuzione multi-normale, ma potrebbe in linea di principio essere una qualsiasi distribuzione per il quale possiamo trovare i derivati di log-densità in relazione ai suoi parametri. Il gradiente naturale è stato introdotto nel campo del machine learning da Amari ed è stato dimostrato possedere numerosi vantaggi rispetto al gradiente normale. Se usiamo θ per denotare i parametri di densità $\pi(z|\theta)$ e f(z) per denotare la funzione obbiettivo per i campioni z, possiamo scrivere l'obbiettivo atteso sotto la distribuzione di ricerca come:
$J(\theta) = \mathbb{E}_\theta[f(z)] = \int f(z)\pi(z|\theta)\, dx$
Il cosiddetto 'log-likelihood trick'ci consente di scrivere

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \int f(z)\pi(z|\theta)\, dz \\
&= \int f(z)\nabla_\theta \pi(z|\theta)\, dz \\
&= \int f(z)\nabla_\theta \pi(z|\theta)\frac{\pi(z|\theta)}{\pi(z|\theta)}\, dz \\
&= \int [f(z)\nabla_\theta log\pi(z|\theta)]\pi(z|\theta)\, dz \\
&= \mathbb{E}_\theta[f(z)\nabla_\theta log\pi(z|\theta)].
\end{aligned}$$

Da questa forma si ottiene la stima del gradiente di ricerca da campioni $z_1 ... z_\lambda$ come

$$\nabla_\theta J(\theta) \approx \frac{1}{\lambda}\sum_{k=1}^{\lambda} f(z_k)\nabla_\theta log\pi(z_k|\theta)$$

dove λ è la dimensione della popolazione. Questo gradiente sull'obbiettivo atteso fornisce una direzione di ricerca nello spazio delle distribuzioni di ricerca.

Uno schema di gradiente di salita puó aggiornare iterativamente le distribuzioni di ricerca nel seguente modo

$$\theta \leftarrow \theta + \eta \nabla_\theta J(\theta)$$

dove η indica il tasso di apprendimento.
Invece di utilizzare il gradiente stocastico normale per gli aggiornamenti, NES segue il gradiente naturale. Quest'ultimo aiuta a mitigare la lenta convergenza di salita dei gradienti normali in contesti di ottimizzazione complicati.
Il gradiente ∇J segue semplicemente la salita più ripida nello spazio dei parametri attuali θ della distribuzione. Ciò significa che data una piccola grandezza di passo ε, produrrà una nuova distribuzione con i parametri scelti dall'ipersfera di raggio ε e centro θ che massimizza J. In altre parole, la distanza euclidea nello spazio dei parametri viene utilizzato per misurare la distanza tra le distribuzioni successive. Chiaramente, questo rende l'aggiornamento dipende dalla particolare parametrizzazione della distribuzione, quindi un cambiamento di parametrizzazione porta a diversi gradienti e diversi aggiornamenti.
L'idea chiave del gradiente naturale è quello di rimuovere questa dipendenza sulla parametrizzazione basandosi su una misura più "naturale"della distanza $D(\theta'||\theta)$ tra le distribuzioni di probabilità $\pi(z|\theta)$ e $\pi(z|\theta')$. Il gradiente naturale può quindi essere formalizzato come soluzione per i problemi di ottimizzazione vincolata.

5.2 REPS

Ai fini della trattazione dell'algoritmo REPS[4], consideriamo un regolare ambiente di apprendimento per rinforzo di un Markov decision process(MDP) stazionario, con n stati s e m azioni a. Quando un agente è in uno stato s, esso compie un'azione $a \sim \pi(a|s)$, da una policy stocastica π. Successivamente, l'agente si trasferisce da uno stato s a uno stato s' con probabilità di transizione $p(s'|s,a) = P^a_{ss'}$, e riceve un reward $r(s,a) = R^a_s \in \Re$. Come risultato di questo trasferimento di stato, l'agente può convergere verso una distribuzione stazionaria di stati $\mu^\pi(s)$ per cui

(1)
$$\forall s' : \sum_{s,a} \mu^\pi(s)\pi(a|s)p(s'|s,a) = \mu^\pi(s')$$

Lo scopo dell'agente è di trovare una policy π che massimizza il ritorno atteso

$$J(\pi) = \sum_{s,a} \mu^\pi(s)\pi(a|s)r(s,a)$$

soggetto al vincolo dell'equazione (1) e che μ^π e π siano distribuzioni di probabilità. In certi casi, solo alcune caratteristiche dell'intero stato s sono rilevanti per l'agente. In questo caso, si richiede solamente la caratteristica di vettore stazionario

$$\sum_{s,a,s'} \mu^\pi(s)\pi(a|s)p(s'|s,a)\phi_{s'} = \sum_{s'} \mu^\pi(s')\phi_{s'}$$

Relative entropy policy search (REPS) cerca di trovare le policy ottime che massimizzano il ritorno atteso basandosi su un intero set di stati, azioni e rewards. Allo stesso tempo, si vuole vincolare la perdita di informazioni misurata utilizzando l'entropia relativa tra la distribuzione di dati osservata $q(s,a)$ e la distribuzione di dati $p^\pi(s,a) = \mu^\pi(s)\pi(a|s)$ generata dalla nuova policy π. Idealmente, si vogliono utilizzare tutti i campioni (s,a,s',r) indipendentemente, dunque, si esprime il vincolo di perdita di informazioni come

$$D(p^\pi \| q) = \sum_{s,a} \mu^\pi(s)\pi(a|s) \log \frac{\mu^\pi(s)\pi(a|s)}{q(s,a)} \leq \varepsilon$$

dove $D(p^\pi \| q)$ denota la divergenza di Kullback-Leibler, $q(s,a)$ denota la distribuzione stato-azione osservata, and ε è la massima perdita di informazioni.

L'obiettivo di REPS viene tradotto in formule nella seguente maniera

$$\max_{\pi,\mu^\pi} J(\pi) = \sum_{s,a} \mu^\pi(s)\pi(a|s)R^a_s,$$

$$\varepsilon \geq \sum_{s,a} \mu^\pi(s)\pi(a|s) \log \frac{\mu^\pi(s)\pi(a|s)}{q(s,a)},$$

$$\sum_{s'} \mu^\pi(s')\phi_{s'} = \sum_{s,a,s'} \mu^\pi(s)\pi(a|s)P^a_{s,s'}\phi_{s'},$$

$$1 = \sum_{s,a} \mu^\pi(s)\pi(a|s).$$

Sia μ^π che π sono distribuzioni di probabilità e le feature $\phi_{s'}$ del MDP sono stazionarie data la politica π.
La ε può essere scelta liberamente, tuttavia a valori grandi corrispondono passi

grandi e se il valore diventa eccessivamente alto, può “distruggere”la policy. Il suo valore dipende sia dal problema in questione, sia dal numero di campioni totali disponibili.

Capitolo 6

Realizzazioni sperimentali e valutazione

6.1 Introduzione

In questo capitolo analizziamo in dettaglio l'architettura del sistema utilizzato, specificando quali programmi sono stati usati. Inoltre vengono analizzati i risultati ottenuti sia durante la fase di training, sia durante la fase di testing.

6.2 Architettura del Sistema

L'architettura del sistema che ci ha permesso di raggiungere gli obiettivi della tesi, è composta dalla piattaforma software Metatrader 5 e dal programma Matlab. Nello specifico, per quanto riguarda Metatrader, sono stati utilizzati il tester strategia e l'editor per il linguaggio MQL5, mentre gli algoritmi di Reinforcement Learning utilizzati negli esperimenti sono stati realizzati in Matlab . Per poter interfacciare queste due realtà, è stato necessario ricorrere alle funzionalità offerte dalla command line di Windows. Infatti, grazie ad essa, è stato possibile lanciare in modo automatico i test dal tester strategia, attraverso una stringa di codice inserita all'interno di Matlab. Quest'ultima eseguendo una system call, esegue Metatrader insieme ad un file di configurazione che permette di configurare automaticamente il tester strategia in modo da poter avviare i test con le impostazioni desiderate.
Lo schema di funzionamento del processo di testing, può essere sintetizzato in questo modo:

1. viene lanciata l'esecuzione del codice di reinforcement learning, presente in Matlab;

2. gli algoritmi di apprendimento determinano i valori dei parametri da passare all'Expert Advisor;

3. viene eseguita la system call, che apre metatrader facendo partire il testing con i parametri appena calcolati dall'algoritmo;

4. al termine della simulazione, l'Expert scrive un file contenente il valore del reward accumulato utilizzando gli ultimi valori dei parametri passati;

5. viene ripresa l'esecuzione del codice in Matlab, il quale, analizza il reward ottenuto dall'ultima simulazione lanciata, avvia il processo di learning e riprende dal punto 2;

6. al raggiungimento del numero delle iterazioni impostate inizialmente il processo viene terminato.

6.3 Training

Abbiamo effettuato svariati test per raggiungere il nostro obbiettivo ed ottenere i migliori risultati possibili.
I principali parametri che determinano il comportamento dell'EA e che abbiamo scelto di ottimizzare per la policy da noi utilizzata sono i seguenti:

- RischioPerTrade: identifica il rischio massimo percentuale del proprio capitale che si vuole avere in ciascun trade in caso di perdita;
- OffsetLivelli: identifica a quale distanza dai livelli di prezzo salvati si vuole entrare in termini di pips. Se il valore è negativo significa che si anticiperà l'entrata rispetto al livello di prezzo salvato, in caso contrario si posticiperà;
- StopLoss: identifica la distanza in pips dal prezzo di entrata nella quale verrà chiusa l'operazione nel caso in cui il mercato non vada nella direzione prevista;
- DeltaActivation: identifica la distanza in pips dal livello di entrata dalla quale si inizierà a spostare lo stop loss usando una particolare trailing stop presente nella policy.

I valori dei parametri utilizzati in precedenza (che identificheremo con standard) sono:

- RischioPerTrade: 0.03;
- OffsetLivelli: 1;
- StopLoss: 15;
- DeltaActivation: 3;

Tali valori sono stati ottenuti in modo intuitivo in base alla nostra esperienza sui mercati finanziari.
Il processo di training è stato effettuato su un periodo totale di due anni, precisamente dal 01/01/2012 al 31/12/2013.

Come primo algoritmo di apprendimento abbiamo utilizzato NES con uno step size che è stato fissato ad-hoc per il problema. Al termine dell'esecuzione, l'algoritmo ha fornito i seguenti parametri che rappresentano la media dei valori ottenuti:

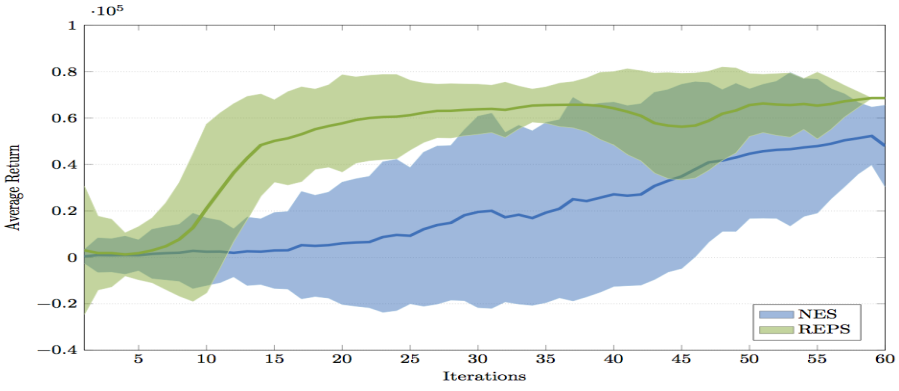
- RischioPerTrade: 0.0437;
- OffsetLivelli: -0.0102;
- StopLoss: 5.9106;
- DeltaActivation: -0.2739;

Successivamente abbiamo utilizzato l'algoritmo REPS con un valore di epsilon pari a 0.9 ed eseguito nuovamente i test.
La policy ottimale ha fornito i seguenti parametri:

- RischioPerTrade: 0.0427;
- OffsetLivelli: 0.0617;
- StopLoss: 5.3106;
- DeltaActivation: 0.4155;

Nella figura 6.1 possiamo notare come è evoluta la performance durante l'apprendimento con i due algoritmi NES e REPS

Figura 6.1: *Grafico gradiente algoritmi NES e REPS*



Dalla figura 6.1 è facile notare che l'algoritmo REPS è stato molto più rapido ad avvicinarsi alla policy ottima rispetto a NES.

Nelle figure da 6.2 a 6.5 abbiamo riportato come si sono evoluti i valori dei quattro parametri ottimizzati con le loro rispettive varianze.

Figura 6.2: *Varianza del parametro RischioPerTrade utilizzando NES e REPS*

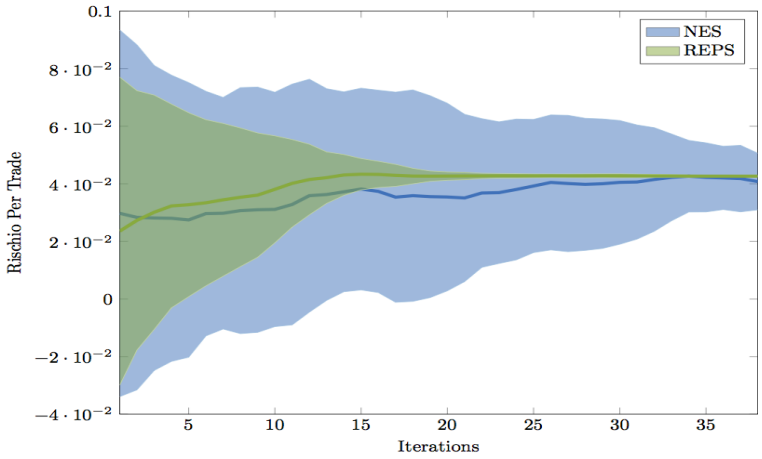


Figura 6.3: *Varianza del parametro OffsetLivello utilizzando NES e REPS*

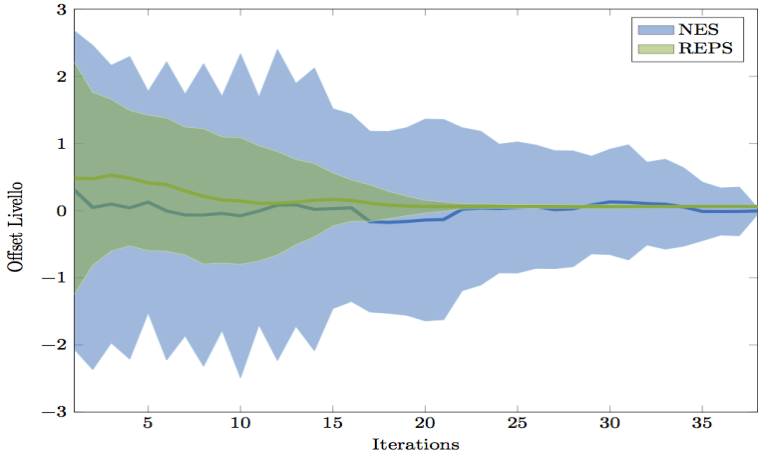


Figura 6.4: *Varianza del parametro StopLoss utilizzando NES e REPS*

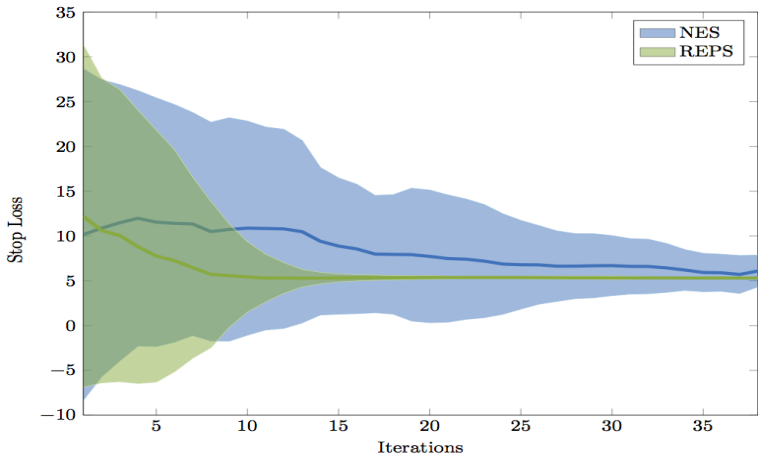
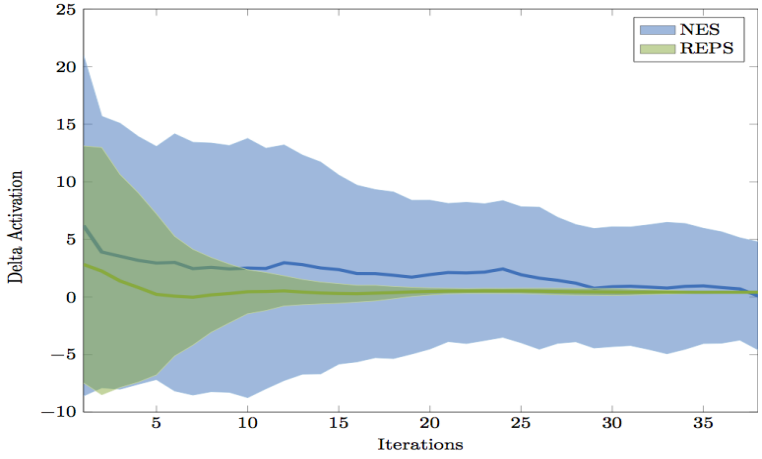
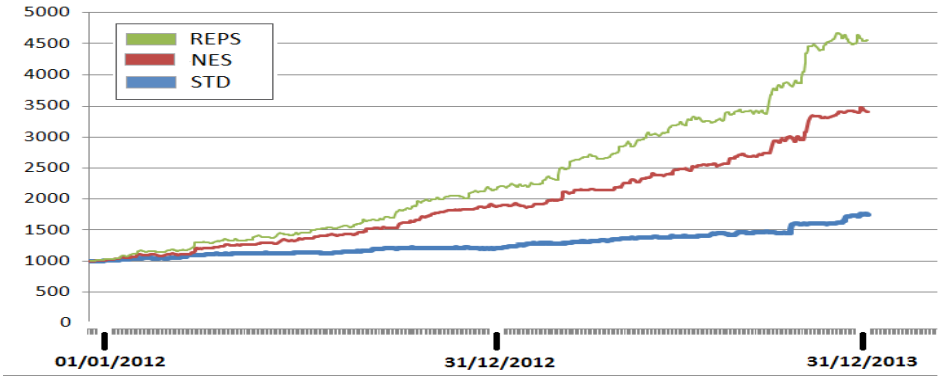


Figura 6.5: *Varianza del parametro DeltaActivation utilizzando NES e REPS*



Riportiamo ora nella figura 6.6 la differenza tra l'andamento del profitto utilizzando i parametri standard e i parametri ottimizzati

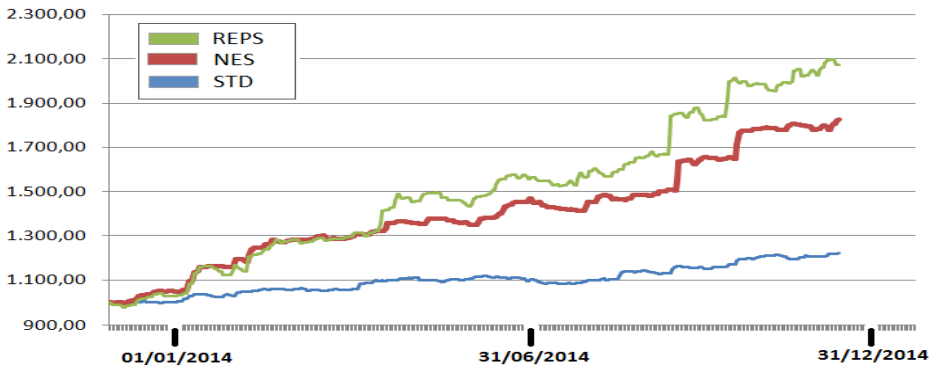
Figura 6.6: *Differenza andamento profitti tra policy standard e policy ottimizzate con NES e REPS nel periodo di training*



6.4 Testing

Dopo aver trovato la policy ottimale con i due algoritmi di apprendimento abbiamo testato il rendimento di quest'ultima su un periodo differente in modo da garantirne l'efficenza.
Nella figura 6.7 possiamo vedere la differenza degli andamenti del profitto utilizzando i parametri trovati dai due algoritmi e quello ottenuto utilizzando i parametri standard

Figura 6.7: *Differenza andamento profitti tra policy standard e policy ottimizzate con NES e REPS nel periodo di testing*



Come si può vedere l'algoritmo REPS è quello che ottiene il maggior profitto, e si conferma essere il migliore sia in termini di velocità di apprendimento sia in termini di rendimento.

Riportiamo infine nelle figure 6.8 6.9 i risultati statistici dei due test effettuati con i parametri di REPS e NES

Figura 6.8: *Statistiche risultati test NES*

Deposito Iniziale	1 000.00				
Profitto Totale Netto	874.15	Bilancio Drawdown Assoluto	2.85	Equità Drawdown Assoluta	2.85
Profitto Lordo	1 247.15	Bilancio Drawdown Massimo	54.43 (3.71%)	Equità Drawdown Massima	60.52 (3.26%)
Perdita Lorda	-373.00	Bilancio Drawdown Relativo	3.71% (54.43)	Equità Drawdown Relativa	4.10% (60.47)
Fattore di Profitto	3.34	Payoff Atteso	2.70	Livello di Margine	722.54%
Fattore di Recupero	14.44	Indice di Sharpe	0.30	Z-Score	-6.82 (99.74%)
AHPR	1.0020 (0.20%)	LR Correlation	0.98	Risultato dell' OnTester	0
GHPR	1.0019 (0.19%)	LR Standard Error	50.25		
Numero di Operazioni di Trading T...	324	Operazioni di Trading Short (vincen...	198 (45.96%)	Operazioni di Trading Long (vincent...	126 (79.37%)
Affari totali	493	Operazioni di Trading in Profitto (%...	191 (58.95%)	Operazioni di Trading in Perdita (% ...	133 (41.05%)
	La più ampia	operazione di trading in profitto	87.80	operazione di trading in perdita	-24.91
	Media	operazione di trading in profitto	6.53	operazione di trading in perdita	-2.80
	Massima	vincite consecutive ($)	16 (115.04)	perdite consecutive ($)	12 (-33.91)
	Massimale	profitti consecutivi (il numero di)	137.61 (6)	perdite consecutive (il numero di)	-47.85 (4)
	In Media	vincite consecutive	4	perdite consecutive	3

Figura 6.9: *Statistiche risultati test REPS*

Deposito Iniziale	1 000.00				
Profitto Totale Netto	1 201.67	Bilancio Drawdown Assoluto	23.21	Equità Drawdown Assoluta	23.21
Profitto Lordo	2 017.67	Bilancio Drawdown Massimo	85.72 (4.08%)	Equità Drawdown Massima	104.37 (4.94%)
Perdita Lorda	-816.00	Bilancio Drawdown Relativo	4.17% (62.31)	Equità Drawdown Relativa	5.00% (75.40)
Fattore di Profitto	2.47	Payoff Atteso	3.79	Livello di Margine	615.14%
Fattore di Recupero	11.51	Indice di Sharpe	0.28	Z-Score	-4.12 (99.74%)
AHPR	1.0025 (0.25%)	LR Correlation	0.99	Risultato dell' OnTester	0
GHPR	1.0025 (0.25%)	LR Standard Error	54.39		
Numero di Operazioni di Trading T...	317	Operazioni di Trading Short (vincen...	193 (62.18%)	Operazioni di Trading Long (vincent...	124 (71.77%)
Affari totali	486	Operazioni di Trading in Profitto (%...	209 (65.93%)	Operazioni di Trading in Perdita (% ...	108 (34.07%)
	La più ampia	operazione di trading in profitto	118.53	operazione di trading in perdita	-33.21
	Media	operazione di trading in profitto	9.65	operazione di trading in perdita	-7.56
	Massima	vincite consecutive ($)	14 (136.16)	perdite consecutive ($)	5 (-85.72)
	Massimale	profitti consecutivi (il numero di)	189.40 (12)	perdite consecutive (il numero di)	-85.72 (5)
	In Media	vincite consecutive	4	perdite consecutive	2

Capitolo 7

Direzioni future di ricerca e conclusioni

L'obiettivo della tesi è dimostrare l'efficacia delle tecniche di apprendimento per rinforzo, per la ricerca dei valori ottimali dei parametri utilizzati all'interno di una strategia di trading automatico, in modo da massimizzarne il rendimento.
Più precisamente sono stati impiegati gli algoritmi di apprendimento per rinforzo NES e REPS. Essi hanno prodotto risultati simili tra loro, che migliorano di molto le performance rispetto a quelle ottenute dai parametri standard. Dai test effettuati si può affermare che l'algoritmo RESP è risultato esser molto più rapido nell'apprendimento rispetto a NES e quindi ha permesso di arrivare velocemente ad ottenere la policy ottima. Date le conferme che sono state ottenute dal periodo di testing sull'anno 2014, siamo soddisfatti dei risultati ottenuti.
In eventuali studi futuri si potrebbe ampliare la ricerca provando ad utilizzare altri algoritmi di apprendimento per rinforzo, per riuscire ad ottenere ulteriori conferme sulla policy ottima trovata.
In alternativa si potrebbero utilizzare gli stessi algoritmi NES e REPS per effettuare un'ulteriore ricerca dei valori dei parametri della policy che riguardano l'intera gestione dei trade aperti, in modo da poter lasciar decidere agli algoritmi di apprendimento quale deve essere il comportamento da tenere una volta che siamo a mercato.

Bibliografia

[1] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[2] Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 1998.

[3] Michael AH Dempster and Vasco Leemans. An automated fx trading system using adaptive reinforcement learning. *Expert Systems with Applications*, 30(3):543–552, 2006.

[4] Jan Peters, Katharina Mülling, and Yasemin Altun. Relative entropy policy search. In *AAAI*, 2010.

[5] IW BANK S.p.A. Guida operativa al forex, 2011.

[6] UfxMarkets. La guida completa al forex di ufxmarkets, 2012.

[7] Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. Natural evolution strategies. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 3381–3387. IEEE, 2008.