UNIVERSITÀ DI MODENA E REGGIO EMILIA

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

# Tecniche di Deep Q-Learning applicate al Forex Trading

**Relatore:**
Prof. Simone Calderara

**Candidato:**
Folloni Riccardo

**Tutore Aziendale**
**Axyon.ai**
Ing. Daniele Grassi

13 Luglio 2017

Alla mia famiglia ed in particolare ai miei genitori, i quali mi hanno sostenuto e consigliato in ogni mia scelta di vita.

Logic will get you from A to B. Imagination will take you everywhere.
*Albert Einstein*

Life is one big training set.
*David Silver*

# Ringraziamenti

Durante lo sviluppo di questa tesi e durante la mia carriera universitaria ho avuto il piacere di ricevere il sostegno di molte persone pronte ad aiutarmi nei momenti di difficoltà. Si ringraziano:

- i miei compagni di corso, con cui ho spesso collaborato e senza i quali diverse volte mi sarei trovato in seria difficoltà;
- i miei amici, sui quali ho sempre potuto contare e i quali mi hanno regalato momenti indimenticabili al di fuori dell'università;
- la mia famiglia ed in particolare mio fratello Eugenio, mia sorella Aurora, e i miei genitori, costante punto di riferimento nella vita in qualsiasi attività svolgessi;
- i miei relatori prof. Simone Calderara e l'ing. Daniele Grassi, i colleghi Jacopo e Martina e il dott. Fabio Lanzi i quali mi hanno sopportato nelle mie innumerevoli richieste di confronti e pareri e sono stati essenziali per il concreto sviluppo delle idee riguardanti questa tesi.

# Abstract

Il Deep Reinforcement learning è stato oggetto di grande interesse negli ultimi anni, raggiungendo interessanti risultati in domini come i Giochi e la Robotica, dove metodi usati in precedenza hanno fallito. Anche se i giochi sono i maggiori responsabili per la crescente importanza del Reinforcement Learning, in questa tesi si studia e applica un particolare algoritmo appartenente a questa famiglia di algoritmi, il Deep Q-Learning, al fine di trovare strategie potenzialmente proficue per il Foreign Exchange Market.

# Indice

# Elenco delle figure

# Capitolo 1

# Introduzione

Il Foreign Exchange market è il più grande mercato finanziario al mondo con il maggior volume monetario scambiato su base giornaliera, aperto 24 ore su 24 per 5 giorni a settimana, coinvolgendo differenti tipologie di partecipanti, pubblici e privati. Mentre fino agli anni '80 gli investimenti erano effettuati via telefono con informazioni ridotte e accessibili a un minor numero di investitori, in particolare banche e istituzioni finanziarie, allo stato attuale il mercato del Forex è cambiato radicalmente. La forte disponibilità di dati in tempo reale tramite, la diffusione a livello mondiale della rete internet e le grandi capacità computazionali dei calcolatori hanno fatto sì che la maggior parte degli investimenti non vengono effettuati da trader umani, al contrario vi è un numero crescente di computer dotati di "intelligenza" in grado di capire quali sono le casistiche migliori in cui è opportuno seguire un certo trend di mercato e agire in modo opportuno. In un primo momento ha preso piede il concetto di *Algorithmic trading*, in cui l'algoritmo è caratterizzato da un set di regole definito in modo da compiere determinate azioni in alcune configurazioni di mercato. Questo set di regole è definito manualmente da trader umani che studiano e definiscono quali sono le migliori condizioni per effettuare particolari scelte di mercato. La naturale evoluzione di questa tipologia di algoritmi è lo sfruttamento di tecniche di machine learning, le quali non prevedono più l'intervento umano a livello di regole decisionali, ma solamente a livello metodologico. L'aiuto apportato da queste nuove tecniche è l'estrazione automatica di conoscenza dai dati passati, permettendo di ricavare un migliore e più definito set di regole potenzialmente vincenti nel mercato finanziario. Il Deep reinforcement learning è una sotto-categoria di algoritmi di machine learning che ha preso moltissimo piede negli ultimi anni grazie ai grandissimi successi ottenuti in ambiti come robotica, giochi da tavolo e videogiochi, riuscendo persino a creare un algoritmo in grado di battere il campione mondiale di Go, uno dei più complessi giochi da tavolo dal punto di vista strategico. Tra le tecniche di Deep reinforcement learning si trova il Deep Q-Learning, metodologia utilizzata per ottenere alti punteggi nell'ambito dei giochi Atari ([18]) spesso molto più alti di quelli ottenuti dagli esseri umani. Viste le caratteristiche di

questa metodologia e le possibilità di elaborare moltissimi dati di tassi di cambio e volumi di cambi di valuta, risulta di particolare rilevanza lo studio e l'applicazione di metodologie di Deep Q-Learning all'interno del Forex trading.

## 1.1 Motivazioni

L'idea dell'utilizzo di tecniche di Deep Q-Learning in ambito finanziario nasce in una azienda estremamente innovativa e all'avanguardia, Axyon AI. Costantemente alla ricerca di nuove soluzioni che possano migliorare i propri sistemi predittivi, essa si pone l'obiettivo di dedicare gran parte del tempo all'innovazione tecnologica, in modo da portare i propri risultati al più alto livello possibile. In questo contesto si sono volute esplorare le potenzialità di questa nuova metodologia che si pone in leggero contrasto con i sistemi utilizzati normalmente in ambito aziendale, introducendo la possibilità che l'agente capisca quali strategie seguire in modo da ottenere profitti più alti senza comunicare esplicitamente a esso quali sono le mosse opportune da compiere.

Il reinforcement learning non è totalmente nuovo nell'ambito del mondo finanziario, ma sono ancora in numero estremamente limitato le risorse e i riferimenti ufficiali che fanno utilizzo di deep reinforcement learning come metodologia completa per la risoluzione di problemi in ambito di Forex Trading.

## 1.2 Obiettivi

Ispirati dal grande successo del reinforcement learning nell'ultimo periodo, lo scopo finale della tesi è valutare in maniera approfondita le potenzialità del Deep Q-Learning come metodologia per permettere a un computer "intelligente" di operare in modo totalmente autonomo nel mondo del Forex Trading. Per fare ciò risulta necessario l'utilizzo di strumenti all'avanguardia e le ultime tecnologie messe a disposizione sul mercato, integrando le stesse con l'utilizzo di parte della piattaforma aziendale per la creazione di Deep neural networks. Inoltre ci si pone lo scopo di creare un sistema di simulazione del mondo del forex, su cui l'agente di reinforcement learning ha la possibilità di testare le proprie strategie e migliorare le proprie prestazioni nel tempo. Grazie a questo sistema sarà possibile valutare in termini qualitativi e quantitativi i risultati ottenuti.

## 1.3 Azienda - Axyon.ai

Axyon AI è una azienda di Modena che offre soluzioni di intelligenza artificiale legate principalmente al mondo della finanza. Nata come spinoff di DM Digital SRL, Axyon si basa su tre vantaggi competitivi, così da porsi in una posizione di forza per diventare leader del mercato nella fornitura di soluzioni altamente accurate nel settore della finanza:

- Tecnologia: da parte dell'azienda è stata sviluppata una piattaforma di Deep Learning (Axyon Platform) proprietaria basata sul mondo finanziario che offre la possibilità di sviluppare modelli predittivi con la massima efficienza.
- Esperienza: dal 2015 sono stati completati da parte dell'azienda una serie di *Proof of concepts* per la finanza e attualmente sono pronti due prodotti per il mercato e un terzo al termine dello sviluppo.
- Skills: team ben bilanciato e supporto da parte di un grande ecosistema di consulenti, partners e a stretto contatto con il mondo universitario.

L'azienda Axyon AI punta a diventare leader di mercato nel settore della finanza ad alto margine offrendo predizioni altamente accurate basate su modelli di *Deep Learning*. Per raggiungere tali obiettivi vengono seguite tre strade:

- Tecnologia: migliorare ed estendere la tecnologia proprietaria che serve a mantenere un vantaggio competitivo nello sviluppo di modelli di *artificial intelligence* per il settore finanziario.
- Soluzioni: offrire i modelli predittivi in soluzioni complete e mirate a specifici casi d'uso nel settore della finanza ad alto margine.
- Mercato: partner commerciali già posizionati sui vari mercati in modo da poter accedere direttamente a livello di *decision-maker*.

## 1.4 Outline dei contenuti

Di seguito si riporta una breve sintesi sui contenuti della tesi, divisi per capitoli.

- Nel secondo capitolo si offre una panoramica riguardante i concetti fondamentali e le dinamiche del mercato del Forex, includendo una breve sintesi teorica in ambito di Machine learning e Deep learning.

- Nel terzo capitolo si affrontano le basi teoriche di Reinforcement learning e il legame tra esso e le Deep neural networks, dando uno sguardo più approfondito all'algoritmo oggetto della tesi, il Deep Q-Learning. Infine si tratta brevemente delle librerie e dei framework utili allo sviluppo del software.

- Il quarto capitolo prevede trattazione a grana fine delle varie fasi di sviluppo, comprendendo l'analisi del problema e la strutturazione del progetto per la risoluzione di quest'ultimo.

- Il quinto capitolo riporta la trattazione dei risultati ottenuti da un punto di vista sia qualitativo che quantitativo.

- Il sesto capitolo tratta gli obiettivi raggiunti durante la progettazione e lo sviluppo del modello di Deep Q-Learning, portando infine un'analisi di quali potrebbero essere le possibili tecniche di miglioramento futuro per quanto riguarda la metodologia utilizzata.

# Capitolo 2

# Contesto e ambiente di sviluppo

Il termine "mercato finanziario" viene utilizzato per indicare il luogo in cui vengono scambiati strumenti finanziari di vario genere. Ogni tipologia di mercato è contraddistinto da diverse caratteristiche, tra cui: l'organizzazione del mercato che ne determina la struttura, gli enti che ne svolgono l'attività di supervisione, le modalità di scambio dei beni costituenti quel mercato e i partecipanti alla compra/vendita degli strumenti finanziari. Il mondo dei mercati finanziari è estremamente vasto. Nell'ambito di questo progetto ci si focalizza su uno dei mercati più importanti al mondo: il Foreign exchange market. Nel presente capitolo ci si pone l'obiettivo di trattare le basi teoriche del Forex Trading rivolgendo particolare attenzione alle modalità con cui è possibile guadagnare all'interno di questa tipologia di mercato tramite l'utilizzo dell'*algorithmic trading* e la relazione di quest'ultimo con le tecniche machine learning e deep learning.

## 2.1 Forex - Foreign exchange market

Il *foreign exchange market*, o FX Market, è un mercato interbancario a livello mondiale per la compravendita di valute estere. Esso è attualmente il mercato di trading più grande al mondo per quanto riguarda il volume di commercio con uno scambio quotidiano di 5.100 miliardi di dollari ad Aprile 2016, stima effettuata da CLS[1]. A quest'ultimo possono partecipare moltissime tipologie di istituzioni e gruppi finanziari, come grandi banche internazionali, centri di investimento e governi statali. Negli ultimi anni però, anche privati e individui singoli hanno la possibilità di esporsi nel mercato per mezzo dei *broker forex*[2]. A differenza del mercato azionario, il mercato del Forex è attivo 24 ore su 24 per 5 giorni

a settimana, chiudendo nei week-end.

Vediamo ora quali sono i concetti fondamentali legati a questa tipologia di mercato, concentrandosi maggiormente su quelli inerenti all'ambito di questa tesi.

### 2.1.1 Tasso di cambio e volume

In ambito finanziario il "tasso di cambio" tra due valute corrisponde al valore unitario con cui una valuta può essere scambiata con l'altra. Questo valore è influenzato da una grande quantità di fattori, tra cui: tasso di inflazione, debito pubblico, stabilità politica e investimenti da parte delle nazioni più autorevoli. Considerando la variazione nel tempo del tasso di cambio, essa può essere considerata come una serie temporale in cui a ogni timestep $t$ corrisponde un valore del tasso di cambio (Figura 2.2). L'influenza da parte di numerosi fattori fa si che l'andamento di qualsiasi serie temporale che descrive l'andamento di un tasso di cambio sia estremamente imprevedibile e la predizione di quest'ultimo con un'elevata accuratezza risulta ancora una sfida aperta.

Le principali coppie monetarie scambiate sono:

- EUR/USD (Euro/Dollaro statunitense)
- USD/JPY (Dollaro statunitense/Yen giapponese)
- GBP/USD (Sterlina britannica/Dollaro statunitense)
- USD/CHF (Dollaro statunitense/Franco svizzero)

La coppia EUR/USD è la più negoziata globalmente, per questo motivo si è scelto di analizzare l'andamento temporale di quest'ultima.

La notazione con cui si indica il tasso di cambio di una coppia di valute è la seguente:

- EUR/USD = 1,0747

Nella coppia di valute la prima moneta rappresentata, in questo caso l'euro, è chiamata "valuta base", mentre la seconda moneta rappresentata, in questo caso il dollaro statunitense, è chiamata "valuta quota". Il tasso di cambio è rappresentato da un numero che indica il valore assunto dalla valuta base rispetto alla valuta quota. Nel caso dell'esempio precedente è possibile vendere un'unità di valuta base (EUR) per ricevere 1,0747 unità di valuta quota (USD).

[1]CLS: istituzione finanziaria molto importante nell'ambito del Forex che si occupa della gestione di transazioni, riducendo il rischio legato alla mancata ricevuta da parte di una delle due controparti della valuta acquistata.

[2]Broker forex: Intermediario di titoli, ente che si occupa di interporsi tra i piccoli investitori e il mercato del Forex, permettendo a essi di operare sul mercato.

Ogni giorno vengono compiute a livello mondiale un grande numero operazioni di compra/vendita per ogni coppia monetaria. Si definisce "volume" del tasso di cambio per una certa coppia di valute la somma, per un periodo di tempo definito (e.g. un giorno) della quantità di denaro scambiato nelle singole transazioni compiute.

### 2.1.2 Posizioni e compra/vendita

Nel mercato del Forex, nel momento in cui un partecipante compra o vende una coppia monetaria a un certo tasso di cambio, egli sta cambiando la sua "posizione" rispetto al mercato. Quest'ultima descrive qual'è la variazione del tasso di cambio che meglio permetterebbe di ricavare guadagno. In particolare, quando un partecipante decide di entrare nel mercato comprando la coppia, esso sta assumendo una posizione *long*, sperando che la valuta base (la prima della coppia) acquisisca maggiore valore rispetto alla valuta quota (la seconda della coppia). Di conseguenza una posizione *long* può generare profitto nel caso in cui il tasso di cambio di una coppia aumenti. Al contrario, quando un partecipante decide entrare nel mercato vendendo la coppia, egli sta assumendo una posizione *short* rispetto al mercato, sperando che la valuta venduta (la prima della coppia) perda valore rispetto alla valuta comprata (la seconda della coppia). Di conseguenza una posizione *short* può generare profitto nel caso in cui il tasso di cambio di una coppia diminuisca. Un partecipante che decide di non compiere nessuna operazione sul mercato, sta assumendo una posizione *flat*, che non comporta guadagni o perdite. Un partecipante che decide di cambiare la sua posizione, andando in *long* o *short*, si dice che sta "aprendo" una posizione. Nel momento in cui si voglia tornare a una posizione *flat*, e quindi definire il guadagno o la perdita, si dice che il partecipante sta "chiudendo" la posizione. Il momento della chiusura della posizione corrisponde sempre a un guadagno o una perdita nel caso in cui nel mercato sia avvenuta una variazione del tasso di cambio della moneta trattata in apertura della posizione.

### 2.1.3 Pip e spread

Nel mondo del Forex Trading l'unità di misura utilizzata per calcolare i movimenti che una valuta compie rispetto a un'altra è il *pip*, o "*percentage in point*". In particolare, il valore del pip è la più piccola unità di prezzo di una valuta ed è settato, nella maggior parte delle coppie monetarie, a 0,0001 rispetto al valore unitario della valuta stessa. Solamente nel caso dello Yen Giapponese (JPY) questo valore è settato a 0,001. Ad esempio, nel caso in cui il tasso di cambio della coppia EUR/USD passi da 1,0747 a 1,0757 avremo un guadagno della valuta base di 10 pip rispetto alla valuta quota. Questo valore rappresenta la variazione unitaria della quarta cifra decimale.
Il pip è estremamente importante perché permette di avere una unità di misura naturale valida per tutte le coppie di valute e risulta essere la componente principale di calcolo del guadagno al momento della chiusura di una posizione.

Nel momento in cui un partecipante al mercato del Forex voglia esporsi si troverà davanti a due tassi di cambio: il prezzo di *bid* e il prezzo di *ask*. Con prezzo di bid si intende il prezzo al quale è possibile vendere la valuta base. Con prezzo di ask si intende il prezzo al quale invece è possibile acquistarla. Si definisce *spread* la differenza tra il prezzo di ask e il prezzo di bid ed è possibile esprimerlo in pip.

Esempio:

| | bid | ask |
|---|---|---|
| EUR/USD | 1,07479 | 1,07576 |

Figura 2.1: Bid e ask nel Forex Trading

In questo caso lo spread equivale a 1,07576-1,07479=0,00097, cioè 9,7 pip. Questo significa che, nel caso in cui si voglia vendere l'euro e comprare il dollaro, si riceveranno $1,07479 per €1. Effettuando immediatamente lo scambio contrario e vendendo quindi $1 per comprare il corrispondente in euro si riceverebbero soltanto 1,07479/1,07576=€0,9991. Si può notare che semplicemente comprando e vendendo, mantenendo gli stessi prezzi di bid e ask, si ha effettivamente una perdita di €0,0009. Lo spread è estremamente importante nel Forex poiché stabilisce quant'è la perdita immediata derivata solo dall'azione di apertura di una singola posizione. La variazione tra prezzo di bid e prezzo di ask varia a seconda della coppia di valute e dal broker a cui si fa affidamento. Il fattore che incide maggiormente da questo punto di vista è la liquidità del mercato. La liquidità indica la quantità di movimento sul mercato da parte di compratori e venditori. In particolare i mercati più "liquidi" sono caratterizzati da un maggiore volume di scambio tra i partecipanti e sono generalmente più sicuri rispetto a mercati meno liquidi. Ne consegue che a un mercato più liquido corrisponde uno spread minore e quindi minori costi dovuti alla sola apertura/chiusura della posizione. Essendo il Forex il mercato con i volumi di scambio maggiori al mondo, generalmente lo spread non risulta molto alto, soprattutto per quanto riguarda le valute che subiscono investimenti più consistenti.

### 2.1.4 Lotti, leva e margine

Per negoziare sono imposti dei limiti di contratto, che intendono stabilire la dimensione minima di valuta base con cui è possibile immettersi nel mercato. La dimensione standard di un contratto (e quindi di un lotto) è di 100.000 unità della valuta base. Questa cifra è difficilmente raggiungibile da un trader privato, che quindi necessita di un aiuto da parte di un broker per poter entrare sul mercato senza mettere a disposizione una cifra minima pari a quella di un lotto.
La leva finanziaria è lo strumento di cui ci si avvale per consentire questo tipo di operazioni. Essa consiste in un prestito da parte del broker che offre una elevata esposizione

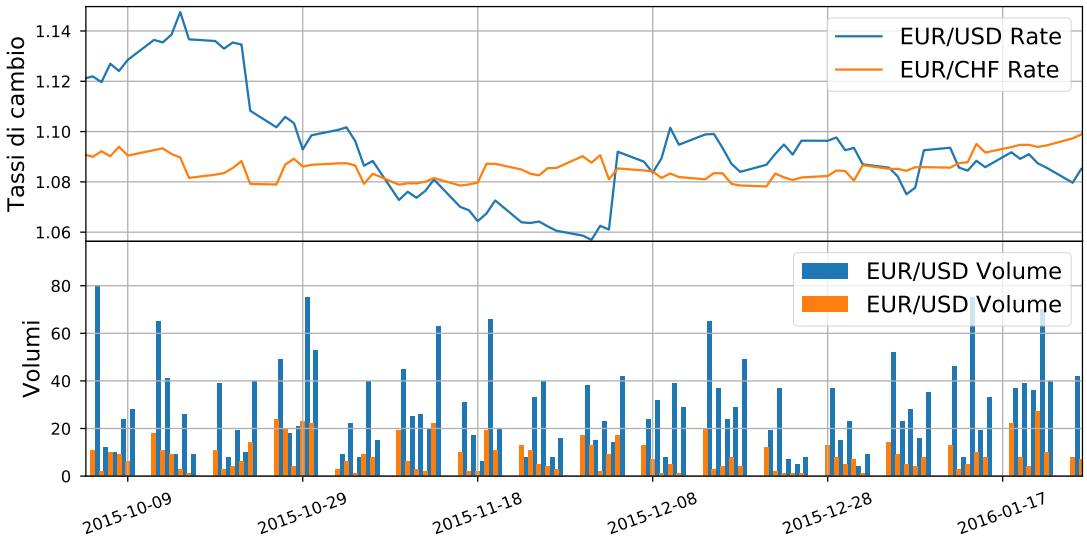



Figura 2.2: Andamento EUR/USD e EUR/CHF

Rappresentazione grafica degli andamenti del tasso Euro/Dollaro statunitense (in blu) ed Euro/Franco Svizzero (in arancio) nella parte superiore e dei volumi di scambio delle valute Euro/Dollaro statunitense (in blu) ed Euro/Franco Svizzero (in arancio) nella parte inferiore.

sul mercato a fronte di un investimento moderato di capitale. Ad esempio, considerando una leva 200:1, sarebbe possibile investire $500 avendo una esposizione sul mercato esattamente pari a quella di un lotto, cioè $100.000. La differenza che intercorre fra la somma investita e la somma con cui ci si espone sul mercato corrisponde alla quantità di denaro prestata dal broker. La leva risulta quindi un ottimo strumento per mettere a disposizione del trader un moltiplicatore di guadagno pari al valore della leva, e.g. con una leva di 200:1 un guadagno dell'1% sull'investimento si trasforma in un guadagno del 200%. Sfortunatamente questo vale anche nel caso contrario, poiché la moltiplicazione riguarda anche le possibili perdite.

All'apertura di un conto il broker chiederà un margine iniziale utilizzato per aprire le prime posizioni. Il margine consiste in una somma di denaro che deve essere presente sul conto del trader per continuare a operare sul mercato. Questo margine permette al broker di coprire una eventuale perdita. Se, ad esempio, come nel caso precedente, un trader volesse esporsi sul mercato di $100.000 utilizzando una leva 200:1, esso dovrebbe depositare una somma di almeno $500 corrispondente appunto al margine. Nel caso in cui vi sia una

perdita dello 0,5% sull'investimento, corrispondente alla somma che il trader ha depositato come margine, il broker sarà costretto a chiudere la posizione del trader evitando perdite non coperte dal margine depositato. Da questo si capisce quanto la combinazione tra leva e margine sia un'arma a doppio taglio all'interno del Forex trading, in quanto maggiore è la leva utilizzata, maggiore è il rischio che si corre nell'investimento prendendo in considerazione anche piccole variazioni di mercato.

### 2.1.5 Algorithmic Trading

Con *Algorithmic Trading* si intende il processo di utilizzare programmi automatizzati che possano seguire un determinato set di istruzioni in modo da decidere automaticamente quando eseguire una certa azione all'interno de mercato. Questo permette di operare e generare profitto in modo completamente autonomo a velocità irraggiungibili da parte di un investitore umano. I set di regole sono formate da osservazioni effettuate generalmente a priori da parte dei trader professionisti e si basano su modelli matematici applicati a volumi, tassi di cambio, indici, indicatori e tempo.

L'algorithmic trading si basa sulla Analisi Tecnica[1] del mercato finanziario, che consiste nell'insieme di tecniche analitiche che pretendono di essere in grado di prevedere gli andamenti dei tassi di cambio nel tempo. All'interno dell'analisi tecnica vengono fatte tre assunzioni principali:

- i tassi di cambio e i volumi di scambio riflettono l'andamento del mercato;
- la storia passata riguardante il mercato si suppone che si ripeta in futuro;
- i prezzi si muovono seguendo specifici trend.

L'algorithmic trading è utile per svariate ragioni, infatti esso permette di verificare simultaneamente una grande quantità di indicatori considerando dati anche molto distanti a livello temporale oppure real-time. Inoltre è possibile compiere azioni opportune nel momento in cui le regole decisionali definiscono la migliore possibilità di guadagno, escludendo totalmente l'insorgere di errori umani dovuti a valutazioni errate o fattori psicologici. All'interno dell'algorithmic trading si possono inserire tutte le tecniche di machine learning applicate a mercati azionari o al Forex Trading, compreso il progetto legato a questa tesi.

### 2.1.6 Teoria del Random Walk

Esiste però una corrente di pensiero che critica ampiamente l'utilizzo dell'analisi tecnica dei mercati finanziari come strumento per trarre profitto da essi. Tale teoria afferma che la variazione dei prezzi nel tempo non appartiene ad alcuna correlazione sequenziale, ma, al

[1] Analisi tecnica: studio di dati storici e grafici riguardanti l'andamento nel mercato, utile a guidare gli investimenti dei trader.

contrario, essa è da considerare totalmente casuale. Una *random walk* rappresenta sostanzialmente un percorso di una variabile lungo il tempo che esprime pattern non predicibili. In particolare, data una serie di prezzi $y(t)$ che si muove su un random walk, il valore di $y(t)$ in un qualsiasi istante di tempo equivale al valore al periodo precedente $y(t-1)$ a cui si aggiunge una variabile totalmente random $\eta(t)$:

$$y(t) = y(t-1) + \eta(t) \tag{2.1}$$

La variabile $\eta(t)$ è spesso conosciuta con il nome di rumore bianco o *white noise*, e ha la caratteristica di rappresentare una serie a media nulla, $\mu(t) = 0$, e varianza finita $\sigma^2(t)$.

Questo equivale a dire che l'andamento futuro dei mercati è totalmente indipendente dal passato e non è in alcun modo prevedibile. Secondo i sostenitori di questa teoria, il guadagno degli investitori di maggiore successo non è dato da una loro abilità nella previsione dei trend di mercato futuri, ma dalla probabilità che su milioni di investitori ne debbano esistere alcuni che casualmente effettuano le scelte giuste d'investimento. Studi più recenti sui mercati finanziari però si oppongono a questo punto di vista, constatando il fatto che i mercati azionari non seguono l'ipotesi del cammino casuale.

### 2.1.7 Dati a disposizione

I dati messi a disposizione dall'azienda riguardano serie temporali di tassi di cambio e volumi di scambio, campionati minuto per minuto.

In particolare sono presenti tassi di cambio e volumi delle seguenti coppie di valute:

- EUR/USD: tasso di cambio per le valute Euro/Dollaro statunitense
- EUR/CHF: tasso di cambio per le valute Euro/Franco Svizzero
- XAU/EUR: tasso di cambio Oncia d'oro/Euro

Altri indici utilizzati di particolare rilevanza:

- JP255: indice di un segmento della borsa di Tokyo valutato in Yen.
- WTICO: indice del prezzo del petrolio al barile valutato in Dollari Americani.
- SPX500: è il più importante indice azionario americano. Esso contiene 500 titoli azionari di altrettante società statunitensi a maggiore capitalizzazione.
- HK33: più importante indice azionario della borsa di Hong Kong, costituito dalle 50 società più quotate.

## 2.2 Machine Learning

Il Machine Learning è un campo di studi appartenente all'ambito della *computer science* e dell'intelligenza artificiale che studia come programmare i computer affinché possano imparare a risolvere un determinato problema. Di conseguenza un algoritmo di machine learning, invece di seguire determinate routines o serie di istruzioni codificate a mano, deve accettare dei dati in input ed estrarre una particolare forma di conoscenza da essi. Dando uno sguardo alla curva dell'*Hype Cycle* della Gartner (fig. 2.4), che mira a rappresentare graficamente le fasi di maturità, adozione e applicazione di varie tecnologie, si può notare come il Machine Learning sia posizionato nella parte più alta della curva, che illustra la fase di massimo interesse nel mondo tecnologico.

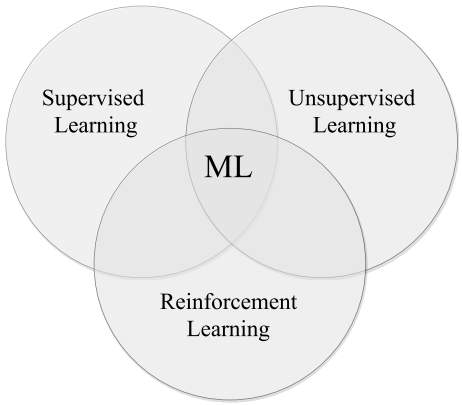



Figura 2.3: Machine Learning branches

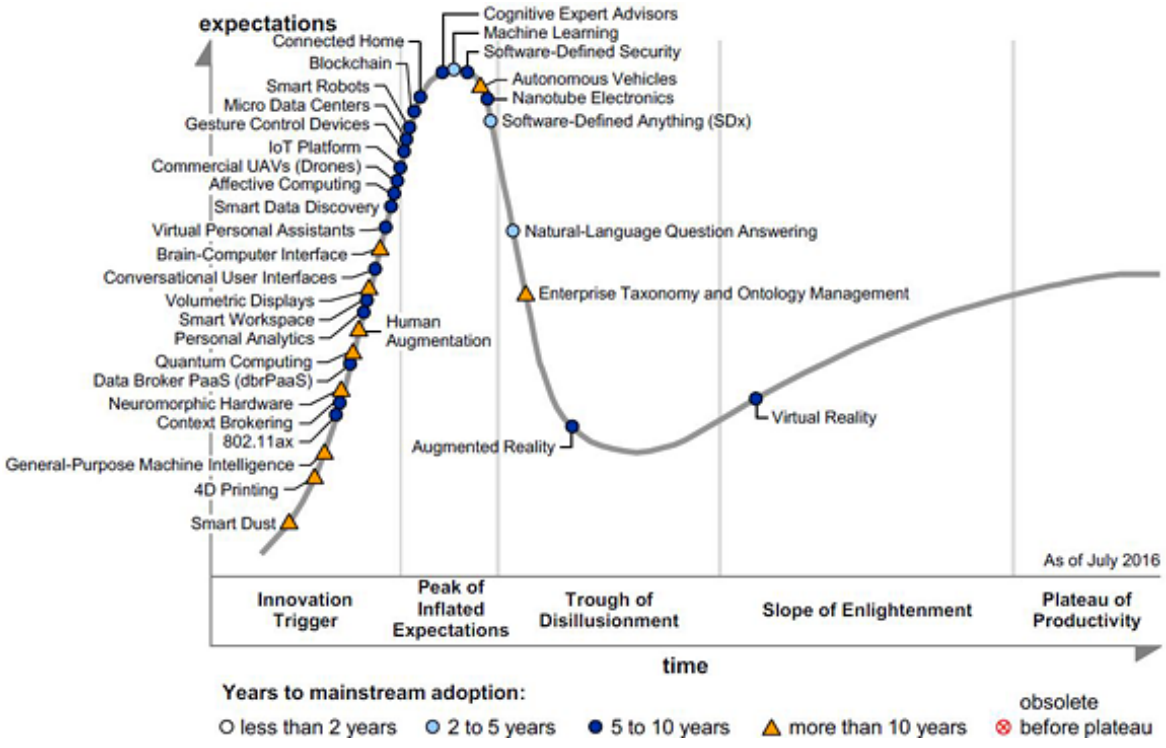



Figura 2.4: Gartner's 2016 Hype cycle

### 2.2.1 Supervised, unsupervised e reinforcement learning

Le modalità di apprendimento di un algoritmo di Machine learning si possono dividere in tre gruppi principali: apprendimento supervisionato (*Supervised learning*), apprendimento non supervisionato (*Unsupervised learning*) e apprendimento per rinforzo (*Reinforcement learning*).

Per quanto riguarda l'apprendimento supervisionato vengono messi a disposizione un insieme di coppie di dati $(x, y)$, con $x \in X$ e $y \in Y$. $X$ corrisponde all'insieme degli esempi e $Y$ corrisponde all'insieme delle classi o dei valori associabili agli esempi appartenenti a $X$. Ogni dato di input è caratterizzato da valori, tipicamente numerici, chiamate *features*, che descrivono in modo quantitativo le caratteristiche di un dato esempio. Lo scopo di un algoritmo di *supervised learning* è quello di trovare una funzione $y = f(x)$ che meglio rappresenti la mappatura tra i dati di input $(x)$ e i corrispondenti dati di output $(y)$. Per raggiungere questo scopo si utilizzano opportune funzioni di costo che rappresentano la differenza tra i valori predetti dall'algoritmo e i valori reali. Minimizzare la funzione di costo significa quindi ridurre la differenza tra i risultati dell'algoritmo e i valori reali corrispondenti. Terminata la fase di *training* la funzione $f$ può essere utilizzata per estrarre valori inferiti da dati mai visti in precedenza dall'algoritmo (i.e. dati di test). I risultati ottenuti possono essere precisamente valutati tramite misure di performance, adattate alla tipologia di problema che si sta affrontando. I problemi di supervised learning a loro volta si possono suddividere in due macro-categorie: problemi di classificazione, nei quali lo scopo è predire la classe di appartenenza dei dati di input, e i problemi di regressione, nei quali invece di predire valori discreti rappresentanti le classi si predicono valori numerici appartenenti a uno spettro continuo.

L'apprendimento di tipo non supervisionato si pone invece un problema molto differente. In questo caso sono forniti una serie di dati di input $x \in X$ e una funzione di costo variante a seconda dell'obiettivo finale. Lo scopo in questo caso è estrarre una struttura nascosta presente nei dati senza necessariamente fornire all'algoritmo una conoscenza a priori del risultato che si vuole ottenere. L'output di un algoritmo di questo tipo può essere, ad esempio, il raggruppamento di dati con caratteristiche simili, la scoperta di esempi che si discostano dalla normale distribuzione dei dati oppure la riduzione dello spazio dimensionale di rappresentazione delle features.

Nel *reinforcement learning* il paradigma d'apprendimento cambia totalmente. Lo scopo è istruire un agente in grado di operare in un ambiente ESTERNO e interagire con esso in modo da ricavare il maggiore guadagno possibile. L'interazione dell'agente con l'ambiente può essere descritta in questo modo: in ogni punto nel tempo $t$, l'agente compie una certa azione $a_t \in A$, dove A rappresenta l'insieme delle possibili azioni, e l'ambiente genera una osservazione $x_{t+1}$ e un reward $r_{t+1}$. L'obiettivo è trovare una certa *policy* $\pi$ di selezione delle azioni che massimizzi il reward totale. Questa branca del machine learning sarà ripresa ampiamente nel terzo capitolo essendo anche base dell'algoritmo studiato per l'applicazione nel mondo del Forex.

## 2.3 Deep learning

Con il termine *Deep learning* si intendono una serie di tecniche e algoritmi afferenti alla branca dell'Intelligenza artificiale, che possono essere inseriti all'interno della macrocategoria del Machine learning. I modelli matematici utilizzati sono chiamati Reti Neurali, concepite e strutturate come metafora del cervello umano, ogni loro parte è riconducibile a una elemento appartenente a esso. In questa sezione si presenteranno la struttura di base delle reti neurali, le motivazioni che portano al loro utilizzo e le principali tipologie e applicazioni.

### 2.3.1 Reti neurali

Spesso si pensa che l'adozione delle reti neurali sia avvenuta in uno periodo di tempo molto recente. In realtà l'inizio della ricerca in questo campo di studi risale al 1943, quando i neuropsicologi Warren McCulloch e Walter Pitts[11] crearono il primo modello computazionale per descrivere il possibile funzionamento dei neuroni attraverso l'utilizzo di semplici strutture matematiche mediante funzioni binarie. Successivamente si ebbe la creazione del *perceptron* (Figura 2.5) da parte di Frank Rosenblatt[13]. Il percettrone nasce come un classificatore binario che associa a un dato input un valore di output binario tramite una semplice funzione lineare. Tra gli anni '60 e '80 ci fu una fase di stallo durante la quale calò l'attenzione riguardo l'ambito delle reti neurali artificiali. Questo perché non si era ancora trovato un modo con cui istruire la rete, modificando stabilmente i parametri in modo da risolvere determinati problemi. La maggiore complicazione riguardava l'impossibilità di risoluzione di problemi di tipo non lineare. Nel 1974 Paul Werbos sviluppò e utilizzò il metodo di allenamento chiamato *backpropagation*. Con esso si riuscì per la prima volta a risolvere il problema dell'*exclusive-or* tramite l'utilizzo di reti neurali chiamate *multi-layer perceptron* o MLP.

Dagli anni '80 è stata portata avanti la ricerca in questo ambito e l'adozione delle reti neurali a oggi è più forte che mai. Considerando che per l'allenamento di queste ultime vi è bisogno di grande potenza computazionale, la motivazione principale che porta all'esplosione di utilizzo di questa tipologia di algoritmi è attribuibile proprio grande crescita in quest'ambito, che ora permette di poter allenare *artificial neural network*, o ANNs, estremamente più complesse rispetto alle prime versioni ed essendo dunque adattabili alla risoluzione di svariate tipologie di problemi.

Oltre a questo tipo di impedimento, un fattore essenziale per l'utilizzo delle ANNs è la possibilità di utilizzo di una grandissima quantità di dati per l'allenamento della rete. a oggi anche questo inconveniente è stato superato in molti domini differenti grazie al crescente interesse nel riutilizzo dei dati e la disponibilità di grande spazio di archiviazione.

Ciò che rende molto interessanti le reti neurali è la capacità di ricavare dai dati, spesso

imprecisi o molto complessi, una serie di pattern che spesso non sono estraibili nemmeno dagli esseri umani. Questo accade poiché ogni livello intermedio che compone la rete neurale ha la caratteristica di poter estrapolare da ogni singola feature in input una rappresentazione della stessa a un livello di astrazione maggiore, combinandola con le altre e ricavando importati informazioni in vista della risoluzione del problema. Questa capacità di astrazione fornisce intrinsecamente alla rete un ottima abilità di adattamento, modificando i propri parametri in modo da risolvere al meglio a qualsiasi tipo di situazione.

### 2.3.2 Struttura e componenti principali

L'elemento chiave che rende le reti neurali estremamente interessanti è il modo in cui esse processano le informazioni derivante dalla loro particolare struttura. Il perceptron è la componente che rappresenta il neurone. Essa non è altro che una tipologia di classificatore binario che riceve in ingresso un numero $n$ di input $I$ opportunamente pesati tramite variabili $w$ chiamate pesi, e mappa in uscita il corrispondente valore $y$.

Questa funzione è esprimibile attraverso la formula 2.2:

$$y(x_i) = f(\sum_{j=1}^{n} I_{ij} * w_j) \tag{2.2}$$

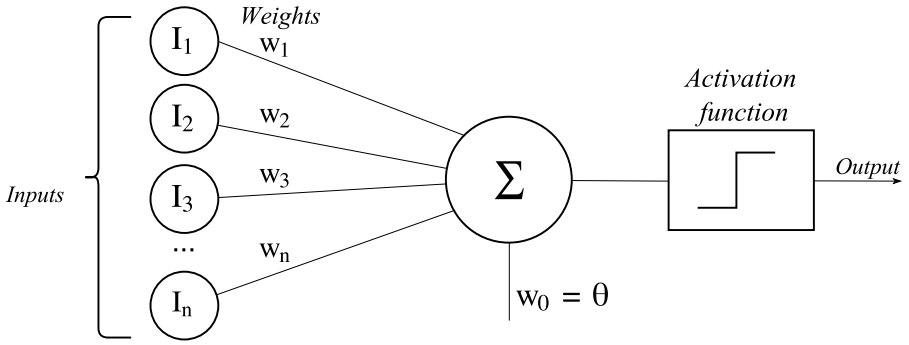



Figura 2.5: Perceptron

dove $x_i$ è l'input $i$-esimo, $I_{ij} \in I_i$, dove $I_i$ è l'insieme delle *features* del corrispondente input $X_i$, $w_j \in W$, con $W$ corrispondente all'insieme dei pesi dove $w_0 = \theta$ è chiamato bias e corrisponde a una variabile di correzione.

La funzione $f$ viene detta funzione di attivazione. Essa modifica il risultato della sommatoria ottenendo l'output del perceptron. Possono essere utilizzate moltissime tipologie di funzioni di attivazione, lineari e non lineari, a seconda della tipologia di problema da risolvere e della forma degli input.

L'insieme di più perceptron forma un layer. Esso corrisponde a una funzione $f(x)$ avente in ingresso $n$ input $I$ e in uscita un numero di output pari al numero di neuroni che compongono il layer finale. Infine i layer possono essere combinati per costruire una rete neurale completa, avente $n$ input $I$ che compongono l'*input layer*, $k$ *hidden layer* formati da un numero variabile di neuroni e un *output layer* costituito da un numero di output pari al numero di neuroni costituente il layer stesso. La rete neurale si dice *Deep* (profonda) nel momento in cui il numero si aumenta il numero di layer intermedi. La maggior parte delle neural networks è costituita da layer *fully connected*, il che significa che ogni unità appartenente a un layer è connessa a tutte le unità appartenenti ai due layer adiacenti.

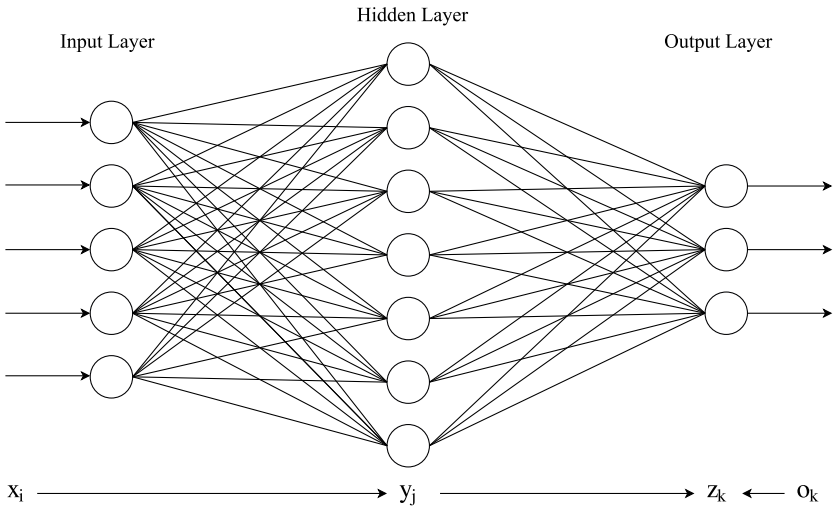



Figura 2.6: Simple artificial neural network

Questa struttura ha lo scopo di approssimare una qualche funzione $f$ in modo che l'output sia il più vicino possibile al risultato migliore. Ad esempio, nel caso della classificazione, la rete neurale ha lo scopo di trovare una funzione parametrizzata $f(x;\theta) = y$, dove $\theta$ rappresenta i pesi della rete, che mappi gli input $x$ con una opportuna classe $y$ in output.

# Capitolo 3

# Deep Reinforcement Learning Background

Tra le metodologie di intelligenza artificiale ricoprono un ruolo estremamente importante, soprattuto dopo gli ultimi anni di ricerca e innovazione, le tecniche di Deep Reinforcement learning. I grandi successi ottenuti contro gli esseri umani nei giochi da tavolo, gli altissimi punteggi ottenuti nel dominio dei giochi Atari e i recenti sviluppi in ambito di sistemi di controllo e robotica, hanno dimostrato le incredibili potenzialità di questa famiglia di algoritmi, in grado di scoprire efficaci strategie a lungo termine in modo completamente autonomo, senza essere direttamente programmati a eseguire particolari azioni per ogni configurazione dello stato di mercato. La distinzione fra gli algoritmi di Supervised Learning e quelli di Reinforcement learning è netta e rende questi ultimi certamente più adatti alle situazioni in cui è necessario applicare scelte decisionali aventi ripercussioni in un futuro non immediato.

In questo capitolo viene presentata la struttura del problema di Reinforcement Learning, gli algoritmi e le tecniche che hanno portato a una rivalutazione di queste metodologie negli ultimi anni.

## 3.1 Reinforcement learning

Il reinforcement learning è una particolare branca del machine learning che studia la scienza dei processi decisionali, in particolare cerca di capire qual'è la via ottimale per prendere le decisioni in un determinato contesto. Il paradigma di apprendimento dell'algoritmo è differente dalle metodologie più comuni, come il supervised o unsupervised learning. In questo caso un agente viene programmato come se fosse un umano, che deve imparare attraverso un meccanismo di *trial-and-error*, i.e. tentativi ed errori, in modo da poter trovare la migliore strategia che gli possa permettere di raggiungere il risultato migliore in termini di reward a lungo termine.

Conseguendo incredibili risultati nell'ambito dei giochi (digitali e da tavolo) e del controllo automatico di robot, il Reinforcement learning è ancora ampiamente studiato tanto che, nell'ultima decade, si è deciso di aggiungere una componente fondamentale: le reti neurali. Questa integrazione fra Reinforcement learning e le Deep neural networks, chiamata Deep reinforcement learning, ha permesso ai ricercatori di Google DeepMind di raggiungere risultati stupefacenti in ambiti prima ancora inesplorati. In particolare nel 2013 è stato possibile, tramite l'algoritmo di Deep Q-Learning (spiegato in dettaglio nella Sezione 3.2), raggiungere le performance di giocatori umani esperti nel dominio dei giochi Atari[18][19] prendendo come dati di input i pixel che rappresentavano la schermata di gioco, ponendo l'agente esattamente nelle condizioni di un essere umano. Un'altra conquista estremamente importante è giunta nell'Ottobre del 2015, quando il medesimo laboratorio di ricerca è riuscito, tramite la stessa famiglia di algoritmi, a battere il campione europeo di Go[15], gioco da tavolo Cinese di grande complessità, conseguendo infine alla vittoria contro il campione mondiale a Marzo del 2016.

La notazione utilizzata nell'ambito di questa tesi corrisponde a quella presente nella seconda edizione del libro "Reinforcement Learning: An Introduction"[16](Sutton e Barto, 2017), ottima fonte di riferimento nell'ambito del RL. Nel momento della scrittura di questa tesi risulta inoltre disponibile un ottimo corso online tenuto da David Silver, ricercatore che attualmente lavora in Google DeepMind nell'ambito dell'Intelligenza Artificiale.

### 3.1.1 Struttura del problema

Il Reinforcement Learning si discosta molto dai problemi di tipo *supervised learning*. Mentre quest'ultima tipologia si basa sull'allenamento dell'algoritmo presentando dati di training $(x_0, x_1, x_2..., x_n) \in X$ e un set di label $(y_0, y_1, x_2, ..., y_n) \in Y$ associate esplicitamente a essi in modo che l'algoritmo possa imparare esattamente la mappatura delle associazioni corrette a ogni dato di input, il problema di Reinforcement Learning non prevede nessuna associazione fra i dati in entrata e le corrette azioni da compiere, di conseguenza la struttura di apprendimento risulta completamente differente.

Il problema di RL ha come concetto principale la presenza di due componenti che interagiscono fra di loro: un *agente* e un *environment* (i.e. ambiente). Un agente di Reinforcement learning impara a prendere decisioni all'interno di un ambiente a esso sconosciuto compiendo una serie di azioni e ottenendo dei reward numerici associati a esse. Accumulando esperienza tramite un processo di *trial and error* l'agente impara quali sono le azioni migliori da compiere a seconda dello stato in cui si trova, definito dall'ambiente e dalla serie di azioni compiute in precedenza. L'agente ha la possibilità di intuire quali sono le mosse di maggiore successo semplicemente valutando il reward ottenuto, e adatta la propria *policy*, i.e. metodologia di scelta delle azioni, in modo da ottenere il massimo reward cumulativo nel tempo.

Il modello di Reinforcement learning è formato da:

- un set di stati $(s_0, s_1, s_2, ..., s_n) \in S$, definito dall'interazione fra l'environment e l'agente;
- un set di possibili azioni $(a_0, a_1, a_2, ..., a_m) \in A$, selezionate opportunamente dall'agente in base allo stato di input;
- un reward $r \in R$ associato a ogni interazione tra l'environment e l'agente;
- una *policy* che mappa ogni stato $s_t$ in input a una azione $a_t$ in output;
- una serie di funzioni chiamate *state-value function* e *action-value function* che determinano rispettivamente il valore dello stato in cui si trova l'agente in un certo momento e il valore che ha compiere una determinata azione da parte dell'agente in un dato momento.

Un agente di RL interagisce con l'environment in un certo istante di tempo $t$. Ad ogni $t$ l'agente riceve in input uno stato $s_t \in S$ e un reward $r_t$. In base a essi l'agente determina l'azione $a_t \in A(s_t)$ da compiere, dove $A(s_t)$ rappresenta l'insieme delle possibili azioni possibili in un certo stato $s_t$. Quest'ultima viene ricevuta dall'environment che la "processa" ed elabora un nuovo stato $s_{t+1}$ e un nuovo segnale di reward $r_{t+1}$, corrispondente al successivo input dell'agente nel periodo di tempo $t + 1$.

Questo processo, considerato in modo ricorsivo, genera l'algoritmo di apprendimento dell'agente di Reinforcement Learning. L'obiettivo dell'agente è guadagnare il più possibile in termini di reward cumulativo finale. Lo scopo può essere raggiunto utilizzando diverse metodologie. L'agente, durante l'allenamento, è in grado di imparare opportune strategie che gli permettono di ottenere un maggior guadagno immediato oppure avere un guadagno maggiore a lungo termine a discapito dei reward immediati.

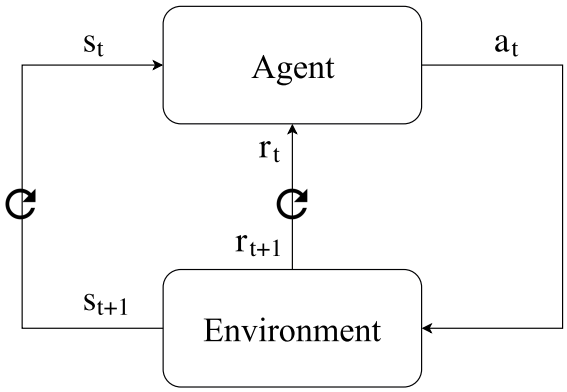



Figura 3.1: Interazione agente-environment nel Reinforcement learning

### 3.1.2 Markov Decision Process (MDP)

Tutti i problemi di Reinforcement Learning possono essere modellati attraverso i Processi decisionali di Markov, tramite cui è possibile descrivere l'evoluzione dell'environment nel tempo, con l'assunzione che esso sia completamente osservabile, i.e. lo stato precedente caratterizza in modo esaustivo tutto il processo passato. Quest'ultima proprietà prende il nome di Proprietà di Markov e formalmente può essere riscritta come segue:

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, S_2, ..., S_t] \tag{3.1}$$

Questa proprietà è estremamente importante poiché ci permette di assumere che la dinamica dell'ambiente sia completamente determinata dallo stato attuale e dalle possibili azioni che l'agente può effettuare. Da ciò deriva anche la possibilità di poter effettuare previsioni su ciò che l'ambiente potrebbe divenire considerando $n$-step futuri, in modo da poter evolvere ed elaborare strategie sia a breve che a lungo termine.

A questa proprietà è possibile associare la "probabilità di transazione di stato". Essa rappresenta, dato uno stato $s$ e una azione $a$, la probabilità di passare dallo stato $s$ allo stato successivo $s'$ compiendo una certa azione $a$:

$$P^a_{ss'} = \mathbb{P}[S_{t+1} = s'|S_t = s, a_t = a] \tag{3.2}$$

Un processo decisionale di Markov rappresenta un environment di reinforcement learning in cui tutti gli stati hanno la proprietà di Markov. In particolare, considerando per semplicità il caso discreto, esso può essere descritto con una tupla $< S, A, P, R, \gamma >$ in cui:

- $S$ è un set finito di stati
- $A$ è un set finito di azioni
- $P$ è una matrice di probabilità di transizione di stato,

$$P^a_{ss'} = \mathbb{P}[S_{t+1} = s'|S_t = s, A_t = a] \tag{3.3}$$

- $R$ è la funzione di reward,

$$R^a_s = \mathbb{E}[R_{t+1}|S_t = s, A_t = a] \tag{3.4}$$

- $\gamma$ è il fattore di sconto $\gamma \in [0,1]$

Anche se spesso vengono fatte queste assunzioni, non tutti i problemi di RL seguono completamente le regole definite nei processi di Markov. Il mondo del mercato azionario e

del Forex ne sono un esempio concreto poiché, data l'estrema imprevedibilità, non esiste una mappatura precisa della probabilità di transazione di stato. Risulta comunque utile approssimare queste tipologie di problemi con gli MDPs, per studiare il problema in modo più semplice applicando algoritmi di Reinforcement Learning e ottenendo comunque risultati di grande rilevanza.

### Reward

Il risultato di una azione compiuta dall'agente e l'inserimento di quest'ultima all'interno dell'environment viene espresso sotto forma numerica e chiamato *reward* $r \in R$, con l'assunzione che $|r_k| < \infty, \forall k < t$. Quest'ultimo fa parte dell'input successivo dell'agente assieme allo stato generato dall'ambiente. Il reward risulta una caratteristica essenziale di un algoritmo di Reinforcement Learning poiché formalizza il concetto di qualità della policy seguita dall'agente ed è parte integrante dell'obiettivo finale, esso rappresenta infatti una parte della quantità cumulativa che deve essere massimizzata.

Si può definire *return* (l'obiettivo da massimizzare) $G_t$ come il *discounted reward* totale a partire dal $t$-esimo timestep:

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1} \tag{3.5}$$

Ciò che si intende ottenere in un algoritmo di RL è la massimizzazione del guadagno finale, parametrizzato opportunamente di un fattore $\gamma$ applicato ai membri della sommatoria nella 3.5. Esso rappresenta il *discount rate*, i.e. fattore di sconto, assumendo $0 \leq \gamma \leq 1$. Con questa assunzione e considerando l'adozione di un fattore di peso pari a $\gamma^k$ con $k$ incrementale, avremo che se $0 < \gamma < 1$ allora questo stesso valore diminuirà in modo tendente allo 0 con $k \to +\infty$. Un reward ricevuto $k$ timestep nel futuro avrà solamente un valore di $\gamma^{k-1}$ volte quello che varrebbe se fosse ricevuto nell'immediato futuro. Considerando $\gamma = 0$ l'agente attribuisce la massima importanza ai reward immediati, mentre con $\gamma = 1$ l'agente attribuisce uguale importanza agli stati immediati e agli stati futuri, fino ad arrivare allo stato terminale. Generalmente si assume che $T = \infty$ oppure $\gamma = 1$, ma non contemporaneamente. Questo poiché il guadagno deve essere un numero finito. Facendo in questo modo è possibile evitare la computazione di un ciclo infinito all'interno dei Processi di Markov (descritti in seguito) ed evitare incertezza legata a troppi timestep nel futuro.

### Policy

Per interagire in modo adeguato con l'environment, l'agente deve avere un modo per poter prendere le decisioni opportune nel momento in cui riceve una particolare osservazione da

parte dell'environment ed elabora lo stato attuale. Per fare questo viene costruita una mappatura stato-azione, chiamata "policy". Essa è il cuore dell'agente di Reinforcement Learning e consiste in una funzione, denominata con $\pi(a|s)$, che determina completamente le scelte compiute dall'agente trovandosi nello stato $s$ al tempo $t$.

Formalmente essa definisce la probabilità per un agente di compiere una azione $a$ percependo lo stato $s$:

$$\pi(a|s) = \mathbb{P}[A_t = a|S_t = s] \tag{3.6}$$

La policy può essere deterministica o stocastica. Nel primo caso a ogni stato viene associata una azione corrispondente e univoca, nel secondo la mappatura consiste in una distribuzione di probabilità stato-azione. Essa è da considerare come una componente essenziale all'interno del RL, poiché l'obiettivo finale dell'agente è stabilire qual'è la mappatura che gli permette di massimizzare il reward cumulativo finale. Esistono vari metodi, i.e. algoritmi appartenenti alla famiglia del reinforcement learning, che permettono, durante l'apprendimento, di modificare la policy sulla base dell'interazione agente-environment tenendo in considerazione i reward ottenuti.

Chiarito il ruolo della policy è possibile ridefinire in modo ricorsivo la matrice di probabilità di transizione di stato e la funzione di reward in funzione della policy stessa.

Dato lo stato attuale $s$, lo stato successivo $s'$, un insieme di azioni possibili $A$, la matrice di transizione e la funzione di reward riferite a una policy $\pi(a|s)$ possono essere definite come segue:

$$P^{\pi}_{s,s'} = \sum_{a \in A} \pi(a|s) P^{a}_{ss'} \tag{3.7}$$

$$R^{\pi}_{s} = \sum_{a \in A} \pi(a|s) R^{a}_{s} \tag{3.8}$$

### Value functions

L'agente ha bisogno di poter calcolare, almeno in modo approssimativo, il valore che hanno i vari stati, e l'importanza che ha intraprendere le varie azioni trovandosi in un certo stato. Questi valori possono essere descritti tramite le funzioni di valore o *value function*. Vi sono due tipi di value function: la *state-value function*, $v_{\pi}(s)$, e la *action-value function*, $q_{\pi}(s, a)$.

La funzione di valore $v_{\pi}(s)$ rappresenta il valore a lungo termine dello stato $s$, calcolando il ritorno aspettato a partire da quello stato in poi. Più formalmente: il *valore* di uno stato $s$ considerando una policy $\pi$ è il ritorno aspettato partendo da $s$ e seguendo $\pi$ da quello stato in poi:

$$v_\pi(s) = \mathbb{E}[G_t|S_t = s] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s] \tag{3.9}$$

La *action-value function* rappresenta il valore di una coppia stato-azione $(s, a)$. Il valore di scegliere una azione $a$ in uno stato $s$ sotto una policy $\pi$, è il ritorno aspettato partendo dallo stato $s$, compiendo l'azione $a$ e seguendo la policy $\pi$. Essa si denota con $q_\pi(s, a)$ e si calcola come segue:

$$q_\pi(s, a) = \mathbb{E}[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s, A_t = a] \tag{3.10}$$

In un MDP in cui il numero di stati e azioni sono finiti e in numero limitato, è possibile utilizzare come funzione di valore una *lookup table*, che consiste semplicemente in una matrice in cui gli stati e le azioni corrispondono alle righe o alle colonne della stessa. In particolare mentre l'agente prende decisioni sulla base della policy $\pi$, si va ad aggiornare i valori della matrice mediando i valori delle esperienze passate in base reward ottenuti, agli stati esplorati e alle azioni compiute. Questo porterà a una futura convergenza alla vera e propria funzione di valore descritta dalla matrice di dimensioni finite. L'utilizzo della lookup table non è attuabile nel caso in cui gli spazi degli stati o delle azioni siano troppo grandi o addirittura spazino in un insieme reale. In questo caso le funzioni di valore vengono approssimate, utilizzando delle funzioni parametrizzate i cui parametri vengono aggiornati in modo da avvicinarsi al valore corrispondente di ritorno di un particolare stato o di una coppia stato-azione. Un particolare caso di approssimazione di funzione si ha nelle strategie di Deep Reinforcement Learning, in cui il ruolo di *function approximator* viene ricoperto dalle Deep Neural Networks.

### Model-free e Model-based reinforcement learning

Possiamo distinguere gli algoritmi di Reinforcement learning in due categorie differenti, algoritmi *Model-based* e algoritmi *Model-free*. In particolare la prima tipologia utilizza un concetto molto importante chiamato Modello. Quest'ultimo consiste in una rappresentazione dell'environment creata all'interno dell'agente attraverso l'esperienza e l'interazione con esso. Si può pensare al modello come una componente concettualmente divisa in due parti. Una parte è dedicata alle transizioni, essa indica in particolare quali sono le condizioni necessarie affinchè vi sia un passaggio da uno stato $s$ al tempo $t$ a uno stato $s'$ a un tempo $t + 1$. La seconda parte è dedicata al reward, in particolare il modello definisce quale reward è associato a una qualsiasi transizione di stato possibile nel sistema. Quando il modello è definito e completo, assumendo che quest'ultimo abbia una rappresentazione di tutto ciò che realmente rappresenta l'environment, l'agente è in grado di pianificare le

proprie mosse sulla base di simulazioni dell'environment reale basate sul modello in modo da ottenere le migliori performance in termini di reward finale.

Per quanto riguarda il model-free reinforcement learning, si ha un algoritmo che non tenta di costruirsi una rappresentazione interna dell'environment. In questo caso l'agente intende imparare le funzioni di valore direttamente dall'esperienza derivata dall'interazione con l'ambiente a ogni singolo step.

### 3.1.3 Bellman equation

L'equazione di Bellman esprime il valore della soluzione ottimale di un problema di ottimizzazione detto *dynamic programming*. Con questo termine si indica un metodo di risoluzione per problemi complessi, effettuandone la scomposizione in sotto-problemi più semplici, risolvendoli e salvando le soluzioni intermedie. La Bellman equation permette di riscrivere il valore di un problema decisionale da un certo tempo $t$ in poi considerando solamente le condizioni dello stato futuro e rimandando la risoluzione del problema allo stato successivo. Questo ci permette di decomporre le funzioni di valore definite precedentemente e riscriverle in modo ricorsivo, utilizzando come primo componente il reward immediato e come secondo componente la funzione di valore scontata allo stato successivo.

Considerando uno stato $s$, una policy $\pi$, un *discount factor* $\gamma$, una state-value function $v_\pi(s)$ e una una action-value function $q_\pi(s, a)$, la *Bellman Expectation Equation* per la state-value function può essere scritta come:

$$v_\pi(s) = \sum_{a \in A} \pi(a|s)(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s')) \tag{3.11}$$

In modo simile si può definire la *Bellman Expectation Equation* per l'action-value function:

$$q_\pi(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a', s') q_\pi(s', a') \tag{3.12}$$

Intuitivamente l'equazione di Bellman permette di arrivare a una stima del valore prendendo in considerazione separatamente il reward e la stima dei reward futuri. Questo è possibile poiché si fa una media di tutte le possibilità future pesate sulla base della probabilità che occorrano.

#### Optimal value function

Lo scopo degli algoritmi di Reinforcement Learning e in generale di tutti gli algoritmi di Machine Learning è quello di trovare la soluzione ottimale del problema. In particolare non ci interessa sapere qual'è il possibile reward futuro seguendo una policy non ottima,

quello che davvero interessa è trovare una funzione che ci indichi il miglior modo di agire all'interno del sistema. A questo scopo da ora in poi si considerano la *optimal state-value function* $v_*(s)$, che rappresenta il massimo valore di reward ottenibile seguendo tutte le possibili policy, e la *optimal action-value function* $q_*(s, a)$, che rappresenta la massima *action-value function* seguendo tutte le possibili policy. L'importanza di queste due funzioni deriva dal fatto che un processo decisionale di Markov si può considerare risolto nel momento in cui esse vengono scoperte.

Risulta importante sottolineare che per ogni MDP esiste una policy $\pi_*$ migliore o uguale a tutte le altre policy, $\pi_* \geq \pi, \forall\pi$. Da questo consegue che ogni policy $\pi$ ottimale permette di raggiungere la *optimal state-value function*, $v_{\pi_*}(s) = v_*(s)$ e la *optimal action-value function*, $q_{\pi_*}(s, a) = q_*(s, a)$.

## 3.2 Deep Q-Learning

Mentre i modelli di *supervised* e *unsupervised* learning sono stati adottati per la risoluzione di moltissime tipologie di problemi, sono stati sviluppati un buon numero di algoritmi di reinforcement learning anche se questa tecnica è rimasta maggiormente in ombra. Grazie ai recenti risultati ottenuti da Google DeepMind nel 2013 e nel 2016, prima riuscendo ad arrivare a livelli cosiddetti *superhuman* nell'ambito videoludico dei giochi Atari, battendo in seguito il campione mondiale di Go, il reinforcement learning torna a suscitare grandissimo interesse all'interno della comunità del Machine Learning. Questo rinnovato interesse si ha anche grazie all'avvento delle Deep neural networks (DNNs) come funzioni di approssimazione, portando le potenzialità di questa tipologia di algoritmi a un livello ancora più elevato. L'algoritmo che ha suscitato maggiore interesse negli ultimi tempi è sicuramente il Deep Q-Learning. Nella seguente sezione si introducono l'algoritmo di Deep Q-Learning e le tecniche di ottimizzazione per ricavarne il massimo delle prestazioni.

### 3.2.1 Q-Learning

Il Q-Learning è una tecnica di model-free reinforcement learning basata sul metodo a differenza temporale (*Temporal difference learning*). L'obiettivo principale è quello di approssimare direttamente la funzione $Q(s, a)$ (action-value function definita nella 3.9) ottimale per ogni $s \in S$ e ogni $a \in A$, in modo totalmente indipendente dalla policy che si sta seguendo al momento dell'aggiornamento del q-value. L'aggiornamento della action-value function che caratterizza il Q-Learning si basa sulla *Bellman Optimality equation* e permette di aggiornare i valori attuali prendendo in considerazione il reward al passo successivo e il massimo ritorno cumulativo che ci si aspetta dallo stato seguente, proseguendo nell'interazione con l'environment.

Formalmente, dato uno stato $s$ al tempo $t$, si compie una azione $a_t$ seguendo una policy di selezione delle azioni (e.g. $\epsilon$-greedy policy, spiegata nella Sezione 3.2.3) e si osservano il reward immediato $r_{t+1}$ e il nuovo stato $s_{t+1}$, dopo di che si aggiorna la action-value function attuale utilizzando la formulazione seguente:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[\overbrace{r_{t+1} + \gamma \underset{a'}{max} Q(s_{t+1}, a')}^{Bellman\ equation} - Q(s_t, a_t)] \tag{3.13}$$

I parametri utilizzati sono:

- $\alpha$: è un valore compreso tra 0 e 1 chiamato *learning-rate*. Rappresenta quanto velocemente i valori della action-value function devono essere aggiustati in direzione dell'errore commesso. Un valore vicino a 0 rappresenta un aggiornamento minimo del q-value attuale, un valore prossimo a 1 rappresenta un aggiornamento che va maggiormente a sovrascrivere il q-value precedente.

- $\gamma$: è il fattore di sconto, anch'esso compreso tra 0 e 1. Esprime la quantità di valore dell'action-value function dello step successivo rispetto al reward immediato. Con il valore di $\gamma$ vicino a 0 si dà maggiore importanza al reward immediato rispetto ai reward futuri, mentre una quantità man mano più prossima a 1 rappresenta un incremento del valore delle coppie stato-azione successive.

- $\underset{a'}{max}$: rappresenta il massimo reward ottenibile dallo stato seguente. $a'$ rappresenta quindi l'azione che massimizza $Q(s_{t+1}, a')$ con $a' \in A(s_{t+1})$, dove $A(s_{t+1})$ è l'insieme di tutte le azioni possibili a partire dallo stato $s_{t+1}$.

L'aggiornamento descritto precedentemente, applicato in maniera iterativa agli stati successivi, permette di trovare la policy di selezione stato-azione ottimale.

### 3.2.2 Function Approximator - Deep Q Network

Considerando un numero discreto e limitato di coppie stato-azione, la action-value function $Q(s, a)$ può essere rappresentata semplicemente con una matrice in cui ogni riga rappresenta i possibili stati e ogni colonna rappresenta le possibili azioni. I valori della matrice in questo caso corrispondono esattamente alla funzione $Q(s, a)$. L'algoritmo base di Q-Learning presenta un grandissimo problema nel momento in cui numero di stati e azioni possibili aumenta e diventa ingestibile dal punto di vista matriciale.

Si pensi solo alla configurazione degli input nel caso della struttura utilizzata da Google per avere grandi prestazioni nell'ambito dei giochi Atari. Lo spazio degli stati è discreto, ma il numero di possibili stati è enorme. Le immagini di input vengono scalate a 84x84 pixel resi in scala di grigi con valori che variano tra [0,255]. Di conseguenza i possibili stati

risultano essere $256^{84*84} = 256^{7056}$. Un numero elevatissimo impossibile da contenere in un'unica matrice in costante aggiornamento. Se si considerano poi il numero di possibili Q-value, esso dipenderà dal numero di azioni possibili. Considerando, ad esempio, $n$ azioni, i Q-value possibili saranno $256^{7056} * 3$.

Ciò che è stato osservato in precedenza accade anche nel caso in cui si consideri uno spazio degli stati e delle azioni continuo, in cui ogni stato può assumere un valore numerico definito in $\mathbb{R}$. In questo caso non solo è impossibile mantenere una matrice poiché si necessita di un grande spazio di memorizzazione, ma poiché non è possibile memorizzare ogni singolo valore.

Per questo motivo si rende necessaria una nuova possibile rappresentazione delle value function, chiamata *Value function approximator*.

Una funzione di approssimazione può essere una qualsiasi funzione parametrizzata, i cui parametri sono indicati con $\theta$ nella formulazione seguente.

$$\widehat{v}(s;\theta) \approx v_\pi(s) \tag{3.14}$$

$$\widehat{q}(s,a;\theta) \approx q_\pi(s,a) \tag{3.15}$$

Un approssimatore di funzione quindi è un modo di rappresentare in forma compatta ciò che prima era una matrice di valori in uno spazio discreto.

Questo tipo di funzione aiuta per due aspetti fondamentali: dal punto di vista della memoria e dal punto di vista della generalizzazione dell'algoritmo. Infatti risulta possibile, una volta stabiliti i parametri della funzione di approssimazione, richiedere alla funzione stessa una valutazione di una coppia stato-azione che è possibile non si sia mai verificata nell'arco dell'apprendimento, permettendo di gestire uno spazio di casistiche estremamente più ampio.

All'interno di un algoritmo di deep reinforcement learning questo elemento ha una grande importanza e ve ne possono essere di vario tipo. La struttura utilizzata nel paper dei giochi Atari da DeepMind [19] prevede un unico vettore di input composto dalle $n$ features dello stato attuale. I valori in uscita corrispondono invece ai valori di ogni singola azione associata allo stato in input $Q(s,a)$. La parte centrale corrisponde a una Convolutional neural network collegata, nell'ultimo hidden layer, a uno strato di funzioni binarie completamente connesse. Graficamente, dato uno stato rappresentato da un vettore di $n$ features $(x_1, x_2, ..., x_n)$ e un output per ognuna delle $m$ azioni:

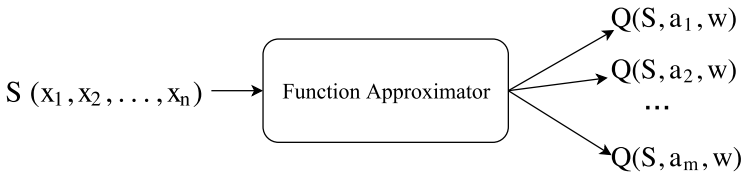



Figura 3.2: Funzione di approssimazione

Può essere utilizzato un qualsiasi tipo di funzione di approssimazione comune anche a vari algoritmi di Supervised Learning (e.g. Decision trees, Nearest neighbour etc.), ma le funzioni utilizzate più comunemente negli ultimi anni sono le Reti Neurali. Esse hanno la caratteristica di essere funzioni differenziabili. La necessità di funzioni differenziabili permette di adattare i parametri della funzione di approssimazione utilizzando una metodologia molto comune all'interno del Deep Learning, la Backpropagation. Per quanto riguarda i problemi di RL i dati hanno una stretta correlazione fra loro, di conseguenza non si può fare un'assunzione che spesso viene fatta nella risoluzione di problemi di supervised e unsupervised learning, cioè che essi siano stazionari e iid, i.e. indipendenti e identicamente distribuiti. Con alcune tecniche però, come si vedrà nella Sezione 3.2.4, è possibile adattare il learning delle neural networks in ambito Q-Learning a questa tipologia di dato.

### 3.2.3 Exploration - exploitation

Ogni volta in cui l'agente deve scegliere quale azione compiere ha fondamentalmente due modalità con cui può portare avanti la propria strategia. La prima modalità si dice *Exploitation* e consiste nel prendere la migliore decisione possibile date le informazioni ottenute fino a ora utilizzando le esperienze passate e memorizzate. Questa informazione è sempre disponibile sotto forma di value function, la quale esprime in ogni momento quale delle azioni offre il maggiore ritorno cumulativo finale per ogni coppia stato-azione. La seconda modalità viene detta *Exploration* e definisce la strategia per prendere decisioni differenti da quella che attualmente si ritiene ottimale. Si rivela di grande importanza la fase di esplorazione, poiché essa serve a raccogliere informazioni su stati ancora inesplorati o su cui si ha la necessità di approssimare ulteriormente la value function. Infatti è possibile che un agente, compiendo solamente l'azione ottimale, si limiti a percorrere sempre la stessa sequenza di azioni, senza aver mai la possibilità di esplorare e intuire che potenzialmente potrebbero esserci strategie migliori e che, a lungo termine, potrebbero portare a risultati nettamente superiori, anche se questo potrebbe portare a un eventuale discredito del guadagno immediato.

La policy maggiormente utilizzata per raggiungere il giusto compromesso tra exploration ed exploitation è la *$\epsilon$-greedy policy*. Essa rappresenta una metodologia di selezione delle azioni basata sulla possibilità di scelta di una azione casuale con distribuzione di probabilità uniforme. In particolare, nel momento in cui si sta svolgendo il training dell'agente, esso ha la possibilità di scegliere una azione random con probabilità $\epsilon$, con $0 \leq \epsilon \leq 1$, oppure di scegliere l'azione che si prevede avrà il massimo guadagno cumulativo finale $G_t$ (come formulato nella 3.5) con probabilità $1 - \epsilon$.

Comunemente durante il training il valore di $\epsilon$ viene inizializzato a un valore vicino a 1, per poi essere ridotto man mano nel tempo, in modo che inizialmente l'agente possa esplorare tutto lo spazio delle coppie stato-azione in modo totalmente casuale, raccogliendo la massima informazione possibile, per poi concentrare maggiormente la propria attenzione sulle strategie che possono essere potenzialmente migliori. Si può notare l'utilizzo di questa policy all'interno dello pseudo-codice dell'algoritmo di DQN (3.2.4) nella riga 7.

### 3.2.4 Experience replay e Target model

Come anticipato nella Sezione 3.2.2 la funzione di approssimazione può soffrire molto a causa della presenza di dati non indipendenti e identicamente distribuiti e non stazionari. Si può sopperire a questo genere di problema grazie a due metodi abbastanza semplici ma al contempo estremamente efficaci.

Il primo metodo viene chiamato *Experience replay*. Durante l'interazione tra l'agente e l'environment tutte le esperienze $< s, a, r, s' >$ (stato, azione, reward e stato successivo) vengono salvate in una *replay memory*, memoria di grandezza fissa e di tipo FIFO (i.e. First In First Out). Durante l'allenamento della rete, invece di utilizzare le recenti esperienze una di seguito all'altra, vengono utilizzati dei mini-batch di esperienze prese in modo casuale all'interno della replay memory. Questo permette di mitigare il problema della sequenzialità dei dati di training che potrebbe portare l'algoritmo a rimanere bloccato in un minimo locale, senza aver la possibilità di raggiungere la soluzione ottima. Si può notare l'utilizzo dell'Experience replay nello pseudo-codice dell'algoritmo di DQN alle righe 1, 10, 11.

Il secondo metodo viene detto *Target model*. Durante la computazione del gradiente e quindi durante l'aggiornamento dei pesi della rete effettuato con la tecnica di backpropagation, si utilizzano contemporaneamente due funzioni di approssimazione, i.e. due reti strutturalmente identiche ma con pesi differenti. L'errore della nuova rete viene calcolato come differenza tra il valore della rete target (rappresentato da $r + \gamma \underset{a'}{max} Q(s', a'; w_i^-)$ nell'equazione 3.16) e il valore stimato dalla rete attuale. Dopo un certo numero di passi, stabilito in base alle esigenze, la Target Q-Network sarà aggiornata con i pesi della nuova rete $w_i$.

Il problema dell'aggiornamento online dei pesi della rete target e della rete attuale deriva sempre dalla grande correlazione tra i Q-value target e quelli stimati in uno stesso

momento. Questo crea una grande instabilità e potrebbe guidare la rete in una spirale di aggiornamento, rendendo impossibile il raggiungimento del minimo globale.

Si può fare riferimento alle righe 2, 3, 13 dello pseudo-codice nella 3.2.4, dove si nota che vengono inizializzate due reti con gli stessi valori di pesi e la stessa identica architettura. Durante l'apprendimento però solamente una delle due reti viene aggiornata a ogni step, cioè quella che rappresenta la action-value function attuale. La rete target viene aggiornata solamente ogni $C$ step, settando i valori della target action-value function a quelli della action-value function.

$$L_i(w_i) = \mathbb{E}_{s,a,r,s'}[(r + \gamma \underset{a'}{max} Q(s', a'; w_i^-) - Q(s, a; w_i))^2] \tag{3.16}$$

Deep Q-Learning Algorithm

**Algorithm 1** DQN Algorithm adapted from [10]

1: Replay memory $D$ initialization
2: Action-value function $Q$ initialization with random weights $\theta$
3: Target action-value function $\hat{Q}$ initialization with weights $\theta^- = \theta$
4: **for** episode=1 to M **do**
5: Initialize sequences $s_1 = \{x_1\}$ and preprocessed sequence $\phi = \phi(s_1)$
6: **for** t=1 to T **do**
7: $\epsilon$-greedy policy, select $a_t = \begin{cases} a \text{ random action} & \text{with prob. } \epsilon \\ argmax_a Q(\phi(s_t), a; \theta) & \text{otherwise} \end{cases}$
8: Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
9: Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
10: Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $D$
11: Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $D$
12: Set $y_j = \begin{cases} r_j & \text{if } \phi_{j+1} \text{ is terminal} \\ r_j + \gamma max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$
13: Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ w.r.t. the network parameter $\theta$
14: Every $C$ steps reset $\hat{Q} = Q$, i.e. set $\theta^- = \theta$
15: **end for**
16: **end for**

### 3.2.5 Double DQN

Definito l'algoritmo standard di DQN, già incredibilmente soddisfacente nel dominio dei giochi Atari, si sono susseguite diverse soluzioni per risolvere problemi minori ma comunque rilevanti per l'apprendimento dell'agente.

Uno di questi problemi consiste nella possibile sovrastima della funzione di valore Q, dovuta all'operatore di massimo all'interno della formula $Q(s,a) \rightarrow r + \gamma max_a Q(s',a)$. Questo accade poiché la stima $E\{max_a Q(s',a)\}$ non è reale, ma è *biased*, i.e. si discosta dal valore reale. Essa risulta in una approssimazione rumorosa della soluzione ottimale $Q_*$. Considerando, per esempio, un set di azioni con lo stesso valore reale di action-value function in un certo stato. La stima che viene fatta dal nostro algoritmo, essendo rumorosa, conterrà delle variazioni in positivo o in negativo del valore reale. Ciò che viene implicitamente fatto dall'operatore di massimo è la scelta dell'azione che ha l'errore di stima più alto in positivo, ripercuotendo la scelta di questa azione anche agli stati e alle selezioni successive. Questo porta a una accumulazione dell'errore che crea seri problemi alla stabilità dell'algoritmo.

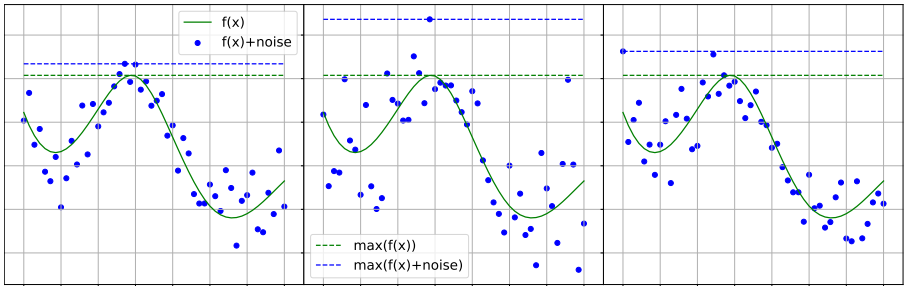



Figura 3.3: Rappresentazione dell'incidenza del rumore nella considerazione del massimo di una funzione a cui viene applicato del rumore bianco.

Come si può notare nella Figura 3.3 abbiamo la rappresentazione di 3 Optimal Q-Value function identiche in verde e le stime rumorose effettuate dall'algoritmo delle tre funzioni. Si osserva che l'azione scelta con l'operatore di massimo è sempre quella con errore positivo maggiore.

Una soluzione possibile è stata proposta in un paper del 2010 [1] da Hado Van Hesselt e viene chiamata Double Q-Learning. Essa consiste in una variazione dell'apprendimento che dissocia, durante l'aggiornamento della funzione di valore, la scelta dell'azione dalla stima del valore della funzione stessa. In particolare se consideriamo due funzioni $Q_1$ e $Q_2$, esse vengono aggiornate separatamente e indipendentemente, usando una delle due per determinare l'azione massimizzante e l'altra per stimare il valore della coppia stato-azione. La funzione $Q_1$ verrà utilizzata per ottenere quella che si ritiene sia l'azione migliore

al momento, mentre la funzione $Q_2$ si utilizza per valutare il valore dell'azione selezionata da $Q_1$.

Il cambiamento effettuato all'interno dell'algoritmo di DQN non è così sostanziale come si potrebbe pensare. Infatti, avendo a disposizione già due reti, la Target Q-Network e la Q-Network, è possibile solamente cambiare la scelta alla riga 12 di 3.2.4 come segue:

$$y_j = \begin{cases} r_j & \text{if } \phi_{j+1} \text{ is terminal} \\ r_j + \gamma \hat{Q}(\phi_{j+1}, argmax_{a_j} Q(\phi_{j+1}, a_j; \theta); \theta^-) & \text{otherwise} \end{cases} \tag{3.17}$$

Quello che cambia dunque è l'utilizzo della Q-network primaria per scegliere l'azione e la Target Q-Network per generare il target Q-value per quell'azione. In questo modo si riduce la sovrastima stabilizzando l'algoritmo.

### 3.2.6 Dueling DQN

Con la nuova architettura delle dueling neural networks si cerca di risolvere un problema differente.

Considerando la action-value function $Q(s, a)$, essa può essere decomposta in due tipologie di valori differenti. La prima consiste nella state-value function $v(s)$ che rappresenta semplicemente il valore dello stato $s$ come definita nella Sezione 3.1.2. La seconda invece viene detta *advantage-function* e si denota con $A(a)$. Essa rappresenta il valore di una certa azione rispetto alle altre in un determinato stato. Si può dunque riscrivere l'action-value function in funzione dei due valori appena considerati nel seguente modo:

$$Q(s, a) = v(s) + A(a) \tag{3.18}$$

Quello che viene effettuato tramite la riformulazione della struttura della Deep Neural Network è il disaccoppiamento fra l'advantage function e la value function, ricombinandole assieme solamente nell'ultimo layer della rete. In questo modo l'agente ha la possibilità di valutare separatamente il valore del singolo stato e il valore di ogni azione. Questo è utile nei casi in cui l'agente non abbia bisogno di valutare tutte e due le tipologie di funzione contemporaneamente e in ogni stato, come nei casi in cui non ci siano azioni che debbano per forza predominare su altre e quindi risulta vantaggioso modellare maggiormente solamente la funzione di valore.

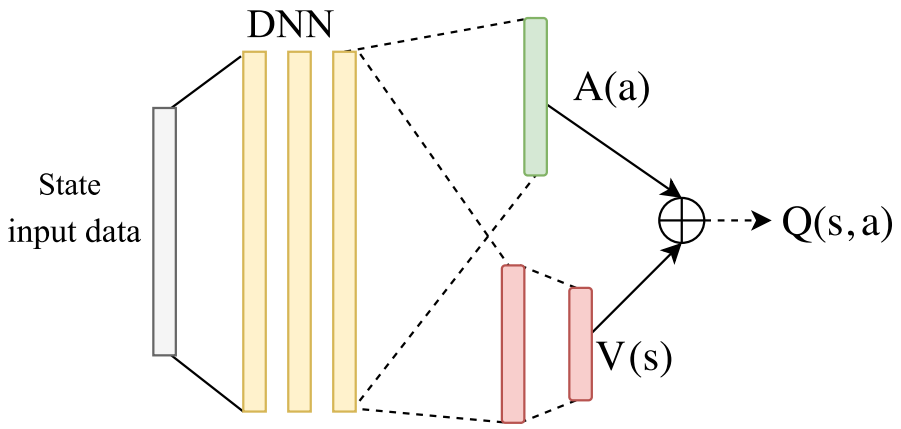



Figura 3.4: Dueling DQN Architecture

## 3.3 Librerie e frameworks

In questa sezione si presentano brevemente le librerie e i *frameworks* utilizzati durante lo sviluppo della tesi, concentrandosi sulle caratteristiche e le motivazioni che hanno portato alla loro adozione finale. Si analizzerà inoltre l'interazione tra di esse all'interno del progetto e i collegamenti con le componenti del Deep Reinforcement Learning.

### 3.3.1 Theano

Theano [17] è una libreria scritta in linguaggio di programmazione Python completamente Open Source, nata con l'obiettivo di definire, ottimizzare e valutare espressioni matematiche. Sviluppata da un gruppo specializzato in Machine Learning, LISA/MILA Lab, all'interno dell'Università di Montreal, essa si concentra principalmente sulla manipolazione di espressioni di tipo matriciale. Theano trova il suo punto di forza nell'implementazione di questa tipologia di operazioni a livello di GPU (Graphical processing unit), riuscendo in molti casi a rivaleggiare con codice scritto in linguaggio C ottimizzato manualmente. Inoltre una caratteristica fondamentale per qualsiasi algoritmo di Deep Learning è l'ottimizzazione delle espressioni matematiche computate ripetutamente in cui il tempo di esecuzione diventa estremamente critico. Queste caratteristiche rendono la libreria estremamente interessante per algoritmi di Deep learning e viene utilizzata come *back-end*[1] in *frameworks* di livello più alto come, ad esempio, Keras, altra libreria ampiamente utilizzata all'interno di questo progetto.

[1]Back-end: libreria o framework che compie un servizio per un'altra libreria non mostrandosi direttamente, ma essenziale al suo funzionamento finale.

### 3.3.2 Keras

Keras [3] è una libreria Open Source di alto livello per la progettazione e l'implementazione di reti neurali scritta in linguaggio di programmazione Python. L'obiettivo principale della libreria viene espresso dalla frase riportata nella documentazione della stessa: *"Being able to go from idea to result with the least possible delay is key to doing good research."*. Questa frase esprime chiaramente la necessità, in ambito di ricerca, di una prototipazione veloce e la possibilità di avere risultati nel minor tempo possibile. Per raggiungere tale obiettivo essa è stata scritta in modo da poter essere modulare, intuitiva e *user-friendly*, mettendo a disposizione inoltre un'ampia documentazione. Per le operazioni di più basso livello, al momento della stesura di questa tesi, offre la possibilità di utilizzare due librerie: Theano e Tensorflow, ma è già in corso l'estensione ad altri *back-end* come, ad esempio, CNTK di Microsoft. L'integrazione con *back-end*[1] che offrono supporto alla computazione su multi-GPU rendono anche Keras intrinsecamente interessante dal punto di vista computazionale.

### 3.3.3 OpenAI Gym

OpenAI Gym [2] è un toolkit per la ricerca legata agli algoritmi di reinforcement learning. Lo scopo è quello di offrire una piattaforma agli sviluppatori per condividere i propri algoritmi e le proprie soluzioni con altri utenti e contemporaneamente poter visionare i risultati e gli algoritmi di tutta la *community*. Essa mette una serie di environment a disposizione di chiunque voglia utilizzarli, dando poi la possibilità di implementare e caricare la propria soluzione in modo da creare una classifica con gli algoritmi aventi risultati migliori. La caratteristica che rende OpenAI Gym molto appetibile per questo progetto è che viene messa a disposizione una interfaccia standard per la creazione degli environment, quindi chiunque può costruirne uno avendo una solida base d'appoggio. Nel caso di questa tesi lo sviluppo è iniziato proprio dal Forex Trading environment, apportando in corso d'opera sempre più modifiche e arricchendolo di funzionalità.

### 3.3.4 Keras-rl

Keras-rl [12] è un'altra libreria Open Surce che implementa alcuni algoritmi di reinforcement learning noti e analizzati nei paper più famosi. Creata da Matthias Plappert e scritta in linguaggio Python, essa si integra nel migliore dei modi con Keras, offrendo la possibilità quindi di utilizzare i *back-end* citati in precedenza, Tensorflow e Theano, e di sfruttare la GPU o la CPU per il training degli algoritmi. L'integrazione con Keras permette inoltre la creazione semplice e modulare dei modelli di rete neurale la valutazione di essi utilizzando tutte le metriche presenti in Keras. Altra integrazione molto importante della libreria è quella con l'ambiente di sviluppo di OpenAI Gym, libreria scelta per l'interfaccia di sviluppo offerta dagli *Environment* definiti in essa. La libreria è in continuo sviluppo e

nell'arco della stesura di questa tesi gli aggiornamenti non sono mancati, passando dalla versione 0.2 alla 0.3 e implementando costantemente nuove funzionalità e nuovi algoritmi.

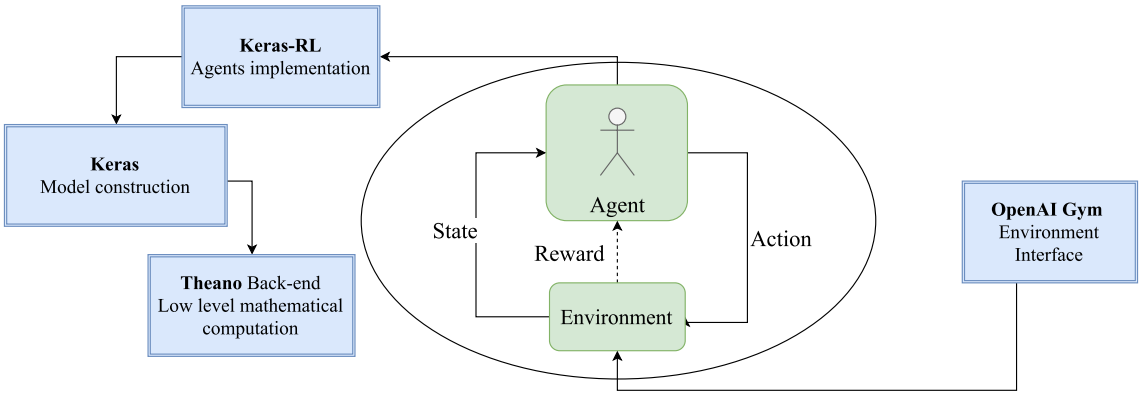



Figura 3.5: Rappresentazione grafica sull'utilizzo delle librerie

# Capitolo 4

# Forex e Deep Q-Learning

Con la metodologia seguente si vuole allenare un agente di Reinforcement Learning ad apprendere una strategia di guadagno nel mercato del Forex su un arco temporale di durata fissa. Per fare ciò l'agente riceve in input lo stato del mercato, rappresentato da tassi di cambio e volumi, successivamente ingegnerizzati per creare nuove features, dette indicatori, offrendo all'agente la possibilità di scegliere in che modo esporsi sul mercato con l'obiettivo di guadagnare il più possibile entro il periodo prestabilito. Ricevendo un reward associato alle mosse effettuate, l'agente capirà quale strategia gli permette di avere il maggior profitto nel lasso di tempo indicato, al termine del quale esso si vedrà chiudere tutte le posizioni aperte, ricevendo il reward finale.

In questo capitolo si esaminano a fondo le tecniche utilizzate durante lo sviluppo del progetto, il setup dell'algoritmo e i dati a disposizione. Si affrontano inoltre le problematiche riscontrate in via di sviluppo e le possibili tecniche di risoluzione di queste ultime.

## 4.1 Definizione del problema

I classici problemi di reinforcement learning prevedono strutture molto differenti da quella utilizzata nell'ambito di questa tesi. Infatti, principalmente per motivi aziendali, si è rivelata la necessità di strutturare il problema in modo congruo al supervised learning. Mentre nei problemi di Deep Q-Learning classici, il training set e il test set coincidono, nel problema del FX market l'agente deve imparare le mosse da compiere su un dataset di test totalmente ignoto e non utilizzato durante la fase di apprendimento.

Per quanto riguarda l'iniziale strutturazione del problema e la regolazione degli iperparametri riguardanti le varie componenti, si è preso spunto dalle impostazioni utilizzate nell'ambito dei giochi Atari ampiamente trattate nel paper «Human-level control through deep reinforcement learning». [19]. L'agente ha la possibilità di acquisire esperienza in modo da massimizzare il reward, fino al momento in cui non termina il gioco a causa di

una sconfitta oppure per il passaggio al livello successivo. A ogni passo al momento $t$ il videogiocatore virtuale (l'agente) riceve i pixel costituenti l'attuale schermata di gioco, che descrivono in maniera univoca lo stato dell'ambiente, i quali vengono pre-processati tramite tecniche di standardizzazione e/o normalizzazione, e dati in input a una DQN, i.e Deep Q-Network, di tipo convolutivo. Avendo a disposizione questi dati l'agente è in grado di stabilire qual'è la migliore azione da compiere a ogni frame (o a ogni gruppo di frame) in base allo stato descritto dai pixel. Si noti che in questo caso i dati ricevuti dall'agente rappresentano la schermata di gioco come una 'vista' che il videogiocatore ha sullo stato della partita, come se fosse un giocatore umano che attraverso un processo di allenamento impara e determina qual'è la strategia migliore da seguire al fine di ottenere il massimo punteggio. L'azione selezionata viene poi processata dall'environment, in questo caso il gioco, che provvederà a restituire un reward dipendente da diversi fattori. Per quanto riguarda il reward descritto nei giochi Atari è risultato necessario descriverlo tramite l'utilizzo di tre semplici valori: +1 ogni qual volta il punteggio aumenta, -1 quando diminuisce e 0 nel caso in cui non vi siano cambiamenti. Congiuntamente al reward associato all'azione e allo stato precedente, viene elaborato e restituito anche lo stato successivo.

A seconda della tipologia di gioco, le partite possono avere o durata infinita, i.e. fino alla sconfitta del giocatore, oppure durata fissa, i.e. determinata da un certo numero di timestep. Il training di durata fissa viene chiamato episodio, al termine del quale viene resettato sia lo stato dell'agente, sia lo stato dell'environment, mentre la partita inizia da capo, con punteggio azzerato. In questo caso l'agente ha l'obiettivo di ottenere il miglior punteggio su un certo episodio, che corrisponde contemporaneamente sia al training che al test.

Si è deciso di strutturare il problema legato al Forex Trading in modo simile, ma rivedendo alcuni concetti fondamentali. L'obiettivo di un agente che opera nel mercato del Forex rimane lo stesso: la massimizzazione del punteggio durante una partita. In questo caso però il punteggio corrisponde al guadagno monetario in percentuale rispetto a una somma investita di valore unitario, mentre con partita si intende un arco di tempo di durata fissa. In questo caso, quindi, non esistono episodi di durata infinita. L'agente in questo arco di tempo ha la possibilità di costruire una strategia basandosi su un processo di *trial and error*. La durata dell'episodio è stata fissata a 120 ore, corrispondenti a una settimana di 5 giorni. Il motivo di questa scelta è dettato dall'apertura e chiusura del mercato a livello mondiale. Il mercato del Forex apre il lunedì alle 17 EST[1] e chiude alle 17 del venerdì della stessa timezone. Al posto dei pixel del gioco l'agente vede esattamente quello che hanno a disposizione i trader del forex, una serie di tassi di cambio, volumi e indicatori. L'agente ha inoltre la possibilità di ricevere come stato di input la propria esposizione attuale sul mercato, il *PMC*, i.e. prezzo medio di carico, e l'ora del giorno codificata in seno e coseno, i quali ne offrono una descrizione compatta e univoca. Avendo a disposizione questi dati, in particolare PMC, tasso di cambio EUR/USD ed esposizione, si offre alla rete la possibilità di associare il possibile reward istantaneo alle azioni da

scegliere, adattando al meglio la propria strategia.

Ogni ora l'agente può decidere di eseguire l'azione di buy, sell o flat, come descritte nella Sezione 2.1.2, esponendosi con un ammontare di denaro unitario. Si limita inoltre la massima esposizione sul mercato dell'agente. Nel caso in cui al termine dell'episodio l'agente abbia ancora delle posizioni aperte sul mercato, la sua esposizione verrà riportata a 0 e otterrà un reward proporzionale al numero di posizioni chiuse.

## 4.2 Dati e indicatori

Un grande vantaggio portato dall'algorithmic trading è la possibilità di elaborare in *real-time* una grande quantità di dati per poter estrarre da essi quanta più informazione possibile, in modo da effettuare la scelta di investimento opportuna. Risulta estremamente importante presentare all'agente i dati che meglio rispecchiano l'andamento del mercato in un determinato momento per poterne prevedere l'evoluzione futura. Nella presente sezione si mira ad analizzare dati e indicatori utilizzati durante le varie fasi del progetto fino alla configurazione finale.

### 4.2.1 Serie temporali di volumi e tassi di cambio

Gli stati presentati all'agente sono formati da serie temporali rappresentanti, per ogni istante di tempo, il valore di una feature nel mercato del Forex. In particolare i dati utilizzati sia come feature non ingegnerizzate, sia per creare gli indicatori (trattati nella Sezione 4.2.2), sono campionanti minuto per minuto in un periodo di tempo che va dal 03-12-2007 al 31-12-2016. Per il training del modello viene effettuato un sotto campionamento di un dato ogni 60 minuti, permettendo all'agente di visualizzare stati e compiere azioni a distanza di un'ora l'una dall'altra. In particolare gli episodi presentati all'agente sono di lunghezza fissa (120 ore) con timestep iniziale corrispondente a un qualsiasi momento all'interno dei dati appartenenti al training set. Potenzialmente l'agente si potrebbe trovare nella condizione di investire a cavallo di due settimane differenti.

Nella seguente tabella si rappresentano le feature e i corrispondenti periodi di tempo a disposizione utilizzabili come dati di training, test e validation.

[1]EST: Eastern Standard Time

Tabella 4.1: Tassi di cambio e volumi a disposizione

| Tasso di cambio e Volume | Data d'inizio | Data di fine | Numero di timestep |
|---|---|---|---|
| Euro - Dollaro statunitense | 03-12-2007 | 31-12-2016 | 3347521 |
| Euro - Franco Svizzero | 03-12-2007 | 31-12-2016 | 3347521 |
| Oncia d'oro - Euro | 03-12-2007 | 31-12-2016 | 3347521 |
| Indice Tokyo (JP255) | 03-12-2007 | 31-12-2016 | 3347521 |
| Indice petrolio (WTICO) | 03-12-2007 | 31-12-2016 | 3347521 |
| Indice azionario americano (SPX500) | 03-12-2007 | 31-12-2016 | 3347521 |
| Indice azionario Hong Kong (HK33) | 03-12-2007 | 31-12-2016 | 3347521 |

### 4.2.2 Indicatori tecnici

Nelle prime fasi di sviluppo si sono utilizzati i dati descritti nella Sezione 2.1.7 senza apportare particolari modifiche. Lo stato del mercato osservato dall'agente consisteva dunque nell'insieme delle features non ingegnerizzate, campionate nel tempo ogni 60 minuti. Queste ultime si sono rivelate essere insufficienti ai fini dell'apprendimento dell'agente, di conseguenza si è deciso di spostarsi verso la direzione di ingegnerizzazione. Questa scelta è stata adottata anche grazie a una consapevolezza a priori del beneficio potenzialmente introdotto dagli indicatori, utilizzati ampiamente sia in contesto aziendale, sia da investitori umani, che spesso basano le loro strategie solamente su di essi.

Il motivo che porta all'utilizzo di questa tipologia di feature è semplice: le serie temporali rappresentate dagli andamenti degli indici di mercato e i tassi di cambio risultano estremamente rumorose, tanto da poter dare l'impressione di avere una variazione nel tempo riconducibile alla teoria del random walk, rendendo estremamente complesso rivelare il trend di mercato sottostante. La rimozione del rumore però non è affatto semplice e vi sono una grande quantità di indicatori potenzialmente utili a estrarre alto valore informativo per la predizione dell'andamento di mercato.

La fase di costruzione degli indicatori dunque consiste in una particolare trasformazione dei dati di input che punta a compiere principalmente due azioni: rimuovere manualmente informazioni inutili riconducibili a rumore e offrire informazioni a un livello di dettaglio utile alla rete a imparare nel modo più semplice possibile. Tali azioni potrebbero anche ripercuotersi dal punto di vista opposto. Spesso accade infatti che, cercando di eliminare il rumore, si riducano le informazioni che realmente servono alla rete.

Un'altra importantissima caratteristica degli indicatori è che essi raccolgono informazioni sul passato mediandole con quelle presenti. Questo permette di formare un insieme

di features che rappresenta non solo lo stato attuale, ma anche un insieme di timestep passati utilizzati per costruire l'indicatore stesso. Si può dunque assumere che gli indicatori abbiano una intrinseca componente di memoria degli stati passati. Ricordando che nel reinforcement learning è comune fare l'assunzione che l'ambiente sia completamente osservabile e che lo stato precedente caratterizzi completamente lo stato futuro (si veda la Sezione 3.1.2 riguardante i Markov Decision Process), l'inclusione degli indicatori nello stato permette di raccogliere una maggiore quantità informazione riguardante gli stati passati.

Si presentano di seguito tutti gli indicatori utilizzati nella fase finale del progetto, molti dei quali sono stati estratti attraverso un adattamento della libreria TaLib, utilizzata per la generazione degli indicatori a partire dai tassi di cambio e volumi a disposizione [1].

Molti indicatori sono caratterizzati da un periodo di campionamento nel tempo opportuno. Quelli elencati di seguito sono stati generati per periodi che variano tra i 5 minuti e le 3 ore, in modo da estrarre il maggior contenuto informativo possibile sia per brevi che per medi periodi. Mediare per periodi di tempo più ampi potrebbe abbassare troppo l'importanza delle informazioni all'istante attuale.

### Medie Mobili - Moving Averages

Grazie alle medie mobili è possibile rimuovere una grande quantità di rumore dagli andamenti analizzati. Esse infatti tendono a smussare le fluttuazioni di mercato, rimuovendo, a seconda del periodo utilizzato, dei "picchi" che non rispecchiano il vero e proprio andamento del mercato. Spesso infatti è possibile che i picchi positivi o negativi rappresentino un discostamento dal trend significativo del mercato, indicando una evoluzione non rappresentativa della realtà sul più lungo periodo.

Le medie mobili possono essere utilizzate in due modi:

- prendendole singolarmente rispecchiano l'andamento del mercato. Scegliendo periodi di tempo appropriati è possibile rappresentare il trend di mercato rimuovendo l'effetto del rumore;
- con medie mobili multiple basate su più periodi temporali si possono invece valutare i "crossover". Essi corrispondono ai punti in cui le medie di lungo e breve periodo si incontrano. Spesso si trovano in corrispondenza di un forte andamento crescente o decrescente del mercato.

[1] Documentazione della libreria TaLib: https://pypi.python.org/pypi/TA-Lib

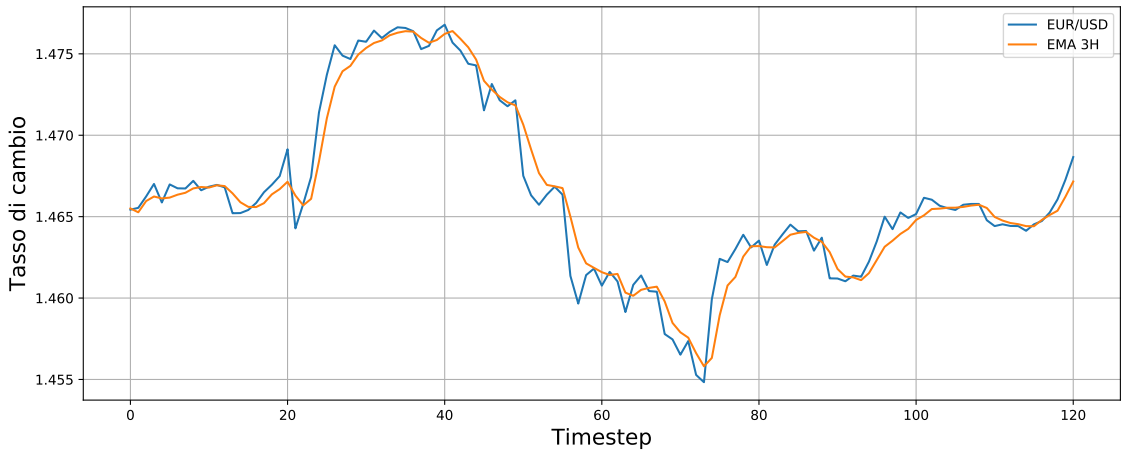



Figura 4.1: Grafico Exponential moving average

Si elencano di seguito le tipologie di moving average utilizzate:

- *Simple moving average*, SMA: calcolata come media semplice di una serie di tassi di cambio considerando $n$ periodi passati. Essa viene calcolata come segue:

$$SMA_t = \frac{\sum_{i=0}^{n-1} p_{t-i}}{n} \tag{4.1}$$

  con $p_t$ indicante il tasso di cambio al tempo t.

- *Weighted moving average*, WMA: si propone come una variante della SMA assegnando un peso a ogni singolo valore preso in considerazione. In particolare i pesi decadono in modo lineare andando indietro nel tempo, in modo che i valori più recenti abbiano un maggiore impatto sul valore finale della media.

- *Exponential moving average*, EMA 4.1: allo stesso modo della WMA, l'EMA include un set di pesi nel calcolo nella media. In questo caso però i pesi decadono in modo esponenziale invece che lineare, aumentando ulteriormente l'importanza del valore del tasso di cambio nei periodi di tempo più recenti.

- *Moving Average convergence-divergence*, MACD: è uno degli oscillatori[1] più utilizzati nell'analisi di mercati finanziari. Rappresenta una combinazione di più medie esponenziali da cui è possibile ricavare cambiamenti di trend di un determinato valore di mercato.

[1]Oscillatore: strumento utilizzato nella analisi tecnica finanziaria per la valutazione dei movimenti di mercato.

### Momentum

Il momento è uno degli indicatori più semplici, ma allo stesso tempo viene ampiamente utilizzato sotto diverse forme. Esso misura la velocità del cambiamento dei prezzi. Si definisce come la differenza fra il prezzo attuale $p_t$ e il prezzo a $n$ timestep nel passato $p_{t-n}$. Calcolato come $M_t = p_t - p_{t-n}$ esso può assumere valori positivi nel caso in cui il prezzo istantaneo sia maggiore di quello a $n$ periodi nel passato, negativi altrimenti. Valori estremamente alti o estremamente bassi del momentum indicano che una valuta, in quel momento, è in *oversell*, i.e. molto venduta, o in *overbuy*, i.e. molto comprata. Spesso la situazione indicata in precedenza sottolinea una maggiore probabilità di cambio di trend.

### Rate Of Change

Il *Rate of change* è un indicatore tecnico del momento che misura la percentuale di cambio di prezzo fra il prezzo al timestep $t$ (prezzo attuale) e il prezzo a $n$ periodi nel passato, i.e. prezzo al timestep $t - n$.

Esso viene calcolato come:

$$ROC_t = 100 * \frac{(price_t - price_{t-n})}{price_{t-n}} \tag{4.2}$$

Intuitivamente esso rappresenta la forza del momento del tasso di cambio, poiché indica la pendenza nel periodo di tempo preso in considerazione. Valori positivi di questo oscillatore indicano una tendenza positiva del mercato, invitando il trader ad aumentare la propria posizione investendo in buy. Valori negativi, al contrario, indicano una discesa del tasso di cambio e indicano al trader di andare in sell.

### Bande di Bollinger

Le bande di Bollinger sono uno strumento utilizzatissimo all'interno del mercato finanziario. Esso è un indicatore costituito da 3 medie mobili. Una media mobile principale calcolata come SMA a un certo periodo di tempo $\Delta t$, mentre le altre due medie sono calcolate a partire dalla principale tenendo in considerazione la deviazione standard del mercato in quel momento. Quest'ultima, moltiplicata da un certo valore K, viene aggiunta o sottratta alla media principale per creare le altre due medie.

In particolare, indicando con $\sigma$ la deviazione standard del mercato, le 3 bande sono definite come segue:

- Banda principale: Media mobile semplice di $n$ periodi nel passato.
- Banda superiore: Banda principale + $K\sigma$
- Banda inferiore: Banda principale - $K\sigma$

Sulle bande di Bollinger si possono effettuare alcune osservazioni importanti:

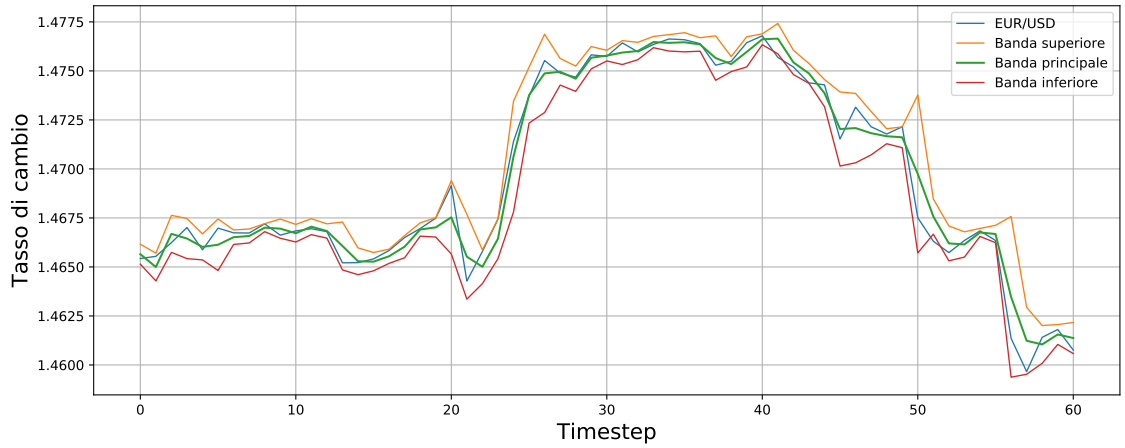



Figura 4.2: Rappresentazione grafica delle Bande di Bollinger calcolate sul tasso di cambio EUR/USD con K settato a 3 timestep.

- Un abbassamento della deviazione standard, i.e. della volatilità dei tassi di cambio, si rispecchia sul restringimento delle bande di Bollinger. Teoricamente nel momento di bassa distanza fra le bande i tassi prendono una determinata direzione. Per determinare quest'ultima in modo più preciso è però necessario aiutarsi con altri indicatori.
- Quando il prezzo raggiunge la banda superiore o inferiore, teoricamente esso dovrebbe seguire l'andamento determinato dalla banda raggiunta per un buon periodo di tempo.

### Volume Weighted Average Price - VWAP

L'unico indicatore utilizzato all'interno di questa tesi che combina fra loro i volumi di trading con i tassi i cambio e gli indici. Esso è calcolato tramite il rapporto tra il valore investito in un certo periodo di tempo e il volume totale di investimenti nello stesso periodo. Il VWAP rappresenta quindi la misura del prezzo medio a cui un tasso di cambio viene negoziato nell'arco di un periodo temporale $\Delta t$.

Indicando con $p_t$ il tasso di cambio al timestep $t$ e con $v_t$ il volume scambiato nel tempo $\Delta t$ intercorrente tra il timestep $t$ e $t-1$:

$$VWAP_t = \frac{\sum_t (p_t * v_t)}{\sum_t v_t} \tag{4.3}$$

### 4.2.3 Standardizzazione e normalizzazione

Nel mondo del machine learning assume un ruolo fondamentale la trasformazione dei dati utilizzati come input feature. Accade spesso infatti che essi siano totalmente differenti dal punto di vista della distribuzione o dell'ordine di grandezza, causando forti squilibri nella fase di apprendimento. Questo problema si ripercuote ampiamente nell'ambito del Forex Trading, poiché il rapporto di scala fra i tassi di cambio e i volumi considerati è completamente differente. Risulta necessaria e cruciale dunque una fase di preprocessing, per fare in modo che essi possano essere interpretati in modo opportuno dall'agente e che la conoscenza ricavata sia maggiormente rilevante. In particolare nelle neural network, scalare i dati è importante per fare in modo di minimizzare il bias tra le feature e, di conseguenza, ridurre drasticamente il tempo di training dell'algoritmo.

I classici metodi di standardizzazione prevedono la necessità di poter stimare i valori massimi e minimi del dataset, oppure di calcolare funzioni su tutto il dataset come deviazione standard, varianza o media. Questo risulta possibile nella situazione in cui le predizioni sono necessarie solamente offline e le scale di valori dei dati e la loro distribuzione sono ben definite. In questo modo è possibile calcolare tutte le metriche senza nessun tipo di problema, a vantaggio del training dell'algoritmo. Ciò non risulta affatto possibile nel momento in cui ci si trova in presenza di dati sconosciuti oppure i cui i valori minimi e massimi non sono stabiliti a priori, come nel caso delle serie temporali in ambito finanziario. Il motivo principale di questa incompatibilità sta nel fatto che le predizioni devono essere effettuate su dati mai visti e in real time, che dunque potrebbero eccedere i livelli di minimo e massimo dei dati conosciuti a priori. Per questo motivo è necessario utilizzare delle tecniche di normalizzazione che non implichino conoscenza a priori di dati futuri nella fase di training. Esistono varie metodologie per raggiungere il risultato desiderato. Si può definire, ad esempio, una conoscenza a priori del dominio di applicazione e stabilire massimi e minimi a seconda di quest'ultimo. In questo caso si utilizza quindi una conoscenza intrinseca non dovuta necessariamente ai dati in quel momento, ma a una consapevolezza più generale del campo di applicazione.

Le metodologie utilizzate per scalare i dati in modo consistente sono chiamate tecniche di normalizzazione e standardizzazione. La più famosa e utilizzata tecnica di normalizzazione è la *Min-Max Normalization.* In questo caso tutti i dati di input sono mappati in un range predefinito tra $[new_{\min}, new_{\max}]$, utilizzando il minimo e il massimo valore dell'attributo da normalizzare.

Per normalizzare un valore $a$ della feature $A$ a $a'$ nel range $[new_{\min}, new_{\max}]$ si applica la seguente formula:

$$a' = new_{\min} + \frac{(a - A_{\min}) * (new_{\max} - new_{\min})}{A_{\max} - A_{\min}} \tag{4.4}$$

dove $A_{\min}$ e $A_{\max}$ rappresentano i valori di minimo e massimo attuali dell'attributo A.

Sebbene sia stata utilizzata nelle prime fasi di test del progetto, si è deciso di abbandonarla poiché richiede la conoscenza di informazioni non disponibili a priori, cioè il massimo e il minimo delle serie temporali.

In un secondo momento ci si deciso di adottare una metodologia differente, considerando una tecnica di standardizzazione anch'essa molto adottata in diversi ambiti: la *Z-Score Normalization*. I valori della serie temporale vengono scalati in modo da portare la media a 0 e la deviazione standard a 1. Molto utile quindi nel caso in cui le feature in input differiscano molto per quanto riguarda la scala di valori. In questo caso vengono utilizzati la media e la deviazione standard della serie da normalizzare. In particolare la formula adottata per passare da un valore $a$ di $A$ a $a'$ è:

$$a' = \frac{a - \mu(A)}{\sigma(A)} \tag{4.5}$$

$$dove \quad \mu(A) = \frac{\sum_{i=1}^{N} a_i}{N} \quad e \quad \sigma(A) = \sqrt{\frac{\sum_{i=1}^{N}(a_i - \mu(A))^2}{N}} \tag{4.6}$$

Questo metodo risulta molto utile nel caso di dati stazionari di cui non si conoscono i valori massimi e minimi, ma di cui si può fare una stima della media e dello scarto quadratico medio. Il problema è che, anche in questo caso, si suppone una conoscenza a priori dei dati poiché nelle timeseries la media e la deviazione standard variano nel tempo.

La forte assunzione fatta durante lo sviluppo della tesi è la seguente:

il valore della media e della deviazione standard del mese precedente alla settimana di training attuale non si discostano molto da quelli riguardanti la settimana di trading stessa. Di conseguenza la standardizzazione effettuata prevede il calcolo della media e della deviazione standard senza prendere in considerazione tutta la serie temporale, ma per ogni episodio vengono prese le 480 ore precedenti la settimana di training attuale (corrispondenti a un mese), si calcolano la media e la deviazione standard su di esse e si applicano come valori di standardizzazione alla settimana di trading.

## 4.3 Training, validation e test

Anche se spesso all'interno del Deep Q-Learning il dataset di training corrisponde al dataset di test, dove l'obiettivo è massimizzare il reward finale all'interno di un gioco con dinamiche sempre uguali, ci si è discostati da questo approccio e si è optato di affrontare il problema con il classico metodo utilizzato in classici problemi di supervised learning. Avendo un dataset molto grande a disposizione esso si divide in 3 parti, training set, validation set e test set. Il dataset viene diviso in modo sequenziale, utilizzando la prima parte come dataset di training, la parte centrale come dataset di validation e la parte finale come

dataset di test. Durante la fase di apprendimento l'agente ha la possibilità di allenarsi sui dati di training, tentando di ricavare strategie con le migliori performance su di essi. Periodicamente viene effettuata una verifica dello stato di apprendimento e generalizzazione, controllando la strategia sul dataset di validation. Questo controllo è necessario poiché spesso accade che sui dati training l'agente impari ad agire nel modo migliore rischiando però che l'andamento di learning non si rispecchi allo stesso modo sui dati di validation e test. Questo è normale poiché i parametri della rete neurale vengono regolati in direzione degli errori che compie sui dati di training. Generalmente a questo fenomeno viene dato il nome di *overfitting*, il caso in cui l'agente memorizza le mosse migliori sul training set abbassando contemporaneamente la capacità di generalizzare le proprie conoscenze su dati che non ha mai visto.

L'obiettivo finale, nel caso del forex trading, non è imparare a guadagnare sul dataset di training, ma massimizzare il reward sul dataset di test, senza che i dati appartenenti a quest'ultimo vengano mai visti da parte dell'agente. Questa è una caratteristica estremamente importante e decisiva per quanto riguarda la significatività dei risultati, cosa strettamente necessaria in un mondo in cui si possono effettuare investimenti ad altissimo capitale.

### Periodi temporali

La suddivisione del dataset in 3 parti ci pone davanti alla scelta della grandezza di ognuna di esse. Anche questo è un parametro estremamente importante poiché è necessario trovare un trade-off tra quanti dati ha a disposizione l'agente per la parte dedicata all'allenamento e su quanti ha la possibilità di verificare le strategie trovate.

Per scegliere la grandezza di split bisogna considerare il trade-off tra:

- training set troppo grande: generalmente per algoritmi che includono reti neurali è normale considerare un training set di grandi dimensioni, poiché le reti tendono intrinsecamente a imparare molto di più avendo un dataset di training più grande possibile a disposizione. Un problema da non sottovalutare sta nel fatto che un training set di dimensioni maggiori equivale a un maggior tempo di apprendimento, soprattutto quando si pretende che statisticamente sia preso in considerazione tutto il dataset di train.
- training set troppo piccolo: la rete in questo caso potrebbe non avere abbastanza dati su cui sviluppare una strategia vincente. Una caratteristica comune delle reti neurali infatti sta nella necessità di avere a disposizione un grande quantità di dati per poter convergere a una soluzione accettabile con possibilità di generalizzazione.

Tenendo conto di questi problemi si è presa in considerazione una configurazione spesso utilizzata: 50% dati di training, 25% dati di validation e 25% dati di test.

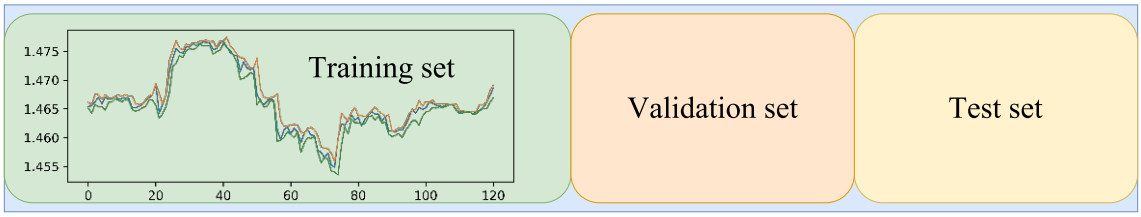



Figura 4.3: Divisione del dataset

Per una maggiore stabilità e credibilità dei risultati viene spesso utilizzata una tecnica differente, chiamata *cross-validation*. Essa consiste nella divisione del dataset in in $n$ gruppi di dati, su cui effettuare ciclicamente il training del modello predittivo su $n$-1 gruppi, effettuando il testing sul gruppo non preso in considerazione durante la fase di training, offrendo una stima del modello molto più precisa. Non è stato possibile utilizzare una configurazione del genere per due motivi principali:

- framing del problema, forex e serie temporali: i dati appartenenti al mondo del forex sono intrinsecamente sequenziali. Potrebbe essere controproducente effettuare il training del modello su dati temporalmente posteriori a quelli di test e non si rispecchierebbero le caratteristiche del campo d'applicazione.
- tempo computazionale: computazionalmente la cross-validation è molto più onerosa. In particolare se si decide di effettuare una $k$-fold cross validation si ha la necessità di fare il training di un modello esattamente $k$ volte, aumentando a dismisura il tempo di computazione all'aumentare delle k-fold. Si veda la Sezione 4.8 per un'ulteriore approfondimento.

### 4.3.1 Fasi di apprendimento

Il training dell'agente avviene quindi nelle seguenti modalità. Considerando il dataset come un insieme di ore disposte in modo sequenziale, la fase di training si concentra sulla prima parte del dataset. Vengono presentate all'agente settimane casuali all'interno del training set, partendo da un'ora all'interno di esso e considerando dall'ora $x$ considerata, le 120 successive come costituenti la settimana di training. Al termine della settimana vengono resettati i parametri episodici dell'ambiente, facendo sì che il training possa ripartire da un'altra ora campionata casualmente all'interno del training set.

Ogni $n$ episodi di training viene effettuato un salvataggio temporaneo dei pesi della rete neurale. Essi vengono caricati da un secondo modello che effettua la validazione sul validation set. Quest'ultima avviene in modo sequenziale e tutte le settimane incluse nel validation set vengono testate, per ricavare una valutazione complessiva della strategia che l'agente ha imparato durante la fase di training. Nel momento in cui il modello ottiene buoni risultati sul validation set, si effettua un salvataggio dei pesi utilizzati per

la validazione. Le configurazioni che hanno ottenuto i migliori risultati sul validation set vengono salvate e utilizzate per il test finale, in modo da verificare se i guadagni ottenuti si riflettono effettivamente su dati mai visti dall'algoritmo.

Il test delle strategie migliori avviene anch'esso in modo sequenziale in modo totalmente congruo alla fase di validazione.

## 4.4 Environment setup

La configurazione di un ambiente che rispecchi un autentico mercato di forex trading è estremamente importante dal punto di vista della buona riuscita di implementazione dell'algoritmo e della simulazione della realtà. Per una rappresentazione ottimale del mercato si è deciso di creare 3 tipologie di ambiente, con caratteristiche simili dal punto di vista implementativo, ma che si differenziano dal punto di vista di informazioni salvate, dataset utilizzato per la simulazione del mercato e plot grafici sull'apprendimento dell'agente. I tre ambienti rispecchiano la divisione del dataset, abbiamo infatti:

- *Training Environment*: utilizza i dati di training e gestisce la fase di apprendimento e l'exploit delle strategie vincenti.
- *Validation Environment*: utilizza la seconda parte del dataset, viene richiamato dall'ambiente di training per testare i risultati periodicamente sul set di validazione.
- *Test Environment*: ha a disposizione i dati di test e viene utilizzato solamente al termine del training per la verifica finale delle strategie salvate. Queste strategie corrispondono a quelle che hanno ottenuto i migliori risultati sul Validation set.

A loro volta, gli environment sono divisi in 3 componenti principali: Environment Interface, Environment Simulator ed Environment DataSource.

### 4.4.1 DataSource

Le serie temporali utilizzate per ricreare il mercato del Forex vengono salvate in un oggetto chiamato DataSource. Ad ognuno dei 3 environment corrispondono archi temporali differenti, a seconda che esso sia il DataSource di training, validation o test. Durante la fase di inizializzazione ci si occupa del caricamento di dati presenti all'interno di un file, allocandoli in memoria in modo che possano essere sempre accessibili dal simulatore. Il ruolo del DataSource è fondamentale poiché provvede a organizzare la fornitura delle osservazioni a seconda della fase dell'algoritmo. Infatti si comporta diversamente nelle seguenti casistiche:

- fase di training: durante questa fase la DataSource stabilisce gli episodi di training in modo casuale all'interno del training set su cui l'agente ha la possibilità di allenare la

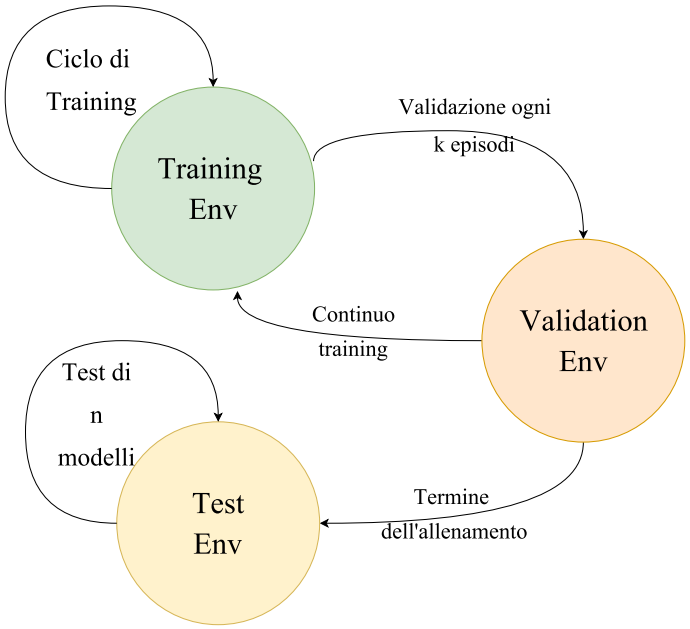



Figura 4.4: Stati e interazione tra gli environment.

propria strategia. Ad ogni nuovo episodio si occupa di definire la porzione seguente di dataset da utilizzare come nuovo episodio di trading e fornisce, ora per ora, tutte le osservazioni consecutive per le 120 ore corrispondenti alla durata dell'episodio.

- fase di validation: considerando che la validazione viene fatta in modo sequenziale, in questo caso al termine di un episodio non verrà definito una nuova ora di inizio. L'inizio del nuovo episodio corrisponde all'ora seguente e i parametri e il contatore indicante lo step attuale all'interno dell'episodio vengono semplicemente resettati.
- fase di test: similmente alla fase di validation, il test viene effettuato sequenzialmente su tutti i dati appartenenti alla porzione finale del dataset.

Avendo a disposizione tutti i dati prima che siano presentati alle altre componenti e all'agente, la DataSource svolge l'importantissima funzione di standardizzazione dei dati, trattata ampiamente nella Sezione 4.2.3.

### 4.4.2 Interface

L'agente ha la possibilità di interagire con l'ambiente attraverso l'Environment Interface. Essa consiste in una estensione dell'interfaccia messa a disposizione da OpenAI Gym adattata al trading, utilizzabile tramite funzioni di callback[1] da parte dell'agente per ricevere le osservazioni dello stato attuale e il reward associato alle azioni scelte. Allo stesso tempo essa è uno strumento essenziale per l'inizializzazione delle altre due componenti, il

Simulator e il DataSource. Essi saranno utili a reperire le informazioni necessarie all'agente per quanto riguarda i dati di training e la simulazione del mercato del Forex.

Durante l'allenamento dell'agente, per monitorare i suoi miglioramenti in ogni momento, è opportuno utilizzare grafici esplicativi e real time (approfondimento nella Sezione 4.9). Il reperimento, l'elaborazione e la graficazione dei dati avviene all'interno dell'interfaccia, poiché quest'ultima ha accesso a tutte le informazioni scambiate tra l'agente e le altre due componenti.

Risulta importante precisare che il progetto è stato impostato in modo che gli iperparametri e le opzioni delle varie componenti dell'algoritmo siano modificabili attraverso un file di configurazione in formato JSON. L'interfaccia assume un ruolo rilevante poiché viene utilizzata come mappatura tra il file di configurazione e i parametri delle componenti DataSource e Simulator, istanziati durante la fase di inizializzazione.

### 4.4.3 Simulator

Il simulatore rappresenta la parte operativa dell'environment di trading. In questa sezione dell'Environment si mira a ricreare il vero e proprio mercato del forex. Essendo intermediario tra la DataSource e l'Interface, esso ha a disposizione sia i dati riguardanti lo stato dell'agente, sia le osservazioni dirette sullo stato del mercato. Di conseguenza le funzioni svolte sono molteplici:

- Interazione con la DataSource: a ogni step riceve le osservazioni sull'environment, appositamente standardizzate da parte del DataSource.
- Salvataggio degli stati: il Simulator viene utilizzato per salvare sia lo stato dell'Enviroment, composto dalle osservazioni comprendenti tassi di cambio, volumi e indicatori, sia lo stato dell'agente, come esposizione sul mercato e PMC.
- Preprocessing delle osservazioni e dello stato dell'agente: prima di inoltrare le osservazioni all'interfaccia, il simulatore le processa in modo da unirle alle informazioni riguardanti lo stato dell'agente, così da presentare informazioni complete riguardo il suo stato, la sua attuale posizione rispetto al mercato e l'environment.
- Calcolo del profit e del reward: avendo a disposizione tutte le informazioni necessarie, esso si occupa di applicare la funzione di reward data l'azione dell'agente, il suo stato e l'osservazione sullo stato dell'environment.
- Reset dell'ambiente: alla fine di ogni episodio esso azzera le informazioni necessarie in modo da reinizializzare la settimana di trading.

[1]Callback function: blocco di codice passato come parametro a un'altra funzione, attivata al verificarsi di determinati eventi o al termine della funzione chiamante.

- Mantenimento delle informazioni necessarie per la graficazione: esso salva le informazioni riguardanti tutte le azioni svolte dall'agente, il momento in cui sono state compiute, reward e profit associati, gli stati derivanti e il guadagno medio e cumulativo.

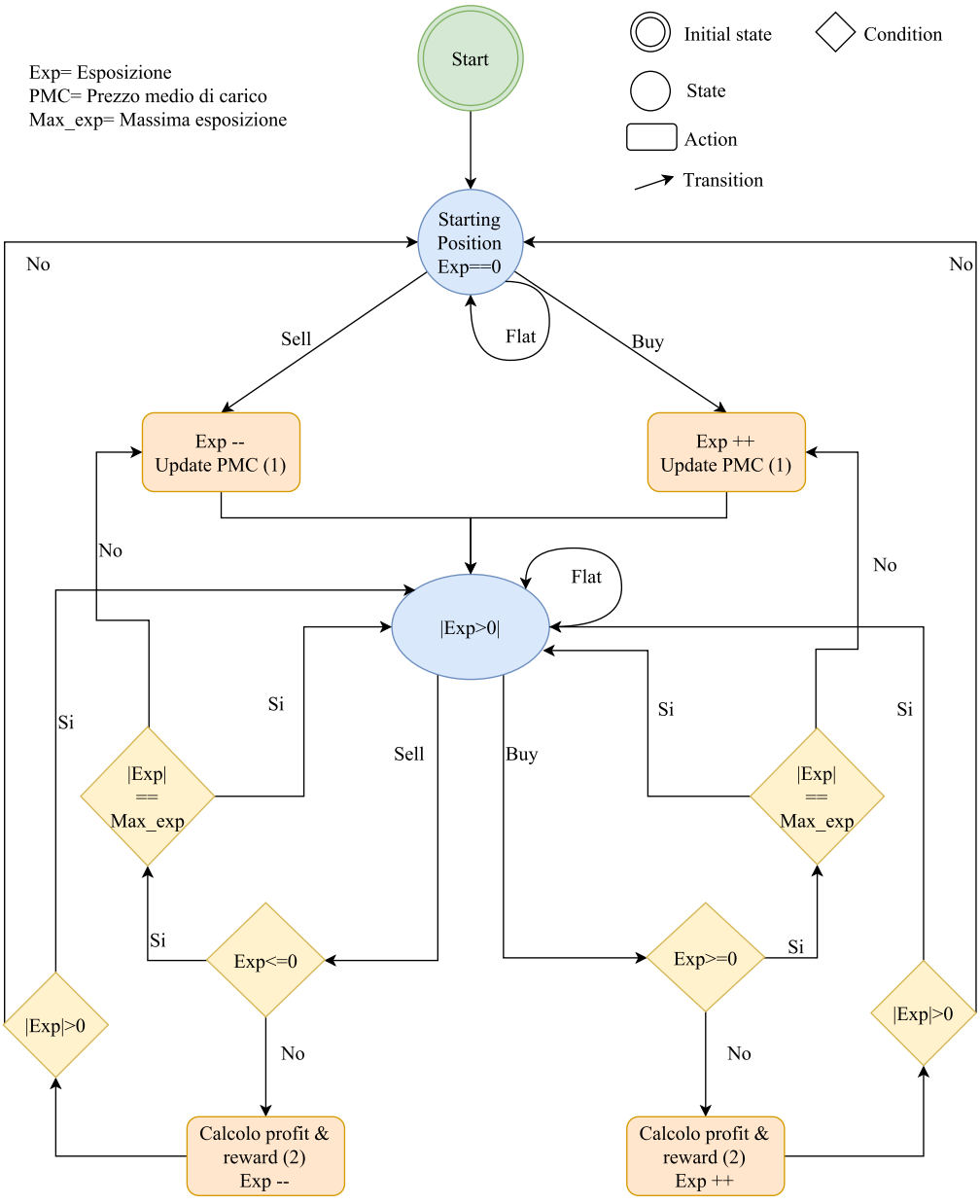



Figura 4.5: Diagramma a stati del Simulatore di trading

Viene riportato alla Figura 4.5 un diagramma comprensivo degli stati, delle transizioni di stato e delle funzioni attivate per ogni transizione di stato. Vengono segnalate anche le azioni in cui viene effettuato il calcolo del PMC con (1) e le azioni in cui viene effettuato il calcolo del profit e del reward con (2), illustrati di seguito nella Sezione 4.5 .

## 4.5 Funzione di reward e calcolo del profit

La funzione di reward è la componente che si occupa di stabilire il reward associato a ogni coppia stato/azione. Il calcolo di questo valore viene effettuato grazie a diversi fattori:

- Stato dell'agente
  Esposizione: $exp$
  Prezzo medio di carico: $PMC$
  Azione selezionata $a$
- Stato dell'environment
  Valore del tasso di cambio EUR/USD

Si è deciso, per quanto riguarda l'ambito di questo progetto, di dividere il concetto di reward dal concetto di *profit*.

Il profit rappresenta un valore calcolato a ogni azione facendo riferimento alle informazioni descritte in precedenza e corrisponde esattamente al guadagno in percentuale rispetto all'investimento unitario effettuato dall'agente in caso di apertura di una posizione. Su di esso si calcolano tutte le metriche quantitative per la valutazione delle prestazioni dell'agente. Con il termine di reward ci si riferisce a un valore calcolato in base al profit. Mentre il reward fa parte degli input dell'agente assieme allo stato successivo dell'environment, il profit è un valore a cui l'agente non ha accesso direttamente, ma viene sfruttato all'interno dell'environment per calcolare opportunamente il reward oltre a essere utile per la valutazione finale delle prestazioni dell'algoritmo.

Per introdurre opportunamente il calcolo del profit e le funzioni di reward è necessario definire il concetto di Prezzo Medio di Carico o PMC. Il prezzo medio di carico indica l'attuale posizione di investimento del trader calcolata in base all'apertura delle posizioni e al valore del tasso di cambio nel momento dell'apertura di ogni singola posizione.

Indicando con $PMC_t$ il prezzo medio di carico, $EUR/USD_t$ il valore di tasso di cambio euro/dollaro americano, $exp_t$ l'esposizione al tempo $t$ e con $exp_{t-1}$ la posizione allo step precedente, il $PMC$ si aggiorna solamente nel caso di apertura di una posizione, in particolare:

**if** $|exp_t| > |exp_{t-1}|$ **then**

$$PMC_t \leftarrow PMC_{t-1} * \left|\frac{exp_{t-1}}{exp_t}\right| + \frac{EUR/USD_t}{|exp_t|} \tag{4.7}$$

**end if**

Come si vedrà in seguito il PMC è un valore fondamentale per calcolare il profit in caso di chiusura della posizione. Viene fatto riferimento al calcolo del PMC nella Figura 4.5 nelle azioni segnate con (1).

Si può dunque descrivere la composizione del calcolo del profit come segue:

**if** $|exp_t| > |exp_{t-1}|$ **then**
    # Caso di apertura di una posizione, si pagano i costi di commissione
    $profit_t \leftarrow -0.00014$
**else**
    **if** $|exp_t| < |exp_{t-1}|$ **then**
        # Caso chisura della posizione

$$profit_t \leftarrow (exp_{t-1} - exp_t) * (EUR/USD_t - PMC_t) \tag{4.8}$$

    **end if**
**end if**

In base al profit è possibile definire tutte le funzioni di reward di cui si è fatto uso nel corso dei vari esperimenti. E' importante sottolineare che l'agente, durante la fase di training, cercherà di aumentare il reward cumulativo finale e la funzione di reward risulta fondamentale ai fini dell'apprendimento di una strategia opportuna da parte dell'agente.

### Profit as reward

La scelta più naturale è quella di utilizzare il profit direttamente come reward, dandolo in input all'agente senza modificarlo in alcun modo. L'agente di conseguenza punta a ottenere direttamente il più alto guadagno percentuale possibile. Come si vedrà dai risultati, anche se questa potrebbe sembrare la scelta più sensata, spesso non si rivela essere la migliore dal punto di vista della riuscita dell'apprendimento di una buona strategia.

### Clipped reward

Altamente più ingegnerizzata è la funzione di reward che mappa i valori continui del profit a valori discreti. Si è deciso di utilizzare una funzione discretizzante per due motivi principali:

- semplificare la funzione di profit reward: avendo ottenuto risultati scarsi utilizzando il profit come reward, si è attribuita parte del mancato apprendimento alla difficoltà

per l'agente di trovare una policy che potesse imparare a ricevere un reward continuo variabile tra $[-\infty, +\infty] \in R$. Si ritiene infatti che in questo modo l'agente possa imparare più agilmente ad attribuire il giusto valore a ogni coppia stato/azione, in modo da potersi adattare maggiormente all'ambiente pur non avendo un feedback diretto per quanto riguarda il profit;

- riprodurre parzialmente la funzione di reward utilizzata nel paper della DQN per i giochi Atari [19], in cui si stabilisce un reward discretizzato equivalente a [-1, 0, 1] nel caso in cui il punteggio aumenti, diminuisca o rimanga costante.

La funzione di reward clipped viene calcolata utilizzando la metodologia seguente:

**if** $profit \leq -0.02$ **then**
  $reward = -4$
**else if** $-0.02 \leq profit < -0.00014$ **then**
  $reward = -2$
**else if** $profit == -0.00014$ **then**
  $reward = -0.1$
**else if** $-0.00014 < profit \leq 0$ **then**
  $reward = 0$
**else if** $0 < profit < 0.015$ **then**
  $reward = 0.4$
**else if** $0.015 < profit < 0.03$ **then**
  $reward = 1$
**else if** $profit \geq 0.03$ **then**
  $reward = 2$
**end if**

Si è deciso di dare un reward maggiormente negativo in caso di perdita per far capire all'agente che è molto più importante trovare una strategia che permetta di non perdere molto rispetto a una strategia che permetta guadagnare molto ma comportando grandi perdite.

### Cumulative reward e Mean cumulative reward

Il reward cumulativo viene calcolato semplicemente come la sommatoria del profit dal primo step fino allo step $t$-esimo all'interno dell'episodio.

$$cum\_reward_t = \sum_{i=0}^{t} profit_i \tag{4.9}$$

Preso singolarmente il reward cumulativo non permette all'agente di estrarre strategie remunerative, ma può essere utilizzato per il calcolo del reward medio:

$$mean\_cum\_reward_t = \frac{cum\_reward_t}{n} \tag{4.10}$$

dove $n$ in dica il numero di step dall'inizio dell'episodio.

### Weighted profit - mean cumulative reward

Un'ultima configurazione di reward utile da segnalare è la funzione composta dalla combinazione del reward medio cumulativo e il profit per trade, calcolato come:

$$weighted\_reward_t = \alpha * mean\_cum\_reward_t + (1 - \alpha) * profit_t \tag{4.11}$$

dove $\alpha$ indica il peso da assegnare alle due parti ed è un numero compreso tra [0,1].

## 4.6 Obiettivo finale e drawdown

Durante il training l'agente ha la possibilità di esplorare ampiamente lo spazio delle soluzioni in modo da trovare la strategia che gli permette di massimizzare il valore atteso del reward cumulativo futuro. Accade spesso però che per raggiungere questo risultato sia necessario modificare la funzione di reward che l'agente effettivamente prova a massimizzare, poiché è necessario strutturarla in modo che esso possa capire il metodo migliore per mappare le varie configurazioni degli stati di input alle azioni opportune.

L'obiettivo finale del progetto non è massimizzare il reward che l'agente effettivamente vede, ma massimizzare il profit che corrisponde al guadagno reale. Risulta inoltre opportuno definire con che modalità l'agente dovrebbe idealmente arrivare al guadagno massimo: esso infatti dovrebbe trarre profitti avendo il minor *drawdown* possibile. Si definisce *drawdown* la differenza percentuale tra il massimo valore di guadagno nella serie di investimento e il successivo punto di perdita massima. Più semplicemente esso rappresenta la perdita massima subita nel tempo dall'agente. Il problema dal punto di vista algoritmico è che il drawdown introduce anche il concetto di rischio, poiché indica intrinsecamente qual'è la percentuale di possibile perdita per ottenere il guadagno finale indicato. Le strategie ad alto rischio (ad alto drawdown), anche se offrono la garanzia di ottenere un buon profitto finale, ne compromettono l'usabilità all'interno di un contesto finanziario in cui si investono ingenti quantità di denaro. In genere è preferibile costruire una strategia che guadagni in modo lineare, senza avere di picchi di perdita troppo ampi, rispetto a una strategia che sul lungo termine guadagni di più ma comporti un rischio molto più alto.

## 4.7 Setup dell'agente

L'agente è la componente che rappresenta il cuore di un algoritmo di Deep reinforcement learning, esso infatti si occupa di prendere le decisioni, adattandole nel tempo alla strategia che permette di ricavare il reward futuro più elevato possibile. Un agente è formato dalle componenti base presentate nel terzo capitolo alla Sezione 3.1. Queste componenti sono la policy, la memoria e la rete neurale. Inoltre si definiscono una serie di altri parametri e caratteristiche inerenti alle modalità con cui l'agente può interagire con l'environment. Esistono moltissime tipologie di agenti e di implementazioni differenti. Nella sezione seguente si espongono nello specifico le scelte implementative relative alle componenti e alle restrizioni date a un agente di Q-Learning nell'interazione con il mondo esterno.

### Azioni

L'agente ha la possibilità di compiere solamente 3 azioni discrete nell'interazione con l'environment. Esso può infatti decidere di aumentare, diminuire o tenere invariata la propria posizione tramite le azioni di buy, sell o flat. Ognuna di queste azioni però ha una conseguenza differente a seconda dell'esposizione dell'agente e dello stato dell'environment. L'azione di buy e quella di sell possono infatti corrispondere all'apertura o alla chiusura di una posizione, a seconda del valore dell'esposizione che può risultare positivo o negativo al momento dell'azione. Oltre a questo, all'agente non viene permesso di superare una esposizione determinata da $max\_exposition$, in modo da limitare la possibilità di andare in sovraesposizione. Il motivo di questa scelta è semplice: dopo aver effettuato alcuni test senza limitazione dell'esposizione, si è notato che l'agente, nel caso in cui compisse una azione errata andando nella direzione opposta al mercato, tendeva successivamente a compiere una serie azioni che tenevano invariata o aumentavano la sua esposizione a meno che il mercato non invertisse il trend. Questo determinava una impossibilità da parte dell'agente a imparare a chiudere le posizioni in caso di grandi perdite.

Nella fase iniziale, durante una ricerca approfondita della configurazione dell'algoritmo, sono state utilizzate altre tipologie di azioni. Un esempio pratico è l'azione "close all positions", che permetteva all'agente di riportare la propria esposizione a 0 con un'unica azione. Tale prova, come tutti i tentativi con configurazioni differenti hanno comunque ottenuto risultati più scarsi rispetto alla semplice modalità con con 3 azioni.

### Policy

La policy caratterizza fortemente il risultato finale, essa rappresenta infatti la politica decisionale dell'agente. La policy utilizzata per l'allenamento dell'agente è la $\epsilon$-greedy policy come descritta nel capitolo 3. I parametri più importanti sono sicuramente i valori iniziali e finali di $\epsilon$ e il numero di step di decadimento. Il decadimento nel tempo è dato da un valore proporzionale al numero di passi selezionati per far sì che $\epsilon$ passi da un

valore di $\epsilon\_max$ a un valore di $\epsilon\_min$. Questo significa che con l'aumentare del numero di step l'agente prenderà con più probabilità le decisioni che seguono la policy ritenuta ottima in quel momento. La variazione del decadimento ha impatto diretto sull'exploitation della policy ottima e si ripercuote concretamente anche sul reward medio per episodio sul training set ottenuto dalla policy finale. Generalmente infatti la scelta di un decadimento più lento permette all'agente di valutare più approfonditamente lo spazio degli stati e delle azioni, approssimando più agevolmente quella che può essere la policy più proficua. Sfortunatamente la crescita del valore di decadimento ha delle forti ripercussioni sia dal punto di vista del tempo di apprendimento, sia della generalizzazione dell'algoritmo sul validation e test set.

In una fase prototipale del progetto sono stati effettuati test utilizzando la Policy di Boltzmann. Essa prevede di scegliere le azioni con probabilità proporzionale al valore di reward aspettato assegnato a esse da parte dell'agente. Dopo alcuni confronti effettuati con la $\epsilon$-greedy policy si è notato che quest'ultima raggiungeva risultati nettamente migliori e più significativi, perciò si è deciso di abbandonare la Policy di Boltzmann.

#### Memoria

La memoria è la componente che conserva un numero $n$ di esperienze passate dell'agente. Ogni esperienza è formata da una tupla $< s_t, a_t, r_t, s_{t+1} >$. Tramite quest'ultima l'agente impara contemporaneamente da esperienze passate e presenti, stabilizzando il proprio apprendimento. Il parametro importante della memoria consiste nel numero $n$ di esperienze passate da tenere in considerazione ed è molto importante ai fini dell'apprendimento di una buona strategia. Le esperienze salvate nella memoria vengono utilizzate per aggiornare i pesi della neural network utilizzando la funzione illustrata al paragrafo 3.2.4.

### 4.7.1 Neural Network

La neural network rappresenta la logica decisionale dell'agente nonché la funzione di approssimazione che mappa ogni configurazione degli stati di input all'output che rappresenta l'importanza di ogni azione, i.e. valore dell'action-value function corrispondente a ogni coppia stato-azione. La tipologia di neural network adottata all'interno di questo progetto è una rete a layer completamente connessi. Il modello a layer densi è uno delle prime architetture inventate e risulta di semplice implementazione. Non si è avuta la possibilità di esplorare un grande numero configurazioni per quanto riguarda le tipologie di rete e gli iperparametri di queste ultime. Si è deciso al contrario di dare maggiore priorità agli iperparametri riguardanti l'algoritmo di reinforcement learning puro e di adottare un approccio standard per quanto riguarda la neural network.

Il parametro più influente nell'ambito delle reti neurali, oltre al numero di neuroni e di layer, è sicuramente il learning rate dell'ottimizzatore RMSProp. La funzione utilizzata

per l'aggiornamento dei pesi della rete è quella custom definita dall'algoritmo di Deep Q-Learning come definito nel capitolo 3 al paragrafo 3.2.4.

## 4.8 Computational time

Generalmente gli algoritmi di reinforcement learning richiedono molto tempo per convergere alla policy ottima. Per le prove nell'ambito di questo progetto si è avuto a disposizione una GPU NVIDIA GTX 970. Mediamente, a seconda delle configurazioni e a seconda del raggiungimento o meno della patience prestabilita, un modello impiega tra le 10 e le 15 ore per arrivare a testare i risultati migliori sul test set. Questo ha un enorme impatto sulle tempistiche con cui si arriva alle conclusioni, infatti una qualsiasi modifica di configurazione di un unico parametro del modello, che riguardi la rete neurale, l'agente o l'environment, richiede una nuova fase di training. Impostare alcuni set di iperparametri potrebbe portare di conseguenza all'impiego di moltissimo tempo. Una delle difficoltà maggiori riscontrate durante lo sviluppo del progetto riguarda proprio questo punto: la configurazione degli iperparametri. Spesso accade infatti che lo spazio delle soluzioni sia esplorabile in modo esaustivo tramite una ricerca degli iperparametri utilizzando la tecnica di *grid-search*. Questo significa che è possibile configurare l'algoritmo in modo che esplori lo spazio delle soluzioni attraverso una ricerca che combina tutti i valori possibili in modo da offrire, al termine della ricerca, il ranking delle configurazioni con risultati migliore. In questo caso considerando l'onere computazionale del deep reinforcement learning è stato necessario eseguire una ricerca molto più mirata, in modo da evitare la perdita di tempo dovuta a configurazioni poco significative.

## 4.9 Plot

Grazie all'implementazione dell'environment è possibile salvare tutte le informazioni riguardanti l'allenamento dell'agente, le metriche di training, test e validation, inclusa la possibilità di visualizzare in tempo reale lo stato di apprendimento dell'agente.

Sono stati utilizzati una serie di grafici estremamente importanti per monitorare l'avanzamento e il miglioramento delle strategie elaborate dall'agente durante la fase di training, facendolo interagire l'ambiente di validation, training e test. Diamo dunque una breve descrizione dei grafici e della loro utilità all'interno del reinforcement learning, facendo particolare riferimento al particolare caso del Forex Trading.

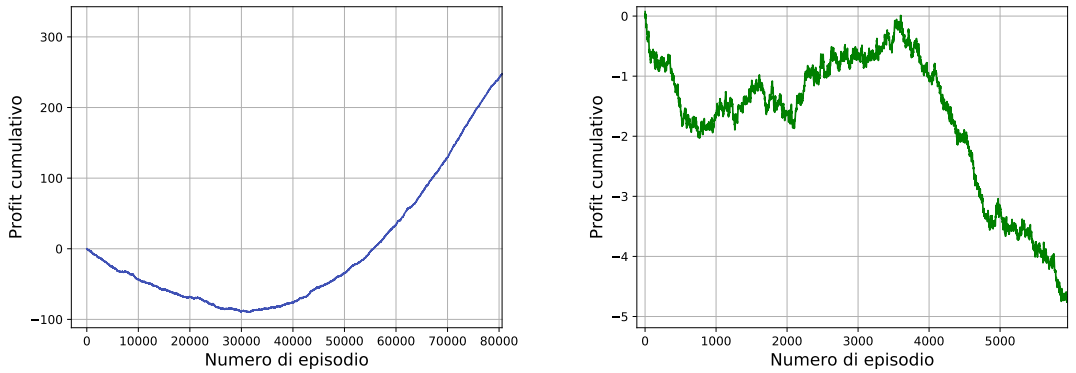
### Plot del profit cumulativo

Il grafico riportato nella Figura 4.6 rappresenta il profit cumulativo dell'agente nel tempo. Esso risulta importante per capire se, con l'avanzare degli episodi di training, l'agente sta guadagnando o perdendo in percentuale alla quantità corrispondente a ogni singola

apertura di posizione. Il guadagno è in percentuale poiché si considera un'apertura di posizione sempre di dimensione unitaria. In particolare una pendenza positiva del grafico indica guadagno, mentre una pendenza negativa indica perdita.

Nel grafico del profit cumulativo sul training set ci si aspetta un andamento inizialmente negativo, poiché la policy di $\epsilon$-greedy fa si che l'agente agisca in modo completamente random. Idealmente però, andando avanti nel tempo, la pendenza della curva dovrebbe passare gradualmente da negativo a positivo, per poi raggiungere la massima pendenza rappresentante il raggiungimento del massimo guadagno medio per episodio.

Nella Figura 4.6 grafico (a) abbiamo un esempio concreto di come dovrebbe essere esattamente l'andamento della curva di training del reward cumulativo.




(a) (b)

Figura 4.6: Rappresentazione grafica del profit cumulativo nel tempo. La figura di sinistra rappresenta un esempio dell'andamento del profit cumulativo nel caso di training, la figura di destra rappresenta invece il caso di validation. L'asse delle ascisse corrisponde al numero dell'episodio, l'asse delle ordinate corrisponde al profit cumulativo per episodio di trading.

Per quanto riguarda il profit cumulativo della fase di validation ci si aspetta che segua un andamento molto più particolare, poiché in questo caso, dopo aver imparato a guadagnare e trovando generalizzazione sul validation set, l'algoritmo tende sempre più a convergere a una soluzione ottimale sul training set causando il fenomeno di overfitting, a discapito della generalizzazione. Questo si rispecchia in un abbassamento del profit medio sul validation set. L'andamento tipico del profit cumulativo del validation set è visibile nel grafico (b) della Figura 4.6

Il reward cumulativo del test set, essendo solamente una prova riguardante i modelli migliori, dovrebbe approssimare una linea retta con pendenza positiva, così da sottolineare un guadagno anche sul test set dovuto all'utilizzo della policy ottima.

### Plot del profit medio nel tempo

Un altro grafico estremamente importante è quello dato dal profit medio per episodio (Figura 4.7). Esso rappresenta la media del profit degli ultimi $n$ episodi nel passato. Risulta significativo poiché permette di capire il momento in cui l'agente non è più in grado di imparare nuove strategie per migliorare il suo guadagno medio. Si riesce inoltre a definire l'arco di tempo in cui la strategia imparata satura e, portandosi a pendenza media 0, stabilisce il massimo livello di guadagno medio raggiunto.

Ci si aspetta dunque che nel caso di training il grafico abbia un andamento crescente fino ad arrivare al momento di saturazione finale, auspicando che la strategia raggiunta permetta di avere una media di profit per episodio positiva.

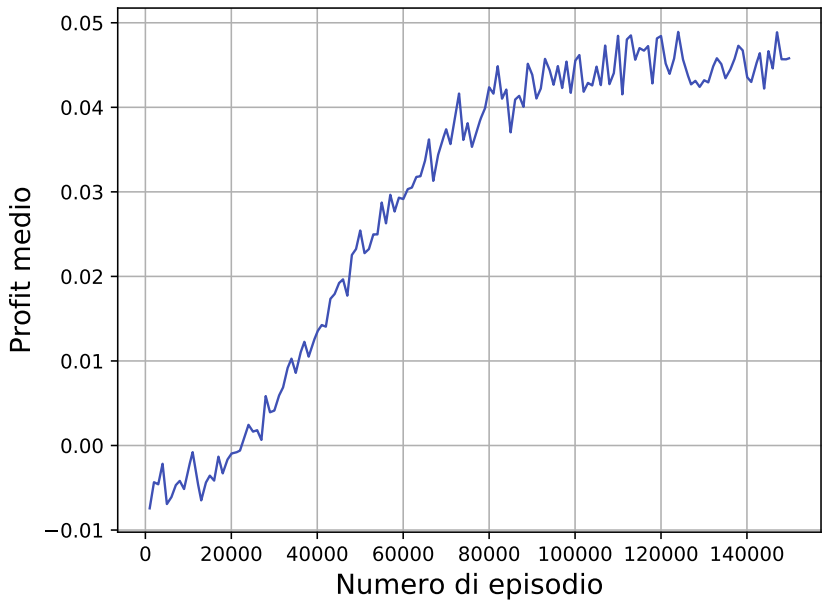



Figura 4.7: Profit medio - Training set

Nel caso di validation invece l'andamento sarà sicuramente più irregolare, dato che in ogni momento l'algoritmo potrebbe trovare nuove soluzioni e strategie per guadagnare sia sul set di training che su quello di validation. Si trova però sicuramente corrispondenza con una parte di grafico coincidente a un andamento negativo dovuto all'overfitting sul training set.

### Plot dei singoli episodi di trading

Per valutare in modo più specifico le differenti strategie adottate dall'agente si è reso necessario visualizzare le azioni compiute a ogni singolo step di trading. In questo caso si è deciso di salvare, per ogni episodio di test, tutta la storia di quell'episodio sotto forma di azioni (long, short, flat). Vengono anche segnalati i casi in cui l'agente tenta di compiere azioni non consentite in un particolare istante di tempo, cioè quelle che tentano di aumentare o superare la massima esposizione consentita avendo già raggiunto la massima esposizione possibile. Si salvano inoltre i reward per step e l'esposizione attuale.

Nella Figura 4.8 si possono notare 4 differenti plot riguardanti uno stesso episodio di test. L'asse delle ascisse è condiviso e rappresenta lo step all'interno dell'episodio a cui corrispondono gli altri valori. Il plot più in alto rappresenta l'andamento del tasso di cambio euro/dollaro nel tempo, standardizzato alle 4 settimane precedenti. Ogni punto colorato sul rate rappresenta l'azione svolta dall'agente, in particolare:

- Pallino verde: azione buy consentita
- Pallino verde chiaro: azione buy non consentita
- Pallino rosso: azione sell consentita
- Pallino arancio: azione sell non consentita
- Pallino nero: azione flat

Con "non consentita" si intende che l'agente ha scelto quella determinata azione, ma la massima esposizione è stata già raggiunta precedentemente. Di conseguenza l'azione scelta dall'agente non viene presa in considerazione dal simulatore, che lo lascia fisso nella posizione precedente. Ad esempio: se $max\_exposition$ è fissata a 3 e l'esposizione attuale dell'agente corrisponde esattamente a 3, nel caso in cui l'agente tenti di eseguire una azione di buy non gli sarà consentito aumentare la propria esposizione, di conseguenza quest'ultima rimarrà invariata.

Ad ogni azione corrisponde un profit variabile in un intervallo reale. Il secondo grafico rappresenta il profit associato all'azione effettuata, calcolato in base al tasso di cambio EUR/USD e all'attuale PMC. Si ricorda che il profit in caso di apertura di una posizione corrisponde ai costi di commissione. Si è deciso di inserire un grafico molto simile riportante il reward per step poiché esso non corrisponde necessariamente al profit, ma è calcolato in base a esso. Il terzo grafico rappresenta il reward per step.

L'ultimo grafico in Figura 4.8 rappresenta invece l'attuale esposizione dell'agente sul mercato. L'agente ha la possibilità di esporsi di un valore unitario, di conseguenza l'esposizione più variare in un intervallo discreto [$-min\_exp$,$+max\_exp$]. Quest'ultima varia a seconda delle mosse effettuate dall'agente durante l'episodio.

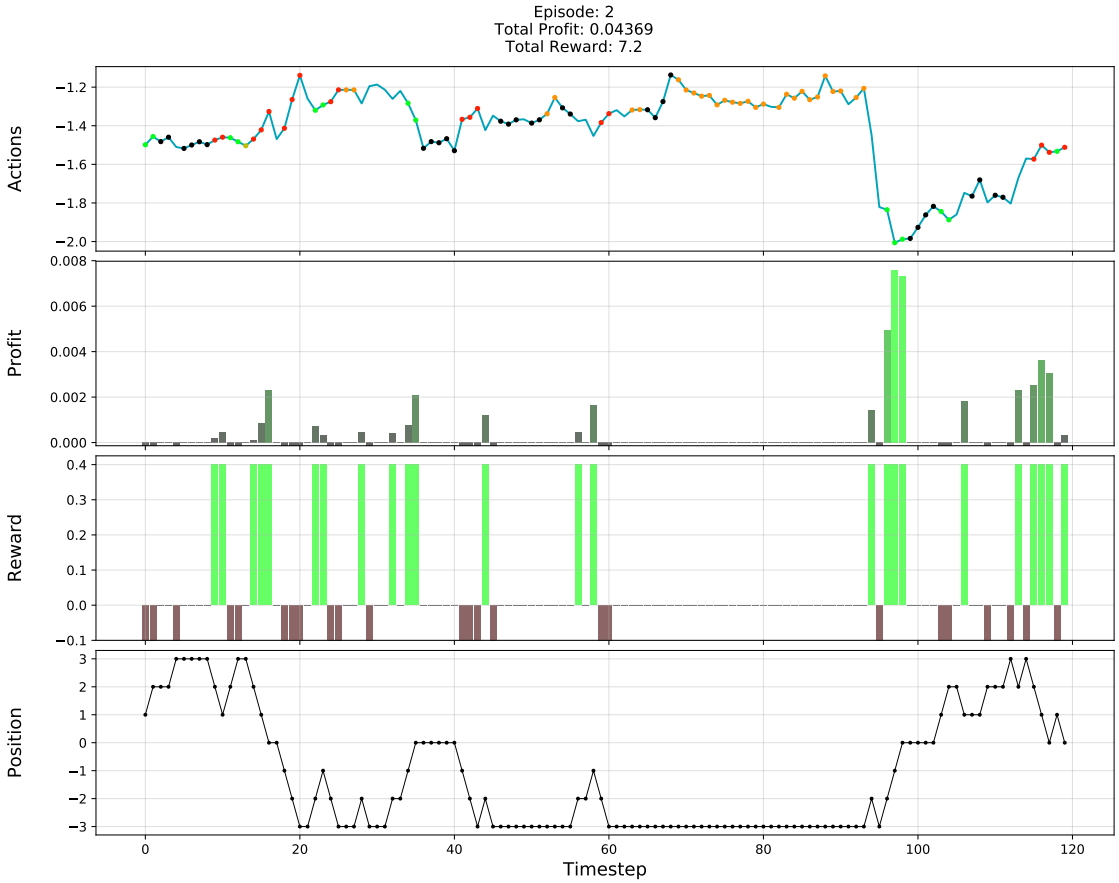



Figura 4.8: Grafico di un episodio di trading

# Capitolo 5

# Risultati

L'applicazione di tecniche di Deep Q-Learning al mercato finanziario e in particolare al mercato del Forex hanno portato a risultati interessanti per quanto riguarda l'apprendimento automatico di un agente a investire in modo autonomo. Sono osservabili una moltitudine di trend differenti in tutte le fasi associate all'apprendimento dell'agente: training, validation e test. Nell'ambito di questa tesi si è cercato di coprire nel miglior modo possibile lo spettro delle soluzioni per quanto riguarda l'utilizzo degli iperparametri più importanti del Reinforcement learning e in particolare del Deep Q-Learning.

Si presentano in questo capitolo le modalità di valutazione dei risultati, introducendo alcune problematiche specifiche riguardo la compatibilità tra algoritmi di Deep Q-Learning all'interno del mondo finanziario. Si offre in seguito una ampia vista nell'ambito dell'esplorazione degli iperparametri e della ripercussione della variazione degli stessi sui risultati finali.

## 5.1 Considerazioni iniziali

Prima di affrontare i risultati reali, si ritene necessario dedicare particolare attenzione ad alcune considerazioni che contraddistinguono il mondo deterministico dei giochi Atari, dominio nel quale il Deep Q-Learning ha ottenuto risultati entusiasmanti sotto molti punti di vista. Si osserva prima quindi la differenza comportata dalla variazione dei differenti iperparametri all'interno del training, validation e test set, facendo alcune considerazioni finali inerenti al mercato finanziario e all'impredicibilità di alcune tipologie di eventi all'interno del Forex Trading.

### 5.1.1 Fase di training

Nel caso del training set è spesso possibile osservare ripercussioni dirette durante l'esplorazione dei vari iperparametri. Esso infatti rispecchia nel modo migliore quello che può

essere il dominio dei giochi Atari, poiché l'agente impara e applica le strategie in un mondo che può avere un numero molto grande ma finito di stati. Come un agente impara a giocare a un gioco con regole e possibili mosse prestabilite, la stessa cosa succede anche in nell'ambito del Forex, poiché il fatto che l'agente abbia la possibilità di esplorare e conoscere l'ambiente gli permette di capire quali siano le strategie migliori nel mondo in cui si è trovato ad allenarsi. Per questo motivo il cambiamento degli iperparametri di apprendimento incidono direttamente sui risultati riguardanti il set di training offrendo la possibilità di confrontare al meglio le varie configurazioni e distinguere i risultati tra loro.

### 5.1.2 Fase di validation

Risulta estremamente più complesso effettuare osservazioni dirette per quanto riguarda il set di validazione. Si ricorda infatti che l'agente non ha alcun modo di allenarsi né sul set di validazione né sul set di test e semplicemente il validation serve per effettuare la valutazione delle strategie imparate durante il training su dati non visti, per capire quanto le strategie imparate possano essere generalizzabili. Dal punto di vista dei risultati in questo ambito spesso risulta estremamente utile fare osservazioni su una funzione di reward più semplice, motivo per il quale quest'ultima è stata modificata in modo da restituire, invece di valori continui, valori discreti possibilmente più semplici da interpretare dal punto di vista dell'agente. Utilizzando questo espediente è possibile visualizzare trend di apprendimento più significativi sul set di validazione.

Non è sempre possibile riscontrare una corrispondenza precisa per quanto riguarda il training set e il validation set e ogni configurazione degli iperparametri comporta differenti comportamenti soprattutto sul validation.

Si possono fare considerazioni simili per quanto riguarda l'utilizzo della stessa strategia sul validation e sul test set. L'osservazione sicuramente più evidente da fare risulta il fatto che un'ottima prestazione sul validation e un conseguente salvataggio della strategia comporta spesso una prestazione pessima sul test set, risultando dunque in un evidente caso di overfitting sul set di validazione. La situazione descritta in precedenza non è rara, accade spesso infatti che nelle prime fasi di apprendimento, momento durante il quale l'agente non ha ancora avuto modo di approfondire le strategie migliori, esso abbia dei picchi di ottime prestazioni sul validation set. Queste prestazioni non si vanno a ripercuotere allo stesso modo sul set di test poiché la strategia non è assolutamente "generale", ma è solamente una configurazione dei pesi della neural network che porta a ottenere risultati positivi sul validation in maniera fortuita, senza rispecchiare gli stessi risultati su altre configurazioni degli input.

### 5.1.3 Fase di test

Teoricamente i risultati sul test set sono quelli più rilevanti poiché dimostrano le reali potenzialità del sistema su dati futuri e mai usati come dati di input per l'agente. Di

conseguenza i risultati mostrati in seguito dal punto di vista delle metriche vengono calcolati su questo set di dati. La differenza di prestazioni rispetto ai dati che l'algoritmo ha avuto modo di vedere è ovviamente abissale ma nettamente più significativo. Gli sforzi di più grande rilevanza sono stati senza dubbio quelli riguardanti la scoperta di una configurazione che permettesse di ottenere i migliori risultati sul test set.

**Rimozione della difesa del franco svizzero**

Si riporta il caso reale in cui un evento bancario ha sensibilmente destabilizzato il mercato economico a livello globale. Nel 2011 la Banca centrale svizzera (SNB) decise di stabilire un limite minimo di tasso di cambio a 1,20 Franchi svizzeri per euro. Ciò significa che la Banca centrale Svizzera si doveva impegnare a evitare che il franco prendesse eccessiva potenza nei confronti delle altre monete. Il motivo di tale decisione è semplice: nel 2011 ci si trovava nel pieno svolgimento di un periodo di grande crisi a livello Europeo e si riteneva che solamente il franco svizzero fosse una moneta su cui poter investire con basso tasso di rischio e alti profitti. L'aumentare della domanda di conseguenza si rispecchia nell'aumento del valore di una valuta, mettendo in grave rischio l'economia di esportazione della Svizzera. Questo accade poiché se il tasso di cambio è molto sfavorevole verso un'altra moneta, chi compra dall'estero i beni prodotti in Svizzera tenderà a cambiare paese fornitore con quello in cui il cambio risulta tendenzialmente più favorevole. Il giorno 15 Gennaio 2015 è stata presa un'importantissima decisione da parte della SNB riguardo il valore della valuta franco svizzero nei confronti delle altre valute monetarie ed è stato rimosso il blocco imposto nel 2011, comportando un aumento vertiginoso del valore del franco svizzero rispetto alle altre valute con una crescita istantanea del 39%, per poi stabilizzarsi in seguito sul 14%. Non si entra nel merito di motivi di tale decisione presa da parte della Banca Centrale Svizzera, ma è importante riportare che questo avvenimento si colloca in un momento nel tempo presente all'interno dei dati di test. Il cambio istantaneo dunque del valore di tasso di cambio EUR/CHF e la sua ripercussione sul mercato è evidente e si ripercuote in scarse prestazioni dell'algoritmo nelle settimane successive all'evento dovuta alla grandissima scossa esercitata sul mercato. Si vedrà successivamente a livello grafico dove si verificano tali perdite.

### 5.1.4 Considerazioni sul mercato del Forex

Il mercato del Forex è, come tutti i mercati finanziari, altamente imprevedibile e influenzato da diversi fattori. E' possibile rilevare nelle economie una serie di eventi molto particolari che possono portare gli andamenti dei tassi di cambio a subire variazioni casuali e assolutamente imprevedibili dal punto di vista algoritmico. Questo accade principalmente per due motivi:

- imprevedibilità intrinseca dell'evento: si possono verificare eventi di varia natura non ipotizzabili a priori che vanno a influenzare tutti i mercati finanziari, incluso quello del Forex.
- imprevedibilità della variazione del mercato allo scaturirsi dell'evento: accade spesso che non vi sia alcuna correlazione tra l'entità dell'evento e la reazione dei mercati all'evento. Di conseguenza due eventi con caratteristiche molto simili potrebbero portare gli stessi indici di mercato a movimenti differenti.

Si sono dunque identificati una serie di tipologie di eventi che vanno a influenzare in maniera più o meno marcata la imprevedibilità dei mercati.

- In primo piano vi sono le occasioni durante le quali parla un alta personalità a livello economico mondiale. Spesso qualsiasi informazione divulgata in ambito finanziario da parte di autorità di grande influenza causano una fluttuazione dei tassi di cambio dovuti alla scossa portata dalle sentenze emanate da queste autorità.
- Altro fattore importante sono le decisioni impreviste in ambito bancario per quanto riguarda direttamente l'economia mondiale ed europea. In questo caso si riporta un evento concreto che si può visibilmente notare nei risultati ottenuti sul test set. Considerando l'importanza dell'evento e i riscontri ottenuti sui risultati, esso si riporta approfonditamente nella Sezione 5.1.3
- Un'altra tipologia molto incisiva sono gli eventi in ambito politico, poiché il panorama politico di una nazione svolge un ruolo di primaria importanza per quanto riguarda l'economia dei diversi paesi. Considerando il legame strettissimo tra l'economia e la moneta appartenente al paese stesso o, nel caso dell'euro, all'economia di più paesi, anche gli eventi politici hanno riscontro diretto sugli andamenti dei mercati. in questo caso si può trovare esempio concreto nella decisione dell'uscita del Regno Unito dall'Unione Europea dopo il referendum del giorno 23 Giugno 2016. Questa decisione ha portato a un crollo netto del valore della Sterlina nei confronti delle altre valute e grandissimi riscontri sull'economia, con il coinvolgimento di banche e grandi società inglesi, comportando anche le dimissioni successive del primo ministro David Cameron.
- Ultimi ma non meno importanti risultano vari fattori legati a cause esterne o fatti di cronaca mondiale. Anch'essi infatti vanno a influenzare le economie dei paesi sotto

vari punti di vista. Si prenda come esempio uno dei più incisivi attacchi terroristici di tutti i tempi: l'attentato al World Trade Center dell'11 settembre 2001 a New York. Un evento di simile entità risulta totalmente imprevedibile da chiunque e ha impatti devastanti sull'economia di un paese. Infatti, a seguito di quest'ultimo, i mercati azionari statunitensi hanno subito un crollo che ha richiesto molto tempo prima che il mercato potesse tornare ai livelli di prima. Il crollo si va a ripercuotere su vari livelli, in particolare sul valore della moneta legata all'economia di quel paese. In particolare, se la moneta in questione è il dollaro la ripercussione avviene sicuramente a livello mondiale per quanto riguarda tutti i tassi di cambio, compreso l'EUR/USD, tasso di cambio utilizzato per il trading da parte dell'agente nell'ambito di questa tesi.

### 5.1.5 Considerazioni sulle tecniche utilizzate

L'obiettivo di questa tesi è la valutazione dell'algoritmo di Deep Q-Learning sul Trading Forex e le potenzialità che esso può avere in quest'ambito. Di conseguenza si è deciso di utilizzare l'algoritmo nella sua versione più recente utilizzando tutti gli espedienti necessari a mitigare le problematiche più comuni. Non si è badato però in alcun modo all'integrazione dell'algoritmo con altre tecniche che potrebbero portare enormi vantaggi nei risultati finali. Inoltre non ci si è inoltrati nel tentativo della previsione di possibili avvenimenti descritti in precedenza ed essi costituiscono sicuramente periodi di grande incertezza la cui predizione permetterebbero di evitare eventuali perdite da parte dell'agente. Si discuterà in seguito dunque i possibili sviluppi futuri del progetto anche in quest'ottica, considerando metodologie con cui i risultati potrebbero essere estremamente migliorati.

## 5.2 Metodi e metriche di valutazione

La valutazione degli algoritmi viene fatta in diverse forme. Per prima cosa si effettua una esplorazione dal punto di vista degli iperparametri più significativi e il loro effetto sul set di training, confrontandoli grazie all'utilizzo di grafici indicanti gli andamenti della fase di apprendimento. Allo stesso modo si faranno considerazioni sul comportamento dal punto di vista dei set di validazione e test. Per i motivi visti in precedenza non si ritiene sia strettamente necessaria una valutazione dal punto di vista metrico per quanto riguarda le prime due parti del dataset.

Sono state utilizzate una serie di metriche per la valutazione della bontà delle strategie, soprattutto per quanto riguarda il test set. Di seguito si riporta la descrizione delle metriche utilizzate.

- *Profit*: la metrica più semplice, indica il guadagno cumulativo finale al termine del periodo di training. Esso viene indicato in percentuale alla somma di denaro possibilmente investibile dal trader nel momento dell'apertura della posizione. Il profit

viene calcolato sotto varie forme. Nell'ambito di questa tesi si è rilevato importante calcolare anche valori che prevedono la misura di profitto e perdita media per trade, questo permette di fare considerazioni molto importanti ai fini dell'analisi dei risultati. Viene inoltre calcolato il profit annualizzato, che rappresenta il profitto medio per anno.

- *Profit Factor*: valore che varia tra $[0, +\infty]$ e consiste nel rapporto tra la somma del guadagno e la somma delle perdite. Esso può essere calcolato includendo le spese di commissione nelle perdite che, come si vedrà, incidono ampiamente dal punto di vista delle perdite dell'agente.

$$Profit\_Factor = \frac{\text{Money won}}{\text{Money lost}} \tag{5.1}$$

- *Risk reward ratio* per trade (RRR): misura il rapporto tra quanto si rischia di perdere in un trade rispetto a quanto ci si aspetta di guadagnare e viene calcolato come segue:

$$RRR = \frac{(\frac{\text{Money won on positive trades}}{\text{Winning trades}})}{(\frac{\text{Money lost on negative trades}}{\text{Losing trades}})} \tag{5.2}$$

- *Win/loss ratio*: è una metrica che non prende in considerazione in nessun modo la quantità di denaro guadagnato o perso, ma solamente il numero di trade vinte o perse. Nel caso di Win ratio, essa indica la percentuale di trade vinte rispetto a quelle perse, nel caso di loss ratio il rapporto è contrario.
  Calcolo di Win Ratio:

$$Win\_Ratio = \frac{\text{Winning trades}}{\text{Total trades}} * 100 \tag{5.3}$$

- *Maximum drawdown*: rappresenta la più alta percentuale di perdita calcolata da un picco di guadagno al successivo punto di maggiore perdita. Si può osservare il grafico rappresentante il maximum drawdown nella Figura 5.1 in cui il primo asterisco verde indica il punto di picco massimo antecedente al punto di massima perdita, rappresentato con l'asterisco rosso.
- *Average profit per trade*: indica il profit medio per trade calcolato come il rapporto tra il profit totale e la somma del numero totale di trade.
- Le altre misure utilizzate sono: numero di trade di apertura, perdita e chiusura, profitto totale e medio calcolato in rapporto alle trade vinte, perdita media e totale calcolato rispetto alle trade perse, perdite causate dai costi di commissione.

Risulta opportuno specificare che si è deciso di riportare anche metriche che non sono particolarmente adatte alla valutazione di un algoritmo di reinforcement learning. Alcune delle metriche descritte in precedenza infatti prendono in considerazione la deviazione standard dei return oppure il massimo drawdown. Non risultano particolarmente adatte

poiché l'obiettivo del reinforcement learning è approssimare la strategia che teoricamente permette, nel lungo periodo, di ottenere i migliori risultati per quanto riguarda il reward finale. Questo comporta una possibile alta variabilità dei risultati, ottenendo scarsi risultati sulle metriche che si basano sul rischio comportato dall'utilizzo della strategia.

## 5.3 Risultati della configurazione migliore

In questa sezione si riportano e discutono in modo approfondito le caratteristiche e le scelte inerenti alla configurazione che offre i risultati mediamente migliori sul test set. Si vedranno in seguito anche le differenze prendendo in considerazione la variazione di diversi parametri, l'utilizzo di dataset differenti e il cambiamento della funzione di reward.

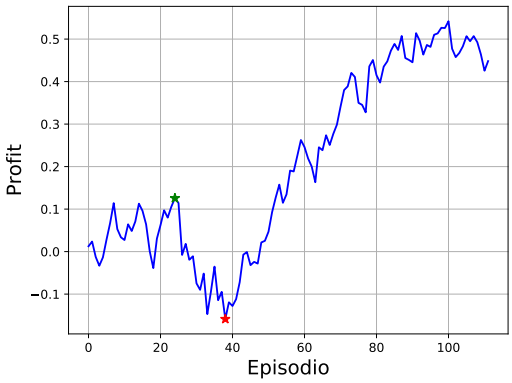



Figura 5.1: Rappresentazione grafica del massimo drawdown

### Dataset

Il dataset che raggiunge i migliori risultati sul test set non è composto da tutti gli indicatori a disposizione. Esso infatti è formato dai semplici dati riguardanti i tassi di cambio e volumi descritti nel paragrafo 4.2.1 e gli indicatori VWAP e Momentum calcolati su archi di tempo di 24, 36 e 48 ore. Gli altri indicatori inseriti e calcolati sugli stessi archi di tempo però riguardano solamente il tasso di cambio EUR/USD. Un migliore rendimento di questa configurazione può essere dato da due principali motivi:

- L'introduzione di troppi indicatori riguardanti tassi che coinvolgono solo indirettamente il tasso di cambio EUR/USD introducono rumore, rendendo molto più complesso per la rete convergere a una soluzione ottimale.
- Si potrebbe cadere in un banale caso di overfitting, in cui l'inserimento degli altri parametri può portare la rete a specializzarsi in maniera troppo marcata sul training set, dando poco spazio a possibilità di generalizzazione.

### Neural network - Function approximator

La struttura della rete utilizzata nell'ambito di questa tesi è una semplice architettura a layer completamente connessi. Lo studio e l'utilizzo di alcune configurazioni differenti

variando la rete in larghezza, i.e. variando il numero di neuroni componenti i layer, e in profondità, i.e. cambiando il numero di layer, ha portato a una struttura relativamente semplice:

- Input layer formato da un numero di neuroni pari al numero di input del dataset configurato in quel momento. Il numero di features utilizzate per la configurazione migliore è 167, quindi l'input layer è formato 167 neuroni.
- Due hidden layer formati da 128 neuroni, funzione di attivazione ReLU e inizializzazione dei pesi uniforme di Xavier Glorot [6].
- Output layer formato da un numero di neuroni pari al numero di azioni che l'agente può compiere, in questo caso 3. La funzione di attivazione utilizzata nell'ultimo layer è quella lineare poiché è opportuno che l'output consista in un valore scalare per ogni azione, che indica il valore della action-value function ($Q(s, a)$) per ogni azione.
- L'ottimizzatore utilizzato è RMSProp con learning rate settato a 0.00005. Non vengono settati altri iperparametri come, ad esempio, il decadimento del valore di learning rate nel tempo, poiché per la regolazione e il decadimento dell'apprendimento sono già fortemente influenzati da altri iperparametri legati strettamente al reinforcement learning. Inoltre RMSProp aggiusta il valore di learning rate nel tempo per ogni parametro.

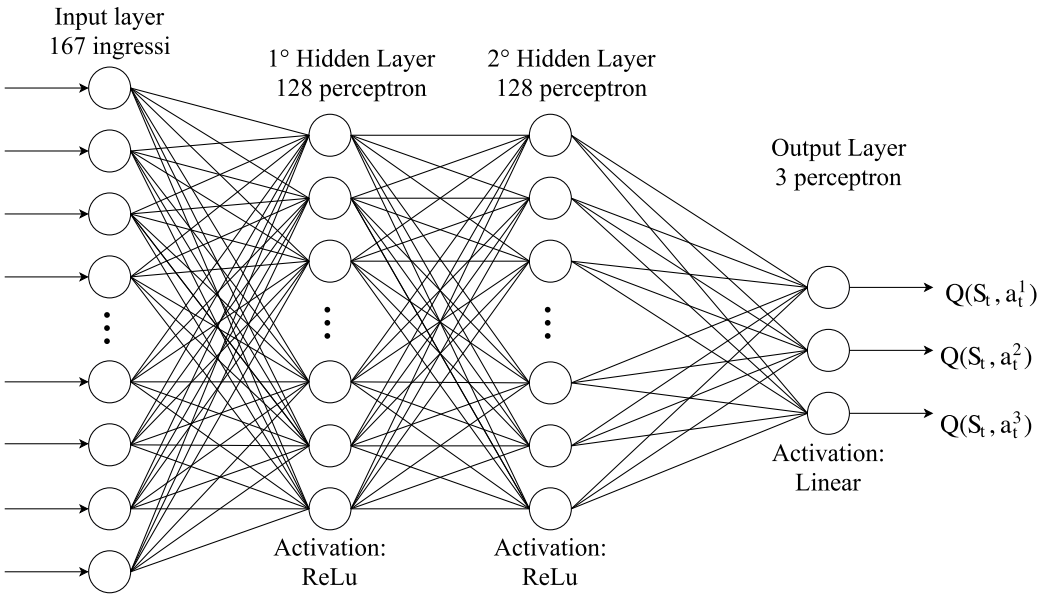



Figura 5.2: Configurazione Neural network

Non è stato possibile esplorare soluzioni tramite l'utilizzo di regolarizzatori come il Dropout e L1, L2 loss per mancanza di tempo e risorse a disposizione.

In fase di sperimentazione si è evidenziato in maniera molto marcata l'incidenza della grandezza della rete. In particolare si evidenzia il fatto che una rete di dimensione inferiore fatica a raggiungere alte prestazioni sul training set, cadendo in un caso di underfitting. In maniera totalmente opposta si verifica l'incidenza di una rete di dimensioni maggiori, ricadendo nel caso di overfitting sul training set e bassa generalizzazione su nuovi dati.

### 5.3.1 Iperparametri del Q-learning

Sono numerosi gli iperparametri legati alla configurazione dell'algoritmo di Q-Learning. Si specificano in particolare quelli relativi alla memoria utilizzata per l'experience replay, alla policy utilizzata e i parametri legati strettamente all'agente.

Si è utilizzata una memoria di grandezza di 5000 tuple $< s_t, a_t, r_t, s_{t+1} >$. Questo significa che l'agente ha la possibilità di allenarsi su esempi che comprendono gli ultimi 200 giorni di trading compiuto e memorizzato.

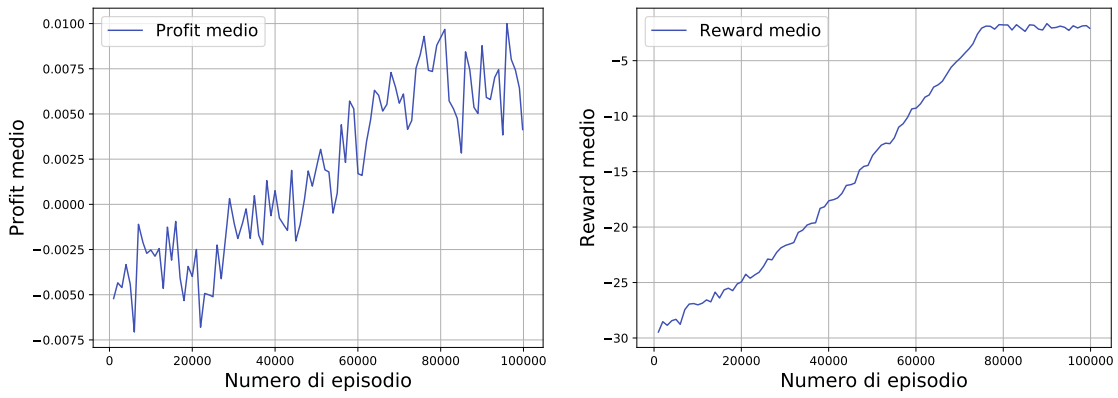
La policy utilizzata è la $\epsilon$-greedy con un valore di partenza di $\epsilon$=1 equivalente al 100% di mosse casuali, fino a un valore di minimo a 0.1, equivalente al 10%. Il momento in cui $\epsilon$ raggiunge il valore minimo corrisponde a quello in cui la rete ha la possibilità di esplorare nuove possibilità di azioni solo per il 10% di probabilità e di rafforzare le strategie che ritiene migliori per il restante 90% delle azioni. Come si osserva in seguito nei risultati sul training set il momento di raggiungimento del minimo valore di $\epsilon$ è nettamente evidente.

Per quanto riguarda i parametri legati in modo diretto all'apprendimento dell'agente e direttamente osservabili nella spiegazione dell'algoritmo di Deep Q-Learning alla Sezione 3.2 si prendono in considerazione:

- Target model update (parametro $C$ nella Sezione 3.2): indica ogni quanti timestep eseguire l'aggiornamento dei pesi della target Q-network. Nella configurazione migliore questo parametro viene settato a 5000 timestep, pari al valore del numero di tuple della memoria.
- Discount factor $\gamma$: valore relativo alla approssimazione del reward aspettato agli stati successivi seguendo la policy ritenuta ottima in un certo momento. Questo valore viene settato a 0.9 anche se, come si vedrà nella sezione dei confronti, sul training set si ottengono profitti ampiamente maggiori utilizzando valori di gamma maggiori.

### 5.3.2 Risultati sul training set

Nella Figura 5.3 è possibile valutare l'andamento della fase di training su cui si possono effettuare alcune considerazioni importanti. I grafici rappresentano il profit e il reward mediati sugli ultimi 1000 episodi. Si ricorda che il profit è un valore a cui, nel caso di questa configurazione, l'agente non ha diretto accesso. Esso infatti ha la possibilità di




(a) (b)

Figura 5.3: Sulla sinistra è possibile notare il profit medio calcolato sugli ultimi 1000 episodi di training. Sulla destra viene rappresentato graficamente l'andamento del reward dello stesso modello, mediato allo stesso modo del profit.

ottimizzare il reward per episodio. La funzione di reward utilizzata nella configurazione con risultati medi migliori è la *Clipped*, come descritta nella Sezione 4.5. Si ricorda che nella fase iniziale dell'allenamento dell'agente le azioni selezionate sono totalmente casuali, non solo a causa del valore di $\epsilon$ settato a 1, ma anche poiché l'agente non ha ancora avuto occasione di esplorare lo spazio degli stati e delle azioni in modo da capire in che modo è possibile guadagnare sul set di training. Il valore di $\epsilon$ cala linearmente fino al raggiungimento del valore minimo, 0.1, all'episodio 75.000, dopo aver visitato 9 milioni di stati e aver compiuto altrettante azioni. Si può notare infatti che le prestazioni dell'agente migliorano linearmente nel tempo sia per quanto riguarda il profit che il reward medio. Al raggiungimento del valore minimo di $\epsilon$ corrisponde il momento in cui, nel reward medio, cala drasticamente il costante miglioramento delle prestazioni. Si può notare come il reward medio rimanga comunque negativo pur avendo raggiunto un 90% di azioni ritenute ottimali da parte dell'agente. Si ritiene che questo sia dovuto alla grande difficoltà di ottimizzazione della funzione di reward utilizzata. Si sottolinea però che l'utilizzo di una funzione su cui è così complesso ottenere buoni risultati si rifletta positivamente in termini di profit. L'obiettivo finale infatti non è quello di ottenere il reward più alto possibile, è molto più importante migliorare il profit finale che rispecchia esattamente quello che l'agente guadagnerebbe nel mondo reale.

Si può notare come l'andamento della funzione di reward sia molto più costante rispetto a quello del profit medio. Questo è dovuto principalmente all'azione di due fattori principali:

- il valore che l'agente ha effettivamente la possibilità di massimizzare è il reward, di

conseguenza è normale che la funzione di profit sia nettamente più variabile.

- la funzione di reward consiste in una discretizzazione del profit non lineare. Essendo la funzione di reward non esattamente corrispondente a quella di profit è normale che l'andamento di una non corrisponda esattamente a quello dell'altra, così come avviene nella Figura 5.3. Il profit infatti risulta altamente più rumoroso del reward medio.

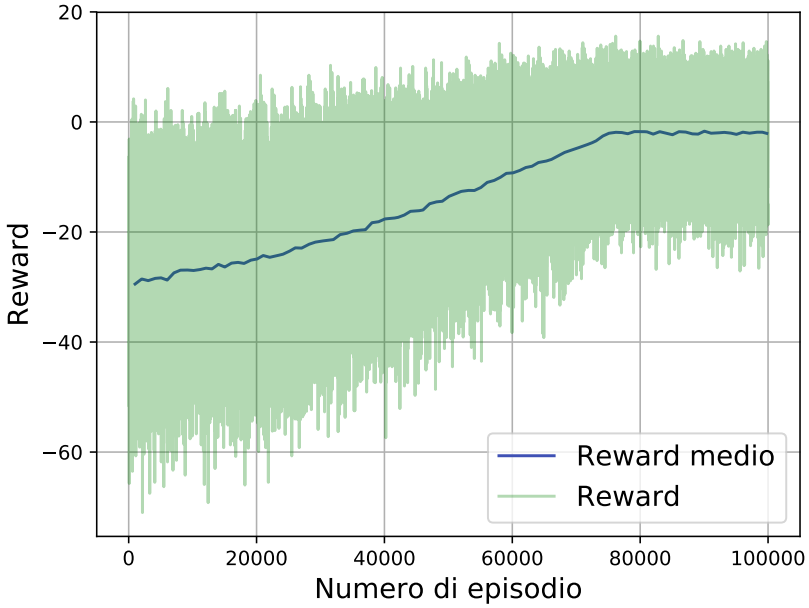
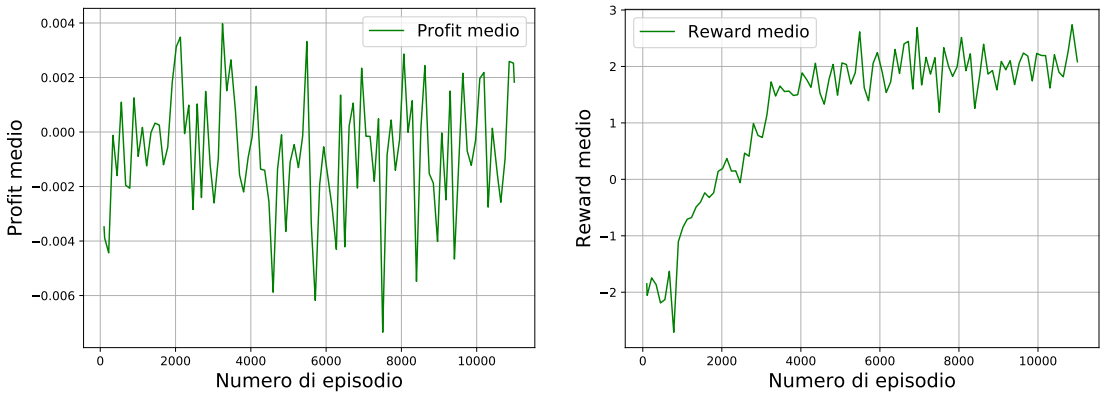



Figura 5.4: Reward medio e reward per episodio - Fase di training

Nella Figura 5.4 è possibile notare che, seppur il reward medio sia costantemente crescente in modo lineare, la stessa cosa non si può dire nel caso in cui si consideri il puro reward per episodio. In questo risultato si evidenzia un grande limite del Deep Q-Learning: l'instabilità dei risultati. Questa instabilità non si verifica nel caso di giochi con uno spazio degli stati abbastanza limitato e semplice, ma è sempre più marcato in caso di giochi complessi. Seppur mediamente vi sia un miglioramento nel tempo, è possibile che siano presenti degli episodi in cui l'algoritmo ottiene scarsissimi risultati. Questo problema può essere sorvolato nel contesto di un gioco a cui successivamente puoi fare una ulteriore partita per provare a battere il punteggio precedente. Questo però non risulta possibile nel contesto in cui l'agente debba essere utilizzato per investire denaro vero e proprio, rischiando di avere performance pessime e perdere in poco tempo moltissimo denaro investito. Si può

notare comunque come la varianza nella distribuzione del reward per episodio si riduca nel tempo, ottenendo comunque risultati con reward sempre meno negativo.

### 5.3.3 Risultati sul validation set




(a) (b)

Figura 5.5: Sulla sinistra è possibile notare il profit medio calcolato sugli ultimi 100 episodi di validation. Sulla destra viene rappresentato graficamente l'andamento del reward dello stesso modello, mediato allo stesso modo del profit.

I risultati sul validation set si discostano nettamente da quelli riguardanti la fase di training. Il principale motivo di questa differenza è dovuto alla percentuale di azioni casuali prese nei due casi. Se nel training set all'inizio dell'allenamento le azioni dell'agente sono totalmente casuali, con la parte di casualità decrescente nel tempo, le azioni sul validation e test set sono sempre quelle che l'agente considera ottime, poiché è necessario per valutare la policy che in quel momento si ritiene migliore e decidere se salvare o no i pesi della neural network che rappresenta quella stessa policy.

Nella maggior parte dei casi la mappatura tra la funzione di reward e la funzione di profit non è molto marcata. In questo caso la corrispondenza fra le due funzioni non è assolutamente netta, infatti la funzione di reward (grafico (b) della Figura 5.5) è in miglioramento netto nel tempo fino a raggiungere il punto migliore attorno all'episodio 5.000 di validation, nel momento in cui $\epsilon$ sul training set si attesta attorno al valore di 0.6, i.e. l'agente compie il 60% delle mosse in modo casuale. Non è necessario quindi arrivare al termine della fase di esplorazione per raggiungere le massime prestazioni per quanto riguarda la funzione di reward sul validation set. Non si può dire la stessa cosa della funzione di profit, che rimane mediamente in positivo fino all'episodio 3.500 e si afferma su un andamento molto instabile, ma mediamente negativo, in seguito. Questo sta a rimarcare quanto non sempre la funzione di reward, per quanto possa essere una

discretizzazione del profit, possa portare a risultati così concretamente simili. Si esplorerà comunque successivamente la casistica in cui la funzione di reward corrisponde esattamente a quella di profit, in modo da capire le motivazioni della scelta di discretizzazione.

Nel caso della validazione risulta molto importante fare un appunto: all'inizio dell'allenamento l'agente non conosce neanche in che modo è possibile ottenere dei reward positivi, la prima parte dell'esplorazione quindi è probabilmente incentrata a capire come ottenere profitti più grandi su singole mosse, più che profitti sul lungo termine. Con l'avanzare del tempo invece l'agente ha la possibilità di elaborare strategie più complesse sul training set e di conseguenza migliorare le proprie prestazioni su di esso.

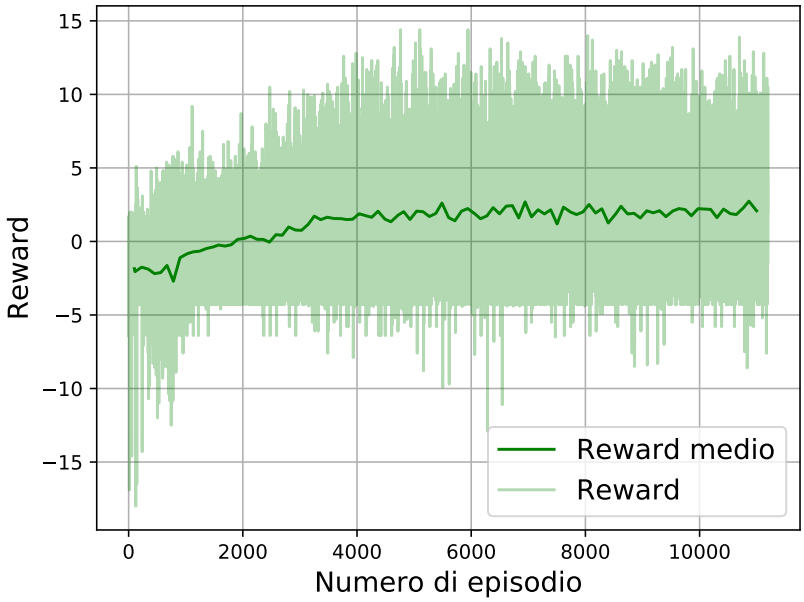



Figura 5.6: Reward medio e reward per episodio - Fase di Validation

Si può notare nettamente nella Figura 5.6 come l'agente utilizzi la prima fase di apprendimento per capire in che modo è possibile guadagnare, per poi portarsi in una fase attorno all'episodio 1.000 (9.000 episodio di training, 1.080.000 step) in cui ha imparato a limitare i danni. Con la funzione di reward è possibile verificare che si attesta attorno a un reward di -4.3, che corrisponde esattamente a 3 aperture e una seguente perdita molto negativa. Se infatti si fa riferimento alla funzione clipped, il reward sarà uguale a $-0.1-0.1-0.1-4=-4.3$. Questo accade nel momento in cui l'agente apre delle posizioni sbagliando totalmente la previsione del trend successivo. In questo caso l'agente apre consecutivamente 2 o 3 posizioni e non le chiude sino al termine dell'episodio, sperando che

il trend dopo una serie di step cambi e gli permetta di trarre profitti invece che perdite. Il fatto che l'andamento medio di guadagno sia comunque in aumento nel tempo significa che la previsione errata avviene sempre meno spesso. Come descritto in precedenza però, la instabilità dei risultati non è stata surclassata utilizzando metodologie standard di Deep Q-Learning.

### 5.3.4 Risultati sul test set

La massimizzazione delle prestazioni sul set di test si è rivelata una impresa estremamente ardua, è infatti molto complesso trovare una configurazione di Deep Q-Learning in grado di raggiungere risultati sempre positivi su un dataset su cui l'agente non ha la possibilità di allenarsi. Seppur si sia riusciti a trovare una configurazione in cui la maggior parte dei modelli trovati hanno profitti sul lungo periodo, non si può dire che i risultati siano totalmente positivi per quanto riguarda le metriche più importanti utilizzate generalmente all'interno del mondo del Forex. Spesso queste ultime infatti prendono in considerazione una serie di fattori, tra cui il massimo drawdown e la deviazione standard dei profitti, che non si può dire siano estremamente positive nei risultati ottenuti. Come specificato infatti nella Sezione 5.3.2 il Deep Q-Learning ha il limite di non riuscire a trovare una impostazione che permette di avere risultati stupefacenti in ogni episodio. Questo problema all'interno del mondo del Forex non si verifica solamente nel caso del reinforcement learning ma anche nell'utilizzo di altre metodologie basate su Supervised Learning. Oltre all'estrema difficoltà del problema sopraggiungono anche quei momenti in cui vi sono eventi inaspettati come descritti in precedenza. Non è stato preso nessun provvedimento per evitare che l'agente compia azioni durante periodi di completa imprevedibilità del mercato ed essi si possono notare all'interno dei risultati ottenuti che si vedranno in seguito.

I 5 test visibili nella Figura 5.7 rappresentano modelli salvati nei 5 momenti più performanti sul validation set tra l'episodio 1000 e 5000 di validazione. E' stato necessario racchiudere il momento di salvataggio dei modelli in questo intervallo poiché si è notato, in modo totalmente empirico, che modelli salvati o prima o dopo l'arco di tempo citato sono più propensi a rappresentare momenti di fortuna all'interno del validation set.

Su 5 test effettuati, in questo caso, risultano tutti terminare con risultati positivi in termini di profitto finale alla fine delle 112 settimane di test. Non sempre questo accade, anche se mediamente i risultati sui modelli di test salvati portano a 7 modelli positivi su 10. Di seguito si riportano i risultati delle metriche che si è deciso di utilizzare sul test set e su cui è possibile effettuare numerose osservazioni altamente significative.

Dalle prime due tabelle (5.1, 5.2) si possono estrapolare molte informazioni interessanti. Per prima cosa si analizzano le metriche che utilizzano solamente il numero di trade. Si nota che la Win Ratio si attesta sempre tra il 41 e il 44% mentre la loss ratio si attesta tra il 3,6 e il 4,3%. Questo sta a significare che il se si analizza solamente il numero di trades,

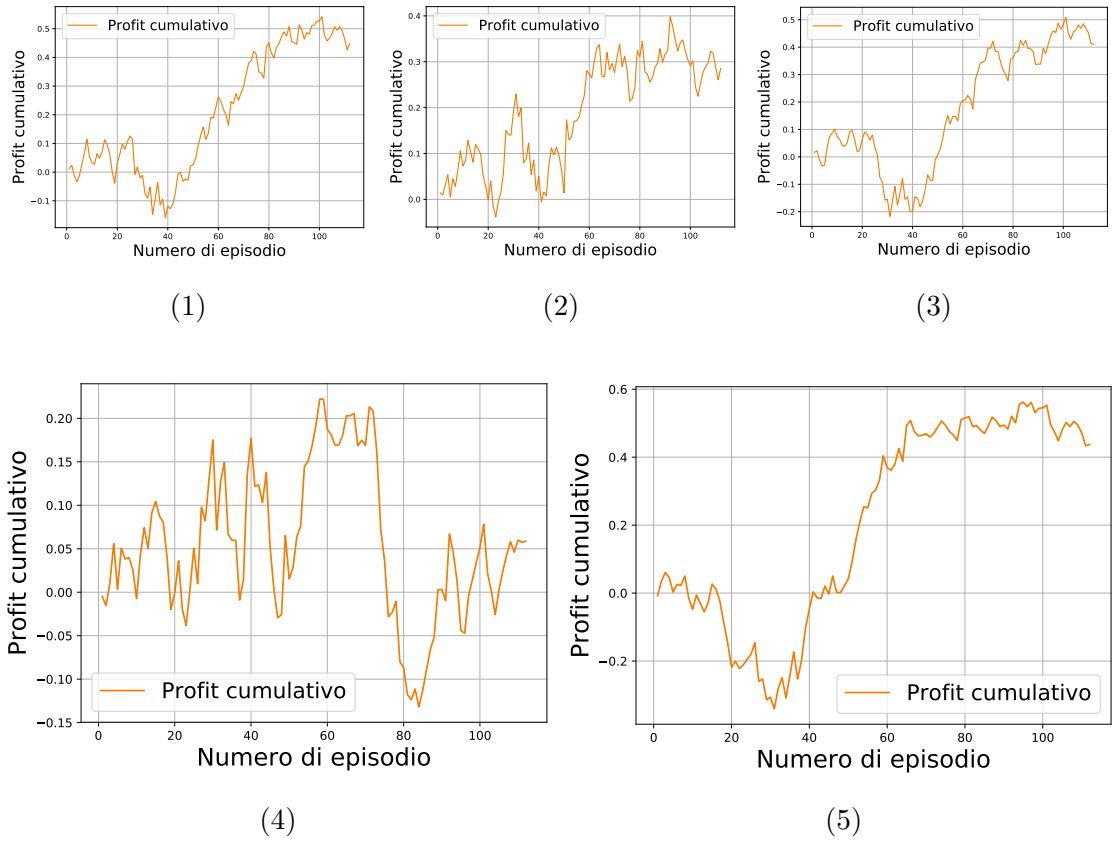



Figura 5.7: Rappresentazione dei profitti cumulativi in percentuale rispetto alla somma investita per trade dei 5 modelli migliori sul validation in un arco temporale definito.

Tabella 5.1: Risultati sui test (1)

| | Win/Lose/Open ratio (%) | Profit factor con/senza commissioni | Money won | Trades won | Avg won/trade |
|---|---|---|---|---|---|
| 1 | 43,822 - 4,036 - 52,143 | 1,345 - 1,184 | 2,88905 | 1.759 | 0,164% |
| 2 | 41,952 - 4,295 - 53,753 | 1,206 - 1,117 | 2,73114 | 1.006 | 0,271% |
| 3 | 43,389 - 4,0 - 52,611 | 1,289 - 1,158 | 3,01534 | 1.562 | 0,193% |
| 4 | 41,655 - 4,184 - 54,161 | 1,098 - 1,024 | 2,55605 | 926 | 0,276% |
| 5 | 43,829 - 3,633 - 52,537 | 1,306 - 1,174 | 2,95075 | 1.520 | 0,194% |

sono ampiamente più numerose quelle in cui l'agente riesce a vincere, in confronto a quelle in cui l'agente perde. E' possibile notare quanto detto anche analizzando direttamente al numero di trade vinte rispetto al numero di trade perse. Il rapporto tra questi due numeri

Tabella 5.2: Risultati sui test (2)

| | Money lost | Trades lost | Avg lost/trade | Money lost commissions | Trades commmissions |
|---|---|---|---|---|---|
| 1 | 2,14787 | 162 | 1,326% | 0,29302 | 2.093 |
| 2 | 2,26512 | 103 | 2,199% | 0,18046 | 1.289 |
| 3 | 2,33982 | 144 | 1,625% | 0,26516 | 1.894 |
| 4 | 2,32878 | 93 | 2,504% | 0,16856 | 1.204 |
| 5 | 2,25863 | 126 | 1,793% | 0,25508 | 1.822 |

si mantiene costante anche nel caso in cui vi sia un numero di trade di apertura inferiore, come si può notare dal caso del test numero 4.

E' inoltre necessario indicare che spesso il numero totale di trade di apertura non coincide con il numero di trade di vittoria sommato a quelle di sconfitta. Questo accade per un semplice motivo: se l'agente è in esposizione maggiore di 1 (in long o in short) nel momento in cui l'episodio termina, la chiusura di più posizioni al termine dell'episodio viene contata come una singola vittoria o perdita, mentre il numero di aperture corrispondente equivale a 2 o 3. Quindi:

$$Trades\_won + Trades\_lost \leq Trades\_commissions$$

Sfortunatamente pur essendo molto più alto il numero di trade positive rispetto a quelle negative, esistono due dati che ci fanno riflettere sul perché non sia nettamente positivo anche il profit in relazione al numero di vittorie rispetto alle sconfitte. Questi dati sono la media di profitto in caso di trade vinta e la media di perdita nel caso contrario. Si nota infatti che la media di profitto per trade positiva di attesta attorno allo 0.002%, mentre la media in caso di perdita si attesta tra lo 0.013 e lo 0.025%, un ordine di grandezza superiore. Questo porta a riflettere sul fatto che un anche se vi è un grande numero di trade vinte esse portano comunque un bassissimo profitto se prese singolarmente, rispetto a una singola trade in perdita. Quanto appena detto non è affatto una sorpresa se si pensa all'analisi effettuata sul set di validazione, dal momento in cui spesso l'agente si intestardisce a tener aperta la posizione sino al termine dell'episodio nel caso in cui vi sia stato un errore iniziale di apertura della posizione in senso inverso a quanto aspettato, sperando che il tasso di cambio inverta la rotta in modo da poter limitare i danni.

Si mostra nella Figura 5.8 la situazione descritta in precedenza. Guardando attentamente le azioni intraprese dall'agente, esso già dai primi step dell'episodio, dopo aver preso 4 reward molto piccoli ma positivi, apre consecutivamente 3 posizioni in long, sbagliando così la previsione del trend successivo in cui, invece di aumentare il valore dell'euro rispetto al dollaro, vede una crescita di valore del secondo dei due citati. Così l'agente, per evitare perdite ulteriori, non chiude la posizione sino al termine dell'episodio, compiendo una serie

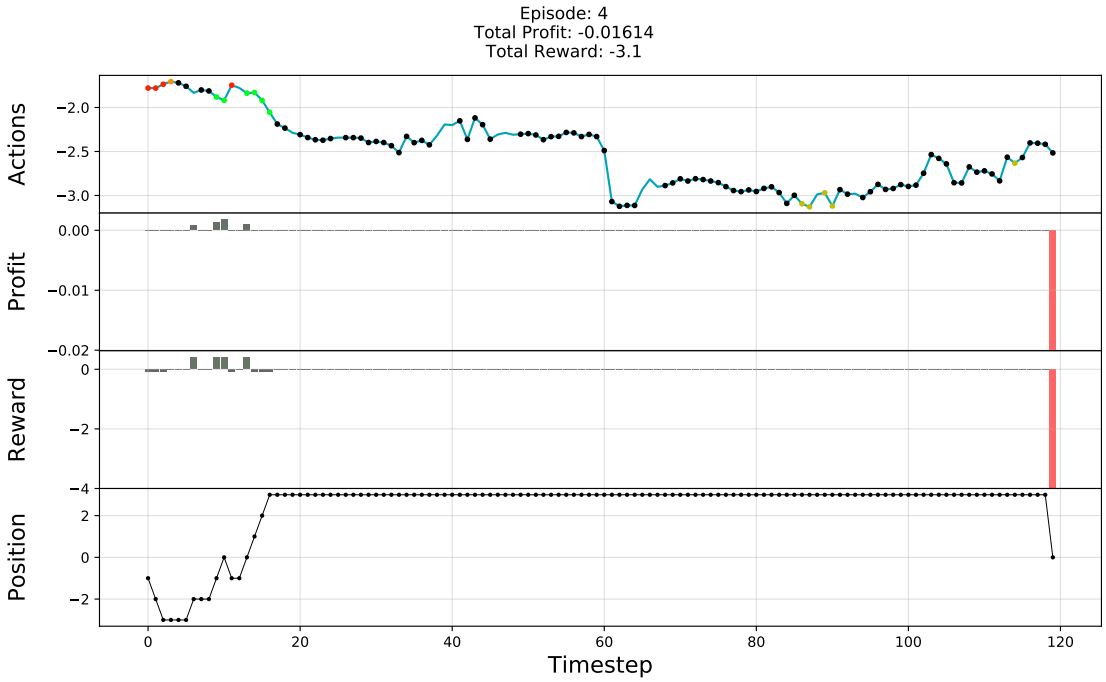



Figura 5.8: Esempio di episodio di trading con agente bloccato

di azioni flat e long che non vengono contate come crescita d'esposizione poiché è già stata raggiunta la massima esposizione possibile. Al termine dell'episodio gli vengono chiuse le posizioni con conseguente grave perdita.

Dalla seconda tabella è possibile notare, sotto la voce "Money lost commissions" che le spese di commissione per apertura di posizioni incidono molto sul risultato finale. Infatti esse sì attestano sempre tra il 10 e il 15% della perdita totale finale calcolata sulle trade perse e quelle di commissione.

Tabella 5.3: Risultati sui test (3)

| | Average profit/trade | Risk reward ratio | Max drawdown | Profit | Annual profit |
|---|---|---|---|---|---|
| 1 | 0,011% | 0,124 | 0,318% | 44,8% | 22,41% |
| 2 | 0,012% | 0,123 | 0,24% | 28,6% | 14,28% |
| 3 | 0,011% | 0,119 | 0,349% | 41% | 20,52% |
| 4 | 0,003% | 0,11 | 0,361% | 5,9% | 2,94% |
| 5 | 0,013% | 0,108 | 0,406% | 43,7% | 21,85% |

Tenendo conto dei risultati osservati si derivano dunque le metriche riportate nella tabella 5.3.

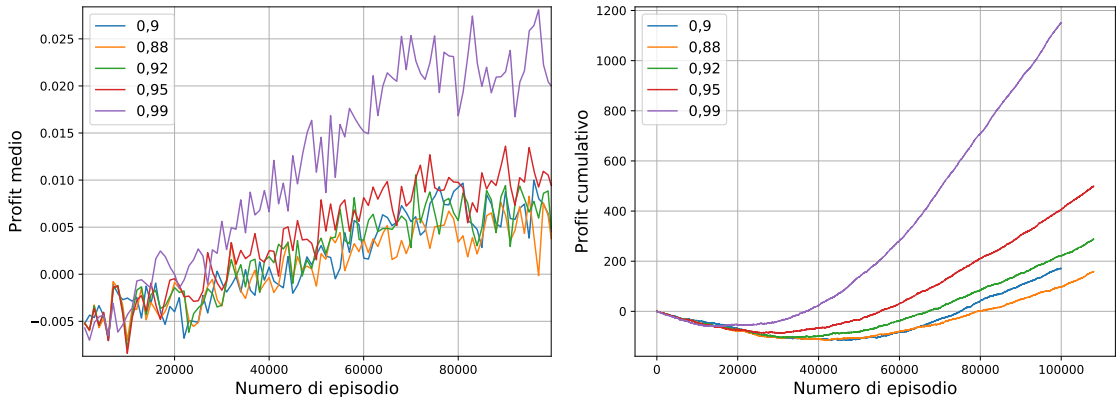
- *Average profit per trade*: il profitto medio risulta ovviamente positivo nel caso in cui il profit finale sia positivo. I modelli più performanti hanno risultati che si attestano attorno al guadagno di 1,2 pips per trade.
- *Risk reward ratio*: dato che essa si basa sul rapporto della media di profitto per trade vinta e la media di perdita per trade persa, questo valore ha una connotazione positiva nel caso in cui sia maggiore di 1, mentre quando è inferiore di 1 indica che a fronte di un rischio di una unità di esposizione si ha rischio di perdita maggiore in caso di trade persa rispetto al guadagno in caso di trade vinta. Questo valore quindi, nel caso dei modelli trovati con questa metodologia, non risulta mai positivo per lo stesso motivo descritto in precedenza. L'agente è portato a guadagnare vincendo tante trade a basso profitto piuttosto che vincendone poche ad alto profitto. In relazione alla metrica Risk Reward Ratio quindi si evince che una singola azione di esposizione porta con se il rischio di una perdita ingente.
- *Maximum drawdown*: questa metrica rappresenta una grandissima nota dolente per i modelli migliori trovati con l'algoritmo di Deep Q-Learning. Esso è un dato estremamente importante poiché indica la massima quantità di perdita in percentuale a cui si può andare incontro in futuro. Come descritto nella Sezione 5.1.4 vi sono una serie di eventi che potrebbero caratterizzare l'impredicibilità del mercato. Senza dubbio questa tipologia di eventi va a intaccare la strategia dal punto di vista del drawdown, poiché un evento che porta instabilità per un certo periodo di tempo potrebbe destabilizzare i risultati dell'algoritmo anche per un arco di alcune settimane. Questo si nota, ad esempio, nei modelli della Figura 5.7, dove spesso in corrispondenza della ventiquattresima settimana si può notare un ampio drawdown. Il periodo corrispondente alle settimane tra la 15 e la 30 è quello in cui si è verificato una grandissima svalutazione dell'euro nei confronti del Dollaro americano e si è verificato l'evento descritto nella Sezione 5.1.3. I massimi drawdown verificati si attestano tra il 25 e il 40% e sono estremamente alti per far sì che sia possibile utilizzare i modelli trovati in un contesto reale. Si discuterà di possibili soluzioni riguardo il drawdown nel capitolo finale.

## 5.4 Confronto con altri risultati

Si analizzano ora alcune variazioni della configurazione principale che permettono di effettuare osservazioni sulla modalità di apprendimento dell'agente. Queste ultime sono effettuate principalmente sui set di allenamento e di validazione, essendo le parti di dataset su cui è possibile riscontrare in modo più approfondito i cambiamenti effettuati.

### 5.4.1 Variazione del valore di gamma

Come prima modifica degli iperparametri si tratta l'aumento e la diminuzione del valore di $\gamma$. Questo è un valore estremamente importante e legato direttamente all'algoritmo di Deep Q-Learning che indica quanta importanza viene data alle mosse più distanti dal punto di vista temporale nel futuro per capire qual'è il ritorno aspettato seguendo la policy ritenuta ottima in quel momento. Questo valore incide ampiamente sulle azioni che esso intraprende e sul successivo apprendimento. Si è deciso di variare gamma in 5 valori: 0,88, 0,9 (come la configurazione con migliore generalizzazione), 0,92, 0,95, 0,99 e i risultati sul set di training sono estremamente interessanti.




(a) (b)

Figura 5.9: Rappresentazione del profit medio e cumulativo con la variazione di 5 valori di $\gamma$ crescenti

Come rappresentato nel grafico (a) della Figura 5.9 si nota che l'apprendimento e le prestazioni dell'agente migliorano molto più facilmente all'aumentare del valore di $\gamma$. Si osserva in particolar modo la differenza tra il valore finale di profit medio nel caso di $\gamma = 0{,}99$ e gli altri casi via via decrescenti, il primo consegue risultati nettamente superiori. Offrire la possibilità all'agente di dare uno sguardo più avanti nel futuro offre a esso un notevole vantaggio che porta i suoi frutti in particolar modo sui dati su cui l'agente ha la possibilità di allenarsi. Allo stesso modo si può notare l'andamento del profit cumulativo durante l'allenamento dell'agente (grafico (b) Figura 5.9) dove i profit cumulativi sono via via maggiori al crescere di $\gamma$.

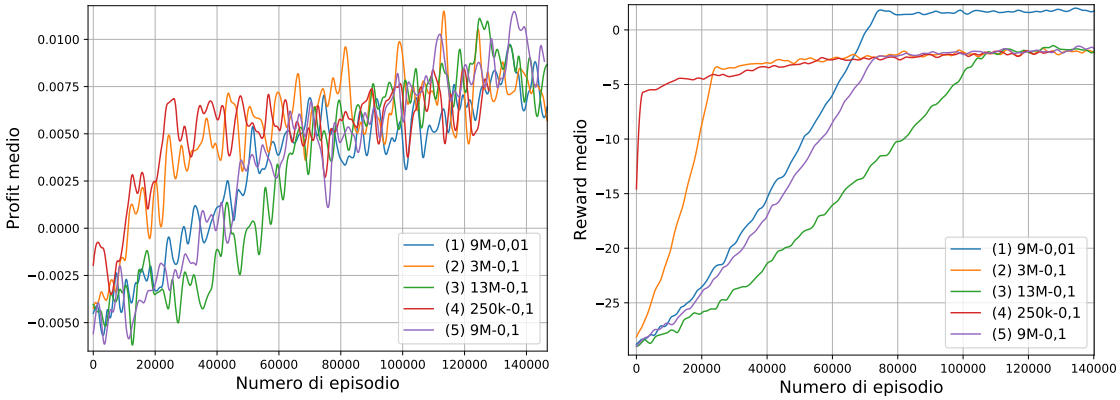
Gli stessi vantaggi non si possono osservare sul set di validazione, dove un valore di gamma troppo elevato porta a una scarsa generalizzazione e risultati per nulla soddisfacenti. Lo stesso comportamento si è notato nei test effettuati con valore di $\gamma$ più elevato di 0,9. Per questo motivo si è ritenuto che quest'ultimo fosse il migliore compromesso per

ottenere la massima generalizzazione possibile.

### 5.4.2 Variazione della policy $\epsilon$-greedy

Nella presente sezione si analizza la variazione di $\epsilon$, il fattore di esplorazione incluso nella policy $\epsilon$-greedy che stabilisce per quanto tempo e in che quantità l'agente deve esplorare lo spazio degli stati per essere certo che esso riesca a valutare al meglio tutte le possibili strategie. Questo valore viene cambiato in diversi modi:

- Cambiamento del numero di passi di decadimento del valore di $\epsilon$ da un massimo di 1 al minimo di 0,1.
- Modifica del valore di minimo da 0,1 a 0,01.




(a) (b)

Figura 5.10: Rappresentazione del profit e del reward medio variando la configurazione della policy $\epsilon$-greedy

I cambiamenti più significativi si possono valutare osservando attentamente la funzione di reward medio del grafico (b) nella Figura 5.10. Le curve contrassegnate con la label (2), (3), (4), (5) rappresentano una policy $\epsilon$-greedy che varia da un massimo valore di 1 a un minimo di 0,1 in rispettivamente 3 milioni, 13 milioni, 250.000 e 9 milioni di step. La policy che ha meno step per effettuare una approfondita esplorazione dello spazio degli stati, dopo 250.000 step raggiunge un reward medio di circa -6, inferiore rispetto al reward medio di -4 e -3 e -3 corrispondenti al valore minimo di epsilon negli altri tre casi. In particolare guardando la corrispondenza del reward medio nei casi (2) e (4) all'episodio 22.000 corrispondente al raggiungimento del minimo valore di epsilon del secondo modello, possiamo notare come una esplorazione maggiore abbia influito positivamente sull'apprendimento dell'agente. Inoltre è possibile osservare che, andando avanti nel tempo, il fattore

di esplorazione al 10% delle azioni porta comunque i 4 modelli sopracitati a raggiungere prestazioni molto simili al termine dell'apprendimento.

Un'altra variazione è quella del modello con label (1) in cui il valore minimo di $\epsilon$ viene portato a 0,01 dopo 9 milioni di step (episodio 75.000). In questo caso ovviamente, considerando che le azioni compiute da parte dell'agente sono per il 99% appartenenti alla policy ritenuta ottima, è normale che le prestazioni di reward medio siano nettamente superiori superando addirittura lo 0. Quello che cambia però è che al raggiungimento del minimo il fattore esplorazione inizia a incidere veramente in maniera esigua, permettendo all'algoritmo di migliorare pochissimo da quel momento in poi.

### 5.4.3 Variazione della funzione di reward

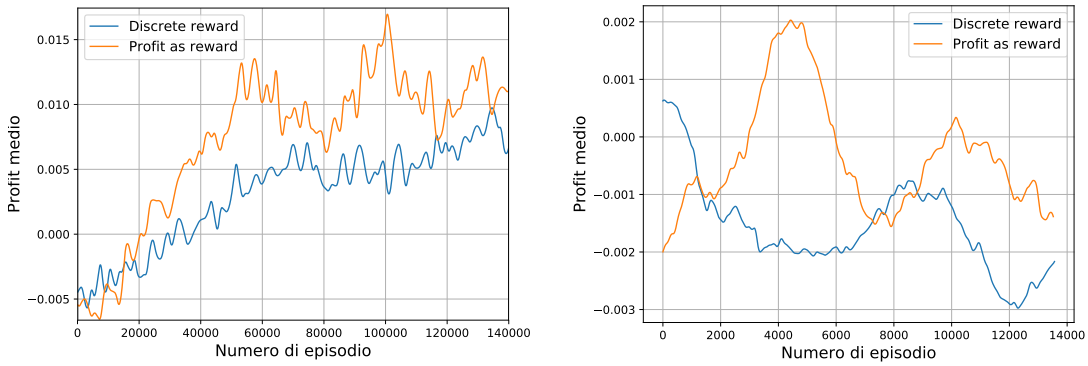
La funzione di reward è sicuramente uno degli aspetti più importanti e va a impattare direttamente sulla strategia dell'agente. Si è tentato, per tutto il primo periodo di sviluppo della tesi, di migliorare la generalizzazione portata dall'utilizzo di una funzione di reward che corrispondesse esattamente al profit, come descritto nella Sezione 4.5, senza avere particolari successi. Per questo motivo si è giunti alla configurazione finale che prevede l'utilizzo della funzione di reward discretizzata.

Sul set di training, come osservabile dal grafico (a) della Figura 5.11, la funzione che descrive l'apprendimento utilizzando il profit come funzione di reward risulta leggermente migliore in termine di profit medio, descrivendo un apprendimento più veloce e con profit medio finale maggiore. Questo è normale poiché la funzione da massimizzare è direttamente quella di profit e visibile sotto forma di reward da parte dell'agente. Allo stesso tempo però si evidenzia una più forte instabilità dell'algoritmo che va a riflettersi anche sui test finali. Si può notare dal grafico (b) un maggiore picco di profit medio nel caso di utilizzo del profit come reward e l'andamento è considerevolmente più consono a quella che dovrebbe essere la fase di validation.

### 5.4.4 Variazione della grandezza della rete

Le reti neurali sono estremamente interessanti sotto innumerevoli punti di vista. E' noto che la variazione in profondità e larghezza cambia decisamente le caratteristiche della funzione possibilmente approssimabile durante la fase di allenamento. Risulta inoltre interessante notare quanto la variazione della rete porti a casi di underfitting o overfitting sul training set.

Facendo riferimento alla Figura 5.12 vengono rappresentati i risultati sul training set di quattro modelli totalmente identici se non per la grandezza della rete che funge da function approximator. Ogni modello ha in comune un primo livello di input a 167 ingressi pari al numero di input feature e un output layer formato da 3 neuroni che caratterizzano il valore delle azioni. Il primo modello è caratterizzato da una rete a 3 hidden layer formati da una configurazione piramidale del numero di perceptron, in particolare: 256, 128, 64




(a) (b)

Figura 5.11: Rappresentazione del profit medio in caso di modifica della funzione di reward. La figura (a) rappresenta il profit medio nella fase di training, mentre la figura (b) rappresenta la fase di validation

perceptron. La seconda rete è costituita da 5 hidden layer con rispettivamente 256, 256, 128, 64, 32 neuroni. La rete caratterizzante il terzo modello è formata invece da 2 hidden layer composti ciascuno da 32 perceptron e infine la quarta neural network è formata da due hidden layer da 128 perceptron ed è quella tipicamente utilizzata come configurazione migliore.

Se si osservano gli andamenti dell'apprendimento delle quattro funzioni è possibile ricavare una informazione importante: sia la funzione di profit che la funzione di reward raggiungono prestazioni nettamente superiori nel caso in cui la rete utilizzata abbia dimensioni maggiori. Di conseguenza è possibile inferire che la grandezza della rete permetta all'agente di raggiungere una approssimazione migliore dei dati di training e ottenere maggiori profitti su di essi. Si possono notare performance nettamente migliori man mano che si aggiungono layer e perceptron alla rete.

La figura (c) rappresenta il profit medio durante la fase di validazione. Com'è intuibile dalla differenza fra le figure (b) e (c), anche se la rete di maggiori dimensioni ha ottenuto più grandi profitti nella fase di training, lo stesso non si può dire per quanto riguarda la fase di validazione. Le reti di dimensioni maggiori tendono a raggiungere l'apice delle performance sul validation molto presto. In questo caso si nota che addirittura la rete di dimensioni inferiori con solo due hidden layer a 32 neuroni ha performance migliori. Ci si ritrova dunque nel caso di overfitting in cui la rete ha ottimi risultati sul training set che non si riflettono in seguito sui dati su cui la rete non si allena.

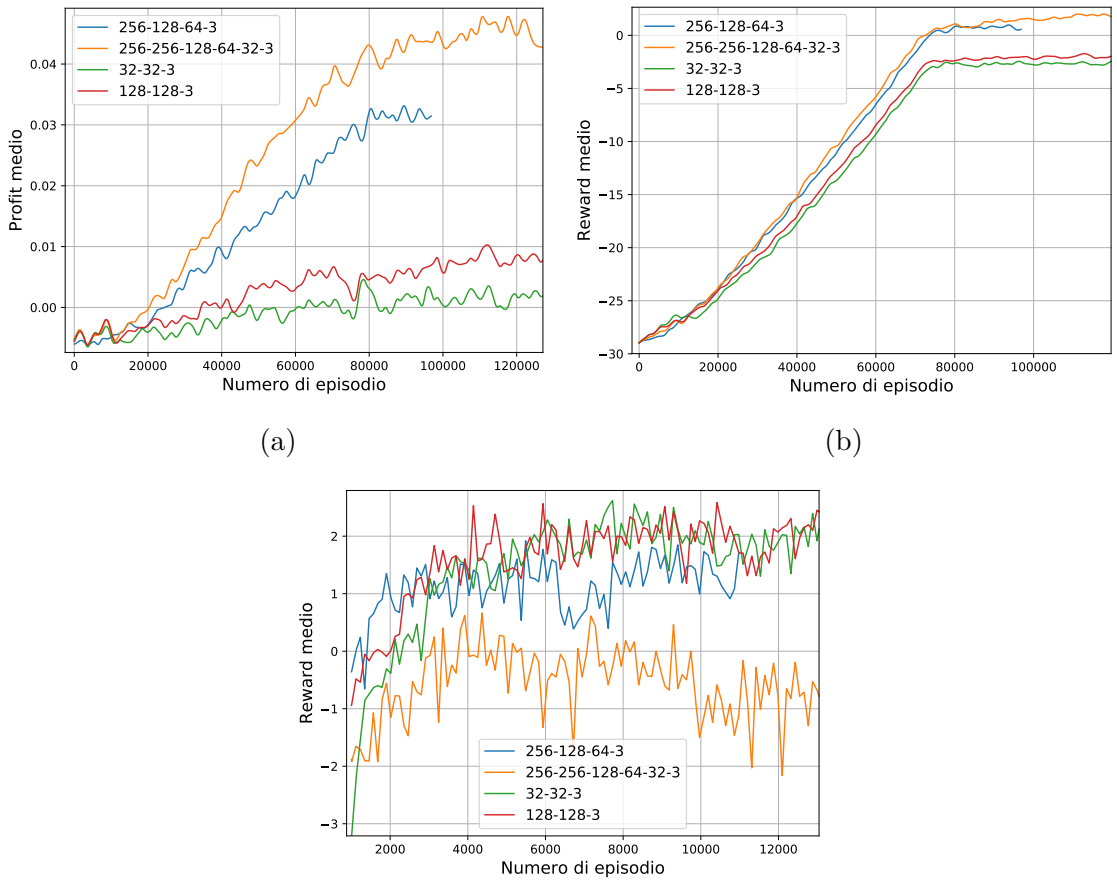



Figura 5.12: Rappresentazione, al variare della rete, di: (a) Profit medio sul training set, (b) reward medio sul training set, (c) reward medio sul validation set

### 5.4.5 Variazione del dataset utilizzato

Spesso all'interno del machine learning e in particolar modo nel Deep Learning non si dà una grandissima importanza alla composizione del dataset, anzi, spesso si potrebbe pensare che un dataset contenente tutte le features possibili possa ottenere di per certo risultati migliori. Per la prima parte dello sviluppo della tesi ci si è soffermati sulla creazione di un dataset che avesse il maggior contenuto informativo possibile per la neural network, in modo che fosse la rete stessa a capire quali sono le features migliori da utilizzare per l'apprendimento. Le Deep neural networks hanno infatti la caratteristica di far lentamente decadere il peso delle features che impattano meno sui risultati della loss function. Questa gestione automatica delle features permette al programmatore di compiere uno sforzo

nettamente inferiore nella fase di preprocessing dei dati di input. Nella fase in cui è necessario però ottimizzare i risultati della rete, ci si rende conto che è molto d'aiuto agire direttamente sulle features in input, aiutando la rete a estrarre i contenuti più informativi prima che inizi la vera e propria fase di training.

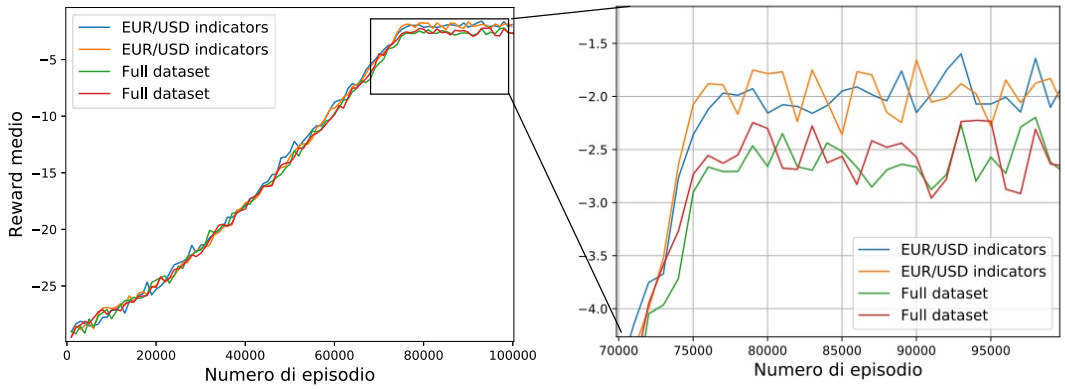



Figura 5.13: Rappresentazione del reward medio in caso di variazione del dataset

La Figura 5.13 è estremamente esplicativa. Gli andamenti delle 4 funzioni rappresentate prevedono l'utilizzo di due tipologie di dataset e stessa identica configurazione del modello di Deep Q-Learning. Il primo dataset contiene gli indicatori calcolati sulle features a disposizione di tutti i tassi di cambio e volumi, su più orizzonti temporali. Il secondo dataset contiene tutte le features di tassi di cambio e volumi, ma gli indicatori sono calcolati solamente sul tasso di cambio EUR/USD, cioè quello su cui è possibile effettuare gli investimenti. Come si può notare dalla parte sinistra della figura l'andamento e l'apprendimento sul training set è molto simile nei due casi, fino al raggiungimento del 75.000-esimo episodio, momento in cui viene raggiunto il minimo valore di $\epsilon$. In questo punto si nota la differenza: i due modelli che utilizzano un dataset contenente un numero inferiore di input features riescono a raggiungere risultati migliori sul dataset di train. In questo caso il comportamento osservato si riflette anche sul dataset di validation e di test.

Da queste osservazioni si può fare una considerazione importante: il motivo di questo miglioramento può essere dovuto principalmente all'introduzione di rumore nel caso in cui si utilizzino un numero maggiore di features, portando una maggiore difficoltà per la rete a imparare e generalizzare.

# Capitolo 6

# Conclusioni e sviluppi futuri

Nel presente capitolo si discutono i risultati ottenuti in base agli obiettivi stabiliti nell'introduzione in termine di vantaggi e svantaggi apportati dalle tecniche di Deep Q-Learning all'interno del Forex trading. Si effettua in seguito una analisi delle possibili tecniche di risoluzione dei problemi affrontati nell'ambito di questa tesi e i possibili sviluppi futuri in ambito aziendale.

## 6.1 Conclusioni

Durante lo sviluppo di questo progetto di tesi si è analizzata l'efficacia delle tecniche di Deep Q-Learning nell'ambito del forex trading, ambiente noto per la sua incredibile imprevedibilità. Con la nascita di quest'idea in un contesto aziendale, si è trovata la necessità di strutturare il problema in modo differente dalle usuali tecniche di Reinforcement learning, in modo che fosse possibile stabilire la reale efficacia di queste ultime nel mondo reale per capire se fossero possibilmente utilizzabili per l'investimento di soldi veri e propri. Si vedano di seguito gli obiettivi raggiunti durante lo sviluppo di questo progetto:

- Si è studiato e approfondito il problema di Reinforcement learning, con particolare attenzione alle tecniche di Deep Q-Learning, strutturando e applicando lo stesso in ambito di Forex Trading in contesto aziendale.
- Si è sviluppato un progetto per la risoluzione del problema del Forex con tecniche di Deep Reinforcement Learning tramite l'impiego di alcune librerie e il linguaggio Python. Si è utilizzata parte della piattaforma aziendale per la costruzione automatica di modelli di Deep Learning tramite un file di configurazione. Quest'ultimo è stato esteso in modo da creare modelli di Reinforcement learning senza apportare modifiche dirette a livello di codice. Sono stati implementati modelli di simulazione dell'ambiente del Forex in modo da monitorare le varie fasi di apprendimento nelle fasi di training, validation e test sotto un punto di vista grafico e con salvataggio automatico dei checkpoint dei risultati ottenuti.

- E' stata effettuata un ampia trattazione dei risultati tramite l'utilizzo del software utilizzato durante tutte le fasi di apprendimento dell'agente di Deep Q-Learning, prestando particolare attenzione alle varie caratteristiche delle metodologie di Deep Reinforcement learning e sottolineando la grandissima instabilità del mercato del Forex. Sono state utilizzate metriche di valutazione per misurare in modo quantitativo i risultati raggiunti e sono stati analizzati, da un punto di vista maggiormente qualitativo, i grafici rappresentanti le curve d'apprendimento dell'agente nel corso del tempo.

Le aspettative iniziali erano alte e le speranze di ottenere buoni risultati non mancavano. Questo progetto però risulta solo l'inizio dell'esplorazione di queste nuove metodologie. Le tecniche di Deep reinforcement learning sono svariate e anche dal punto di vista del Deep Q-Learning si hanno ampi margini di miglioramento, soprattutto utilizzando una serie di miglioramenti che saranno in seguito sottolineati nella sezione dedicata agli sviluppi futuri.

## 6.2 Sviluppi futuri

Sono numerose le idee che potrebbero portare al miglioramento della metodologia sviluppata fino a ora. Si offre quindi uno sguardo a quali sono stati i principali problemi riguardanti i risultati finali e quali tecniche potrebbero essere utilizzate al fine di portare il modello a una maggiore stabilità dei risultati e a mitigare le principali problematiche incontrate.

### Problemi di instabilità di risultati e alto drawdown

I maggiori problemi riscontrati sono principalmente dovuti a una grande instabilità dei risultati e alle corrispondenze tra l'apprendimento sul training set e la validazione sul validation set. Nel corso dello sviluppo della tesi con un accurata ricerca degli iperparametri migliori si sono riusciti a migliorare sia i risultati sul test set che il comportamento nel validation. Un altro problema riscontrato dovuto all'incertezza del mercato del Forex e in buona parte anche all'instabilità dei risultati dell'algoritmo di Deep Q-Learning è sicuramente quello dell'alto drawdown. Molte volte infatti dopo svariati episodi di allenamento l'agente riesce a ottenere risultati molto buoni su alcuni episodi, ma vi è la possibilità che non riesca a gestire particolari situazioni che si verificano in molti episodi consecutivi. Questo porta a un conseguente aumento del drawdown e una successiva perdita netta in un particolare periodo di tempo.

### Esplorazione iperparametri e dataset

Molti dei problemi incontrati nel tempo sono stati parzialmente superati con una buona esplorazione dello spazio degli iperparametri della neural network costituente la funzione di approssimazione e l'algoritmo di Deep Q-Learning. Tuttavia si ritiene che ci sia ancora del lavoro da fare per quanto riguarda sia lo spazio degli iperparametri dedicato al reinforcement learning, ma soprattutto quello riguardante la Deep neural network. La sperimentazione nell'ambito di questa tesi è stata fatta tenendo in considerazione solamente una tipologia di ANN, la tipologia completamente connessa. In questo caso l'esplorazione si è basata solamente sulla grandezza in profondità e larghezza della rete e in maniera ridotta sull'ottimizzatore utilizzato. Ci sarebbero invece molte altre accortezze da provare a mettere in gioco, come, ad esempio:

- Maggiore esplorazione degli ottimizzatori e dei loro iperparametri.
- Regolarizzatori sui pesi della rete e sulla funzione di loss.
- Anche se generalmente le funzioni di attivazioni ReLu sono le più utilizzate, potrebbe comunque risultare proficuo tentare l'utilizzo di differenti funzioni di attivazione oppure le semplici variazioni della ReLu.

Oltre alla esplorazione degli iperparametri potrebbe essere opportuno un più profondo studio del dataset e dei dati offerti in input all'agente. Seppur sotto questo punto di vista vi sia stato un ampio miglioramento dalle fasi iniziali del progetto, gli indicatori utilizzati sono ridotti rispetto a quelli disponibili in campo azionario e l'inserimento di alcuni indicatori rimuovendone altri potrebbe fare la differenza e migliorare notevolmente le prestazioni.

Potrebbe anche risultare molto interessante utilizzare tipologie completamente differenti di Neural networks. Quelle più interessanti da prendere in considerazione sono sicuramente le Convolutional Neural Networks, utilizzate anche nel paper [18] per ottenere entusiasmanti risultati in ambito dei giochi Atari, e le Recurrent Neural networks. Quest'ultima tipologia si presta in modo particolare all'ambito di applicazione studiato in questa tesi, poiché sono particolarmente indicate per la previsioni di serie temporali e, di conseguenza, si potrebbero rivelare decisive in un contesto in cui è necessario prendere una sequenza di decisioni correlate fra loro come il mercato azionario o del Forex.

### Sistema di voto

Una importante modifica al progetto potrebbe riguardare le modalità con cui i modelli prendono le decisioni in fase di test. Si ritiene che differenti strategie riescano a gestire diverse casistiche, di conseguenza potrebbe essere estremamente utile costruire un sistema di voto in cui la decisione non viene presa da un unico agente di Deep Q-Learning, ma potrebbe essere presa tramite un utilizzo sincrono di più modelli tale da costruire un

sistema di votazione di vari agenti. Questo permetterebbe di evitare di prendere decisioni in caso di incertezza, eliminando quindi parte del rumore introdotto dall'utilizzo di un unico agente. In particolare si può analizzare il numero di agenti che si trovano in accordo su una certa azione in modo da decidere con più sicurezza qual'è meglio selezionare. Oltre alla riduzione dell'incertezza quindi si ha una maggiore confidenza del modello e un possibile abbassamento di gravi perdite.

### Classificatore di incertezza

Un differente approccio per affrontare il problema del drawdown e dell'incertezza è il seguente. Considerando come dato di fatto che nel mercato del Forex vi siano certe tipologie di eventi che risultano altamente impredicibili, lo stesso non si può dire della strategia con cui l'agente opera.

Si considera dunque un'agente di Deep Q-Learning che ha già trovato una policy convincente ma con evidenti problemi di drawdown. Potrebbe essere utile costruire un classificatore che, dati lo stato attuale dell'agente, lo stato attuale dell'environment e la mossa selezionata dall'agente, discrimini in maniera binaria se in quel momento conviene che l'agente investa o no. In pratica avviene una discriminazione che indica la probabilità che in seguito a una mossa o a una serie di mosse vi sia alta probabilità di drawdown. Si parla dunque di una rete avente come dati di input quelli descritti in precedenza e come output la classificazione del momento successivo come possibile alto drawdown. Se fosse possibile effettuare questa classificazione le performance dell'agente migliorerebbero in maniera netta anche in termini di metriche che tengono in considerazione il Maximum drawdown e la deviazione standard dei profit.

# Bibliografia


[1] Hado van Hasselt Arthur Guez David Silver. «Deep Reinforcement Learning with Double Q-learning». In: (2015). URL: https://arxiv.org/abs/1509.06461.

[2] Greg Brockman et al. *OpenAI Gym*. 2016. eprint: arXiv:1606.01540.

[3] François Chollet. *keras*. https://github.com/fchollet/keras. 2015.

[4] James Cumming. *An Investigation into the Use of Reinforcement Learning Techniques within the Algorithmic Trading Domain*. 2015.

[5] ForexGuida.com. 2016. URL: https://www.forexguida.com/ (visitato il 03–2017).

[6] Xavier Glorot e Yoshua Bengio. «Understanding the difficulty of training deep feedforward neural networks». In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS-10)*. A cura di Yee W. Teh e D. M. Titterington. Vol. 9. 2010, pp. 249–256. URL: http://www.jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf.

[7] Ian Goodfellow, Yoshua Bengio e Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016. Cap. 5.

[8] Investopedia. *Forex Tutorial: The Forex Market*. URL: http://www.investopedia.com/university/forexmarket/ (visitato il 03–2017).

[9] Investopedia. *The Forex Market Tutorial*. URL: http://i.investopedia.com/inv/pdf/tutorials/ForexMarket.pdf (visitato il 03–2017).

[10] Yuxi Li. «Deep Reinforcement Learning: An Overview». In: abs/1701.07274 (2017). URL: http://arxiv.org/abs/1701.07274.

[11] Warren S. McCulloch e Walter Pitts. *Neurocomputing: Foundations of Research*. 1943. Cap. A Logical Calculus of the Ideas Immanent in Nervous Activity, pp. 115–133.

[12] Matthias Plappert. *keras-rl*. https://github.com/matthiasplappert/keras-rl. 2016.

[13] Frank Rosenblatt. «The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain». In: (1958), pp. 386–408.

[14] David Silver. *David Silver Online Course on Reinforcement Learning*. 2015. URL: `http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html` (visitato il 02–2017).

[15] David Silver et al. «Mastering the Game of Go with Deep Neural Networks and Tree Search». In: (28/01/2016). URL: `https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf`.

[16] Richard S. Sutton e Andrew G. Barto. *Reinforcement Learning : An Introduction*. MIT Press, 2017. URL: `"http://people.inf.elte.hu/lorincz/Files/RL_2006/SuttonBook.pdf"`.

[17] Theano Development Team. «Theano: A Python framework for fast computation of mathematical expressions». In: *arXiv e-prints* abs/1605.02688 (mag. 2016). URL: `http://arxiv.org/abs/1605.02688`.

[18] David Silver Alex Graves Ioannis Antonoglou Daan Wierstra Martin Riedmiller Volodymyr Mnih Koray Kavukcuoglu. «Playing Atari with Deep Reinforcement Learning». In: (2013). URL: `https://arxiv.org/abs/1312.5602`.

[19] David Silver Andrei A Rusu Joel Veness Marc G Bellemare Alex Graves Martin Riedmiller Andreas K Fidjeland Georg Ostrovski Stig Petersen Charles Beattie Amir Sadik Ioannis Antonoglou Helen King Dharshan Kumaran Daan Wierstra Shane Legg Demis Hassabis Volodymyr Mnih Koray Kavukcuoglu. «Human-level control through deep reinforcement learning». In: 518 (2015). URL: `http://www.nature.com/nature/journal/v518/n7540/full/nature14236.html`.

[20] Matteo Hessel Hado van Hasselt Marc Lanctot Nando de Freitas Ziyu Wang Tom Schaul. «Dueling Network Architectures for Deep Reinforcement Learning». In: (2015). URL: `https://arxiv.org/abs/1511.06581v3`.