University of Warsaw
Faculty of Economic Sciences

Krzysztof Wojdalski

Nr albumu: 310284

Reinforcement Learning for Trading

Praca magisterska
na kierunku ECONOMICS

The thesis written under the supervision of
dr Pawel Sakowski

2025-05-03

Table of contents

Bibliography

Abstract

This thesis examines the application of reinforcement learning techniques to foreign exchange market trading strategies. The investigation commences with a comprehensive analysis of the FX market structure, including its organizational framework, key participants, and recent evolutionary developments. The author provides a critical assessment of prevailing trends and potential future trajectories in this domain.

The subsequent section establishes the theoretical foundation for the research, presenting a detailed exposition of relevant financial models and artificial intelligence methodologies that underpin the analytical framework.

The empirical component in the third chapter implements a Q-learning based reinforcement learning approach, yielding inconclusive results that warrant further investigation.

1. Introduction

The conventional representation of trading floors as chaotic environments characterized by vocal trader interactions was historically accurate in the 1980s. However, the financial industry has since undergone substantial structural transformation.

The financial sector has consistently functioned as an early adopter of computational innovations. This integration represents a strategic imperative—competitive advantage through technology frequently correlates with financial performance. Hence, the industry has systematically pioneered the deployment of state-of-the-art technologies, ranging from sophisticated Bloomberg terminal infrastructure in the 1990s to contemporary blockchain applications and ultra-low latency systems, all oriented toward maximizing operational efficiency.

Trading entities, functioning as economic agents, aim to optimize market-derived profits. Multiple methodological approaches exist to achieve this objective. Various approaches exist across the investment spectrum: value investors like Benjamin Graham and Warren Buffett employ fundamental analysis with long-term horizons; quantitative firms such as Two Sigma utilize computational methods for systematic trading; while high-frequency trading entities like Citadel Securities implement ultra-low latency strategies to capitalize on microscopic price discrepancies.

In recent years, methodologies predicated on artificial intelligence and machine learning algorithms have demonstrated increasing significance. This phenomenon correlates with computational processing capacity, decreasing infrastructure expenditure requirements, and empirical recognition of cognitive biases in human decision-making processes. A growing consensus within the literature suggests that algorithmic systems should supplement or potentially supersede human involvement in decision-making and execution processes Turner (2015). The progressive automation of trading activities will likely continue its direction, contradicting the stereotypical trading floor representation.

Although machine learning's theoretical foundations date to the 1950s, its explicit implementation in trading contexts remains relatively limited. For example, in institutional foreign exchange trading, only entities with substantial capital resources and sophisticated quantitative infrastructure have developed effective machine learning trading systems Mosic (2017).

This research investigates potential applications of machine learning—specifically reinforcement learning methodologies—in developing trading systems capable of generating statistically significant positive outcomes.

Currently, most systems documented in academic literature aim to maximize either absolute trading profits or risk-adjusted performance metrics. Despite numerous methodological approaches to create consistently profitable systems utilizing variables derived from financial econometrics, fundamental analysis, or machine learning algorithms, many have demonstrated suboptimal performance due to several factors, including:

- Large drawdowns resulting in excessive performance volatility
- High transaction costs rendering strategies impractical
- High computational complexity, particularly problematic for high-frequency trading

Even when empirical research demonstrates exceptional results, upon publication, any competitive advantage tends to diminish through market efficiency mechanisms. Strategies with persistent alpha-generating capabilities must remain proprietary to maintain their effectiveness.

1.1. Statistical Arbitrage

In financial markets the term arbitrage means the practice of exploiting riskless opportunities to make profit. In the purest form, if an asset is priced differently at two different places and a trader has access to both then he would have an arbitrage opportunity. In practice financial markets are designed in such a way that any deterministic arbitrage opportunities vanish, i.e., are being exploited by market participants. Still there are patterns in the market which seem to be persistent. Strategies which use statistical techniques to extract highly reliable patterns and then carry out trades to exploit them are called statistical arbitrage strategies.

Statistical arbitrage is a general term in which assets are put into baskets by market-based similarities. Once the relation between the co-moving financial instruments is found, any deviation from it can be exploited. If that relation is true than either one or both stocks are mispriced and a trader can take a long position on the undervalued one(s) and short position on the overvalued asset(s) to bet on their return to the prevalent relative price behaviour. Statistical arbitrage also fittingly resembles the fact that the strategy hedges risk from market movements. As the pair trade goes short and long on similar assets, the overall risk balance should cancel out and the strategy should be profitable without market risk exposure.

To determine if a security is over- or undervalued one can either use some theoretic approach to determine what the absolute price of the security should be or one could use the idea of relative pricing. In this approach it is only of interest how one security is priced in relation to another security. Now to form a simple mean-reverting pair trading strategy the only thing that remains is a definition of what constitutes a reliable price relationship between the asset prices. There are many ways to identify such security pairs. One approach would be to measure fundamentals (such as some accountancy variables) of the firms. The statistical approach, which also makes the whole concept easily applicable in a system is to use cointegration.

1.1.1. Statistical Arbitrage vs Other Types of Arbitrage

While statistical arbitrage relies on probabilistic relationships between securities, several other arbitrage types exist in financial markets, each with distinct characteristics:

1.1.1.1. Classical Arbitrage

Classical arbitrage involves simultaneously trading two or more fungible instruments and converting between them to capture price differentials. At least one leg might involve derivatives:

- **ETF arbitrage**: Buying an ETF while selling its constituent basket, then redeeming the ETF
- **ADR arbitrage**: Selling an ADR while buying its underlying foreign stock and currency, then creating the ADR
- **Futures arbitrage**: Buying a future while selling its underlying, holding both until expiration

1.1.1.2. Latency Arbitrage

Latency arbitrage leverages speed advantages to be first to complete trades triggered by discrete events. Examples include racing to take stale bids/offers on one exchange after observing trades on another, or using private fill prices to trade ahead of other participants.

1.1.1.3. Time Arbitrage

Time arbitrage captures spreads between buyers and sellers demanding liquidity at different periods. US equity wholesalers practice this by guaranteeing retail buy orders, expecting offsetting sell orders to follow. Opportunities also arise when linked instruments trade on exchanges with different operating hours.

1.1.1.4. Microstructure Arbitrage

This captures price dislocations using exchange-specific idiosyncrasies like matching semantics, fee treatment, and specialized order types. Common opportunities arise from non-continuous matching mechanisms such as opening/ closing auctions and their associated order types.

1.1.1.5. Statistical Arbitrage

Statistical arbitrage trades spreads between instruments based on probabilistic estimations of convergence (mean reversion) or divergence (momentum). The "dispersion trade" bets that index option prices should converge to weighted baskets of options on the index's components. Unlike classical arbitrage, statistical arbitrage involves probability-based relationships that may break down, introducing additional risk elements.

A Taxonomy of Arbitrage Trading

1.1.2. Critical Parameter Selection and Optimization

1.2. Scope and Objectives of the Research

1.2.1. Hypothesis

The research is based on the following hypotheses:

1. AI-driven algorithms can deliver superior value by exceeding benchmark performance metrics in terms of both risk management and return generation
2. Enhanced performance is particularly evident in high-frequency trading scenarios as well as across extended time horizons
3. These algorithms demonstrate the capability to implicitly learn market patterns that would be impossible for humans to identify and efficiently select securities. In consequence, they generate alpha

1.2.2. Objectives of the Research

The primary research goal is to evaluate Reinforcement Learning-based algorithms for multiasset trading, with particular focus on pair trading strategies. The research objectives are structured in three interconnected phases:

1. **Design and Implementation**
   - Develop both basic and extended Reinforcement Learning trading agents for pair trading applications
   - Construct robust testing frameworks to evaluate agent performance
   - Create agents capable of processing financial time series data
2. **Testing and Evaluation**
   - Assess agent performance on out-of-sample data against established benchmarks

    - Measure performance using comprehensive risk-return metrics
    - Analyze robustness under varying market conditions
    - Determine if agents can outperform conventional trading approaches
3. **Analysis and Implications**
    - Examine agent learning evolution and pattern recognition capabilities
    - Investigate how agents integrate with traditional pair trading methodologies
    - Assess transaction cost management effectiveness
    - Determine if agents can identify non-trivial patterns more efficiently than human traders
    - Evaluate the feasibility of deployment in real-world trading environments
    - Identify promising directions for future research

The underlying premise is that these algorithms can systematically outperform benchmarks by detecting and exploiting market patterns that would remain imperceptible to human traders, potentially generating alpha in both high-frequency trading environment.

1.2.3. Methodological Framework

1.3. Thesis Organization and Chapter Overview

This thesis is structured to provide a comprehensive framework for understanding reinforcement learning algorithms applied to trading contexts.

The first section provides a detailed examination of the intersection between artificial intelligence methodologies and financial markets, exploring the historical relationship between quantitative finance and computational science.

The second chapter presents a systematic review of selected literature from quantitative finance, examining both classical equilibrium models such as CAPM (the established paradigm in equity research) and contemporary approaches. This section evaluates advantages and limitations of various financial models, with particular emphasis on algorithmic trading methodologies employed in comparable research contexts.

The third section analyzes machine learning frameworks, providing a theoretical basis for why reinforcement learning may represent an optimal approach for trading applications. It presents a taxonomic comparison of major machine learning categories to elucidate their methodological distinctions, introduces key reinforcement learning concepts with illustrative examples, and addresses potential limitations and implementation challenges associated with these algorithms.

The fourth part details the experimental methodology, including research objectives, data characteristics, experimental design parameters, and empirical results. The primary objective was to develop trading agents capable of statistically outperforming established benchmarks on risk-adjusted performance measures in the foreign exchange market—agents characterized by statistical robustness, adaptive learning capability, and consistent performance metrics. This chapter presents the mathematical formulations and procedural implementations leading to the empirical results, examining each component of the trading system.

The implemented algorithms utilize a dynamic optimization approach. Beyond a value function based on Differential Sharpe Ratio, the system incorporates various technical indicators such as Relative Strength Index to inform algorithmic decision-making processes. The methodology incorporates transaction cost models to simulate realistic trading conditions.

The value function integrates multiple statistical measures, including Sharpe and Differential Sharpe Ratios, to capture both risk and return dimensions. The algorithm outputs agent actions in the discrete action space $-1, 0, 1$. The final section of this chapter evaluates the reinforcement learning-based trading system against two benchmark methodologies:

- A buy-and-hold strategy (maintaining consistent long positions in selected currency pairs)
- Random action generation—producing stochastic values in the domain of $-1, 0, 1$ to determine positions in underlying pairs. This benchmark excludes transaction costs, as such a strategy would incur prohibitive cumulative costs with position changes occurring in approximately two-thirds of states.

The concluding section presents a comparative analysis with similar research and proposes directions for future investigation, addressing research questions such as:

- What additional implementations could enhance performance metrics?
- What methodological limitations were encountered and how might they be addressed in subsequent research?

2. Classical Finance Context and Literature Review

The section introduces articles that correspond with the subject of the current thesis and are considered as fundamentals of modern finance. Specifically, the beginning contains financial market models. The next subchapter includes basic investment effectiveness indicators that implicitly or explicitly result from the fundamental formulas from the first subchapter.

2.0.0.1. Capital Asset Pricing Model

Works considered as a fundament of quantitative finance and investments are Sharpe (1964), Lintner (1965), and Mossin (1966). All these authors, almost simultaneously, formulated Capital Asset Pricing Model (CAPM) that describes dependability between rate of return and its risk, risk of the market portfolio, and risk premium.

Assumptions in the model are as follows:

- Decisions in the model regard only one period,
- Market participants has risk aversion, i.e. their utility function is related with plus sign to rate of return, and negatively to variance of portfolio rate of return,
- Risk-free rate exists,
- Asymmetry of information non-existent,
- Lack of speculative transactions,
- Lack of transactional costs, taxes included,
- Market participants can buy a fraction of the asset,
- Both sides are price takers,
- Short selling exists,

Described by the following model formula is as follows:

$$E(R_P) = R_F + \frac{\sigma_P}{\sigma_M} \times [E(R_M) - R_F]$$

where:

- $E(R_P)$ – the expected portfolio rate of return,
- $E(R_M)$ – the expected market rate of return,
- R_F – risk-free rate,
- σ_P – the standard deviation of the rate of return on the portfolio,
- σ_M – the standard deviation of the rate of return on the market portfolio.

$E(R_P)$ function is also known as Capital Market Line (CML). Any portfolio lies on that line is effective, i.e. its rate of return corresponds to embedded risk. The next formula includes all portfolios, single assets included. It is also known as Security Market Line (SML) and is given by the following equation:

$$E(R_i) = R_F + \beta_i \times [E(R_M) - R_F]$$

where:

- $E(R_i)$ – the expected i-th portfolio rate of return,
- $E(R_M)$ – the expected market rate of return,
- R_F – risk-free rate,
- β_i – Beta factor of the i-th portfolio.

2.1. The Modern Portfolio Theory

The following section discuss the Modern Portfolio Theory developed by Henry Markowitz Stulz (1995). The author introduced the model in which the goal (investment criteria) is not only to maximize the return but also to minimize the variance. He claimed that by combining assets in different composition it is possible to obtain the portfolios with the same return but different levels of risk. The risk reduction is possible by diversification, i.e. giving proper weights for each asset in the portfolio. Variance of portfolio value can be effectively reduced by analyzing mutual relations between returns on assets with use of methods in statistics (correlation and covariance matrices). It is important to say that any additional asset in portfolio reduces minimal variance for a given portfolio but it is the correlation what really impacts the magnitude. The Markowitz theory implies that for any assumed expected return there is the only one portfolio that minimizes risk. Alternatively, there is only one portfolio that maximizes return for the assumed risk level. The important term, which is brought in literature, is the effective portfolio, i.e. the one that meets conditions above. The combination of optimal portfolios on the bullet.

The Markowitz concept is determined by the assumption that investors are risk-averse. This observation is described by the following formula:

$$E(U) < U(E(X))$$

where:

- $E(U)$ – the expected value of utility from payoff;
- $U(E(X))$ – utility of the expected value of payoff.

The expected value of payoff is given by the following formula:

$$E(U) = \sum_{i=1}^{n} \pi_i U(c_i)$$

where:

- π_i – probability of the c_i payoff,
- $U(c_i)$ – utility from the c_i payoff.

One of the MPT biggest flaws is the fact that it is used for ex post analysis. Correlation between assets changes overtime so results must be recalculated. Real portfolio risk may be underestimated. Also, time window can influence the results.

2.2. Efficient Market Hypothesis

In 1965, Eugene Fama introduced the efficient market term. Fama claimed that an efficient market is one that instantaneously discounts new information arrival in the market price of a given asset. Because this definition applies to financial markets, it determined the further belief that it is not possible to beat the market because assets are correctly priced. Also, if this hypothesis were true, market participants could not be better or worse. Their portfolio return would be a function of new, unpredictable information. In that respect, the only role of an investor is to manage assets so that the risk is acceptable. Fama (1965)

It is highly unlikely that EMH exists in its strongest form due to successful quantitative hedge funds that consistently beat the markets. For instance, Renaissance Technologies hedge fund generated on average 40% per annum in the last 30 years Shen (2017).

Formally, Efficient Market Hypothesis states that a market is efficient with respect to information set F_t if it is impossible to make economic profits by trading on the basis of that information set. In other words, it is not possible to achieve any better than risk-adjusted average rate of return. In its essence, that claim is consistent with classical price theory Weber (2012). Over time, other versions (forms) of the EMH have been introduced - weak, semi-strong, and strong Fama, E. F.;Malkiel (1970).

Definition 1. Weak Form of the EMH F_t represents only the information contained in the past price history of the market as of time t

According to this form, investors cannot achieve above-normal returns by analyzing historical price patterns and trading volumes to forecast future price movements. Nevertheless, fundamental analysis may still yield exceptional results since markets are not entirely efficient at identifying under or overvalued securities. Therefore, market participants can potentially discover profitable investment opportunities through thorough examination of companies' financial reports.

Definition 2. Semi-Strong Form of the EMH F_t represents all information publicly available at time t

It states that neither technical nor fundamental analysis can be exploited for gaining superior returns, and only non-public material information might help in achieving above average results.

Definition 3. Strong Form of the EMH F_t represents all information (public and private) known to anyone at time t.

The strong form rejects the idea of any possibility of consistently beating the market. According to this idea, any kind of information, public or non-public, is completely embedded into current financial asset prices. In other words, there is no advantage for anyone in the market. Returns that deviate from expected values are attributed to pure randomness.

2.2.0.1. Critic of strong form of the EMH

There are at least a few documented anomalies that contradicts with efficient market hypothesis. For example, price/earnings (P/E) measure can help in systematically outperforming stocks Malkiel (2003). The neglected firm effect claims that “uninteresting” companies, often ignored by market analysts are sometimes incorrectly priced, and offer investors potentially fruitful opportunities. Another phenomenon that cannot be explained by the strong form of EMH is so called the January effect Haug and Hirschey (2006). According to the authors of “The January Effect” working paper, returns reached in January has predictive power for the upcoming 11 months. It persists for both small and large cap companies.

Although the strongest form in its essence is justified, logically correct, it is rather unlikely that it explains the reality, even due to the effects mentioned above.

2.3. Factor Models

2.4. Modern Approaches in Algorithmic Trading

2.5. Critical Analysis of Traditional Financial Models

2.5.1. Criticism of the Efficient Market Hypothesis and Modern Portfolio Theory

The Efficient Market Hypothesis (EMH) posits that investors cannot earn excess returns on a risk-adjusted basis. Complementarily, Markowitz’s Modern Portfolio Theory (MPT) provides a framework for constructing portfolios that optimize risk-return profiles based on available investment options.

If markets are truly efficient as EMH suggests, portfolios constructed using the Markowitz framework would not generate excess returns. However, if market inefficiencies exist, these portfolios could potentially earn returns exceeding market benchmarks. Notably, MPT allows investors to incorporate subjective views regarding expected asset returns, contributing to its widespread applicability and Markowitz’s Nobel Prize recognition.

Both theories gained popularity synergistically and share similar criticisms. Despite MPT becoming an industry standard, it faces several significant limitations:

1. **Problematic Risk Measurement**: MPT treats upside and downside price movements equally in risk calculations. It also assumes constant volatility, whereas real-world markets demonstrate volatility clustering across various time horizons.

2. **Simplified Cash Flow Assumptions**: The model ignores transaction costs and tax implications. It also fails to account for investor preferences regarding dividend income.

3. **Unrealistic Liquidity Assumptions**: MPT assumes investors can take arbitrarily large positions at current market prices without affecting those prices, contradicting market impact realities.

4. **Idealized Investor Behavior**: The theory presumes all market participants are rational, risk-averse, and share identical investment time horizons.

2.6. Selected investment performance measures

Literature review does not explicitly mention performance metrics. Nevertheless, the theoretical frameworks established by these scholars remain valuable, as they constitute the foundation for numerous measures. The widespread adoption of these indicators can be attributed to their accessibility and interpretability for typical investors (Marte, 2012).

Sharpe (1966) introduced the $\frac{R}{V}$ indicator, commonly known as the Sharpe Ratio (S), expressed by the formula:

$$S_i = \frac{E(R_i - R_F)}{\sigma_i}$$

where:

- R_i – the i-th portfolio rate of return,
- R_F – risk-free rate
- σ_i – the standard deviation of the rate of return on the i-th portfolio.

Treynor (1965) proposed an alternative approach using β_i instead of σ_i in the denominator. This formulation is expressed as:

$$T_i = \frac{R_i - R_F}{\beta_i}$$

where:

- R_i – the i-th portfolio rate of return,
- R_F – Risk-free rate
- β_i – Beta factor of the i-th portfolio.

Both S and T indicators are relative measures. Their values should be compared against a benchmark to determine whether a portfolio is effectively managed. Higher (lower) values indicate that the analyzed portfolios performed better (worse) than the benchmark.

Jensen's alpha constitutes an absolute performance metric that quantifies the differential between realized returns and those predicted by the Capital Asset Pricing Model. Positive values signify outperformance relative to theoretical expectations for the i-th portfolio. This measure, which enjoys substantial credibility among professionals, is mathematically defined as:

$$\alpha_i = R_i - R_F - \beta_i(R_m - R_F)$$

where:

- R_i – the i-th portfolio rate of return,
- R_F – Risk-free rate
- β_i – Beta factor of the i-th portfolio.

Drawdown measures the decline from a historical peak in an asset. The formula is expressed as:

$$D(T) = \max\{max_{0,t\in(0,T)} X(t) - X(\tau)\}$$

The Sterling ratio (SR)

The maximum drawdown (MDD) at time T represents the maximum Drawdown over the asset's historical trajectory, expressed as:

$$MDD(T) = \max_{\tau\in(0,T)} \left[\max_{t\in(0,\tau)} X(t) - X(\tau)\right]$$

2.7. Empirical Evidence Against Perfect Market Efficiency

2.7.1. Market Anomalies

2.8. Key Concepts from Reinforcement Learning

2.9. Seminal Works in RL

2.10. RL Applications in Trading Systems

3. Machine Learning

In this chapter, the term **machine learning** and its subfields are explained. The discussion also encompasses potential applications for trading financial instruments.

As the field evolves, numerous definitions of machine learning emerge from various sources. In this subchapter, the author has selected definitions that accurately capture the essence of the discipline.

What is machine learning? The most widely accepted definitions are as follows:

- “Field of study that gives computers the ability to learn without being explicitly programmed.” - Arthur Samuel, a pioneer in machine learning and computer gaming Samuel (1959)
- “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.” - Tom Mitchell, a computer scientist and E. Fredkin University Professor at Carnegie Mellon University (CMU) Mitchell (1997)

The latter is particularly regarded as an elegant and modern definition. Less formal, but equally relevant observations come from textbook authors in the discipline:

- “Pattern recognition has its origins in engineering, whereas machine learning grew out of computer science. However, these activities can be viewed as two facets of the same field...” - Christopher Bishop
- “One of the most interesting features of machine learning is that it lies on the boundary of several different academic disciplines, principally computer science, statistics, mathematics, and engineering. ...machine learning is usually studied as part of artificial intelligence, which puts it firmly into computer science ...understanding why these algorithms work requires a certain amount of statistical and mathematical sophistication that is often missing from computer science undergraduates.” - Stephen Marsland Marsland (2009)

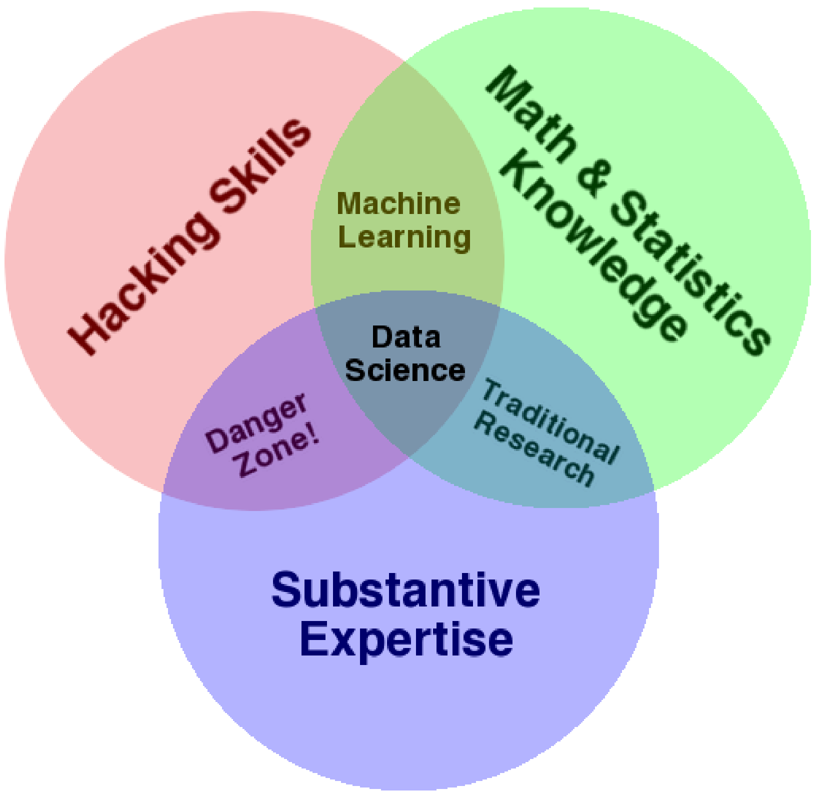

Figure 1: Data Science Graph

Despite numerous concepts and perspectives on what machine learning entails, the general objective remains consistent: Machine learning involves building models that sufficiently resemble reality, are optimal with respect to a value function, and can subsequently be utilized for predictions on new data.

3.1. Why is machine learning important?

Machine learning facilitates solving problems that are difficult or impossible to address deterministically Jason (2013). Variables may be missing or observed values may contain embedded errors. Traditional models are often susceptible to being under- or overdetermined. They may fail to generalize adequately or may be excessively general. An appropriate machine learning model should provide an approximate solution incorporating only relevant components.

3.2. Classification of machine learning algorithms

In machine learning (ML), tasks are categorized based on how learning/feedback (P) is received and/or the type of problem they address. The following categories can be distinguished:

3.2.1. Supervised Learning

Supervised learning represents a fundamental paradigm in machine learning where algorithms learn from labeled training data to make predictions or decisions without explicit programming. At its core, supervised learning involves a dataset consisting of input-output pairs, where each example contains features (input variables) and their corresponding target values or labels (output variables). The primary objective is to learn a mapping function that can accurately predict the output value for new, previously unseen inputs.

Formally, given a dataset of pairs $(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n)$, where X_i represents the feature vector and Y_i is the corresponding target value, supervised learning aims to find a function f such that $Y = f(X)$ which minimizes a predefined loss function $L(f(X), Y)$. The function f is constrained by the model class chosen for the task, which determines the complexity and expressiveness of the relationships that can be captured.

The mathematical foundation of supervised learning lies in statistical learning theory and optimization. For instance, in a linear regression model, we seek parameters β that minimize the sum of squared errors: $\min_\beta \sum(Y_i - X_i\beta)^2$. In more complex models like neural networks, we optimize weights and biases across multiple layers using gradient-based methods to minimize error functions across the entire training dataset.

Supervised learning tasks typically fall into two main categories:

- Classification: When the output variable Y is categorical or discrete (e.g., spam/not spam, fraud/legitimate, image categories)
- Regression: When the output variable Y is continuous (e.g., stock prices, temperature, house prices)

The choice of algorithm depends on various factors including the nature of the problem, dataset characteristics, computational resources, and the desired balance between model interpretability and predictive performance. Common supervised learning algorithms include:

- Linear models: Linear regression for regression tasks and logistic regression for classification
- Tree-based methods: Decision trees, random forests, and gradient boosting machines
- Support vector machines: Effective for both classification and regression with high-dimensional data
- K-nearest neighbors: A non-parametric method that makes predictions based on similarity measures
- Ensemble methods: Combining multiple models to improve overall performance and robustness
- Neural networks: Deep learning architectures capable of capturing complex non-linear relationships

A critical aspect of supervised learning is the bias-variance tradeoff. Simple models may underfit the data (high bias), failing to capture important patterns, while overly complex models may overfit (high variance), learning noise rather than underlying relationships. Techniques such as regularization, cross-validation, and ensemble methods help manage this tradeoff to create models that generalize well to unseen data.

In financial applications, supervised learning has become important for various tasks such as price prediction, risk assessment, credit scoring, fraud detection, and market sentiment analysis. For instance, in predicting stock prices, historical market data with known outcomes serves as the training set, where features might include technical indicators, fundamental data, and macroeconomic variables, while the target variable could be future price movements or returns.

The effectiveness of supervised learning models for financial markets is often challenged by the non-stationary nature of markets, where relationships between variables change over time and in the presence of noise. This necessitates continuous model updating and validation against recent data. Additionally, feature engineering—the process of creating relevant variables from raw data—plays a crucial role in financial applications, often requiring domain expertise to identify meaningful predictors.

3.2.2. Unsupervised Learning

Unsupervised learning constitutes an approach in machine learning wherein computational algorithms identify intrinsic patterns and structural relationships within unlabeled datasets without explicit instructional guidance. In contrast to supervised methodologies, this paradigm operates in the absence of predetermined target variables or classificatory labels to direct the analytical process. The algorithm instead autonomously discerns the inherent organizational structure embedded within the data corpus itself.

Formally, in unsupervised learning, the dataset consists of a collection of unlabeled examples $X_1, X_2, ..., X_N$, where each X_i represents a feature vector. The primary objective is to create a model that processes these feature vectors to either transform them into another representation or extract meaningful patterns that can solve practical problems.

The mathematical foundation of unsupervised learning involves finding patterns, relationships, or structures within the data space. For instance, in clustering algorithms, we seek to minimize within-cluster distances while maximizing between-cluster distances, often expressed as optimization problems such as minimizing $\sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2$ where C_i represents clusters and μ_i their centroids.

Unsupervised learning encompasses several key application areas:

- **Clustering**: Identifying natural groupings within data where instances within the same cluster exhibit high similarity while being dissimilar to instances in other clusters. Common algorithms include K-means, hierarchical clustering, and DBSCAN.
- **Dimensionality Reduction**: Transforming high-dimensional data into lower-dimensional representations while preserving essential information. Techniques such as Principal Component Analysis (PCA), t-SNE, and autoencoders fall into this category.
- **Density Estimation**: Modeling the probability distribution that generates the observed data, enabling better understanding of data characteristics and generation of new samples. Methods include Gaussian Mixture Models and kernel density estimation.
- **Anomaly Detection**: Identifying instances that deviate significantly from the norm or expected patterns within the dataset. These outliers often represent rare events, errors, or fraudulent activities that warrant special attention.
- **Feature Learning**: Automatically discovering useful representations from raw data that can subsequently enhance performance in downstream tasks, including supervised learning problems.

In financial applications, unsupervised learning proves valuable for market segmentation, identifying trading patterns, detecting fraudulent transactions, and discovering latent factors driving market movements. The absence of labeled data makes unsupervised learning useful in exploratory analysis and when dealing with novel or evolving financial phenomena where historical classifications may not exist or apply.

The effectiveness of unsupervised learning in finance is often measured by metrics such as silhouette scores for clustering quality, explained variance for dimensionality reduction, or business impact metrics like improved portfolio diversification or fraud detection rates. As markets evolve and data complexity increases, unsupervised learning provides tools for uncovering hidden structures and relationships in financial data.

3.2.3. Semi-Supervised Learning

Semi-supervised learning represents a hybrid approach in machine learning where the dataset contains both labeled and unlabeled examples. Formally, the explanatory variables X_i are available for all observations, but the labels Y_i are only available for a subset of the data. Typically, the quantity of unlabeled examples significantly exceeds the number of labeled examples.

The primary objective, similar to supervised learning, is to discover the relationship $Y = f(X)$. This is generally accomplished through a strategic combination of supervised and unsupervised learning techniques. The underlying principle involves labeled observations effectively "diffusing" their labels to unlabeled observations that exhibit high similarity according to specific criteria.

The appeal of semi-supervised learning lies in its ability to leverage large amounts of unlabeled data, which is often more abundant and less costly to obtain than labeled data. By incorporating these unlabeled examples, the learning algorithm can potentially develop a more robust and generalizable model than would be possible using only the limited labeled examples.

Several key techniques in semi-supervised learning include:

- **Self-training**: An iterative process where a model trained on labeled data makes predictions on unlabeled data, then adds high-confidence predictions to the training set.
- **Co-training**: Using multiple views or feature subsets to train separate models that teach each other by labeling unlabeled examples for one another.
- **Graph-based methods**: Constructing similarity graphs where nodes represent data points and edges represent similarities, allowing label propagation through the graph structure.
- **Generative models**: Using techniques like Gaussian Mixture Models to model the joint distribution of features and labels.
- **Semi-supervised SVMs (S3VMs)**: Extensions of Support Vector Machines that incorporate unlabeled data by seeking decision boundaries that avoid dense regions of unlabeled points.

3.2.4. Reinforcement Learning

Reinforcement learning (RL) represents a distinct paradigm in machine learning where an agent learns optimal behaviors through trial-and-error interactions with an environment. The agent perceives the environment's state as a feature vector, executes actions, and receives rewards that may lead to new states.

The fundamental objective in RL is to develop a policy—a function mapping states to actions that maximizes expected cumulative rewards over time. This approach differs significantly from both supervised and unsupervised learning paradigms.

Key characteristics of reinforcement learning include:

- **Sequential Decision Making**: RL addresses problems where decisions occur in sequence and have long-term implications.
- **Delayed Rewards**: Actions may not yield immediate benefits but contribute to greater cumulative rewards in the future.
- **Exploration vs. Exploitation**: Agents must balance discovering new potentially better strategies against utilizing known rewarding actions.

RL is particularly well-suited for non-stationary environments where relationships between variables evolve over time. The agent continuously adapts its policy to changing conditions, making it useful for applications where dynamics constantly shift.

This interdisciplinary field incorporates influences from engineering, economics, mathematics, neuroscience, psychology, and computer science. Applications span diverse domains including game playing, robotics, resource management, logistics, and, increasingly, financial trading strategies.

Unlike supervised learning, RL does not rely on labeled examples that specify exactly what an algorithm should do. Instead, the agent must learn from its own actions, sense states, and accumulate experience. The only feedback received is a scalar reward signal.

While unsupervised learning focuses on identifying structures in unlabeled datasets, RL specifically aims to maximize a long-run value function comprising summed (discounted) rewards. Finding data patterns may be useful, but it addresses a fundamentally different objective than the RL problem.

The environment changes stochastically and interacts with the agent, requiring a policy design that balances exploration of new actions against exploitation of known solutions. This prevents purely greedy behavior that might lead to suboptimal outcomes.

3.3. Algorithmic Trading Systems

In its essence, the investor's or trader's main goals is to optimize some measure of trading system performance, such as risk-adjusted return, economic utility, or simple profit. In the work the author presented direct optimization methods, based on reinforcement learning. It is flexible and can work out for trading a single asset (+ risk-free instrument), but also a portfolio consisting of n-instruments. There are also other options which often directs to non-optimal solutions. In the following section the author brought up different types of algorithmic systems and outlines advantages, disadvantages and differences between them.

3.3.1. Components of Automated Trading Systems

3.3.1.1. Data Handlers

3.3.1.2. Reference Data

3.3.1.3. Optimization

3.3.1.4. Execution

3.3.1.5. UI / UX

3.3.2. Rule-Based Trading Strategies

A rule-based strategy uses explicit, human-defined rules for buying and trading. These are interpretable and do not rely on forecasting models or machine-generated predictions but rather employ conditional logic where trading decisions follow deterministic paths based on market conditions. Typically, such rules derive from established market heuristics or technical analysis principles with proven effectiveness.

The clear causal relationship between market conditions and trading actions provides transparency, allowing traders to understand exactly why positions change.

Rule-based systems apply consistently across all market conditions, eliminating psychological biases and ensuring reproducible trading decisions. This deterministic nature enables straightforward performance evaluation

The framework adapts to changing markets while maintaining its structure. Parameter optimization enhances performance while preserving core logic, allowing systematic evolution through iterative refinement.

3.3.2.1. Example: Cross-Venue Arbitrage Strategy

Strategy exploits temporary price discrepancies of the same asset traded on different exchanges or venues.

The arbitrage opportunity can be formalized through the price differential:

$$D_t = P_{X,t} - P_{Y,t}$$

where: - P_X = price of asset on venue X - P_Y = price of asset on venue Y

In theoretically perfect market efficiency, this differential would approach zero after accounting for all transaction costs, as arbitrage opportunities would be immediately exploited by market participants.

The trading signal generation follows this decision rule:

$$\text{Signal} = \begin{cases} \text{BUY on Y, SELL on X,} & \text{if } D_t > \tau + c \\ \text{BUY on X, SELL on Y,} & \text{if } D_t < -\tau - c \\ \text{CLOSE POSITIONS,} & \text{if } |D_t| < \delta \\ \text{HOLD,} & \text{otherwise} \end{cases}$$

where: - τ represents the entry threshold - δ represents the exit threshold - c accounts for transaction costs including fees and slippage

For instance, with `MSFT` stock listed on two electronic venues:

$$D_t = P_{\text{MSFT,Venue1},t} - P_{\text{MSFT,Venue2},t}$$

If the price differential exceeds 3 pips plus transaction costs ($\tau = 0.0003$), the strategy signals to buy on the venue offering the lower price and simultaneously sell on the venue with the higher price.

The optimization problem for this strategy can be expressed as:

$$\max_{\tau,\delta} \sum_{t=1}^{T} P_t(\tau, \delta) - TC_t$$

where: - $P_t(\tau, \delta)$ represents the profit at time t given thresholds τ and δ - TC_t represents the transaction costs

3.4. Trading based on Forecasts

Forecast-based trading systems employ supervised learning techniques to generate price predictions from a set of input variables. These systems optimize for statistical accuracy metrics such as mean squared error during the training phase.

The forecasting process represents merely an intermediate step in the trading workflow. The predicted prices generated by the Forecasting System subsequently inform trading decisions within the Trading Rules module. Performance evaluation occurs through financial metrics in the Profits/Losses evaluation module $U(\theta, \theta')$. A significant limitation of this approach is that the Forecasting module typically outputs only the predicted price while discarding potentially valuable contextual information from the original inputs. This exemplifies a fundamental limitation in forecast-based trading systems: the information loss problem. When a system predicts that MSFT will reach \$300 tomorrow based solely on technical indicators, it discards critical contextual variables such as abnormal trading volume and volatility patterns. The formal representation of this problem can be expressed as:

$$f : X \to \hat{y}$$

where: - $X \in \mathbb{R}^n$ represents the input space - $\hat{y} \in \mathbb{R}$ represents the scalar price prediction

The dimensionality reduction from $\mathbb{R}^n$ to $\mathbb{R}$ results in significant information loss that may be crucial for optimal trading decisions.[1]

3.4.0.1. Training a Trading System on Labeled Data

This methodology employs direct integration between the Trading System and Input Series components. The system generates trading signals based on a corpus of labeled trades (training dataset), while execution occurs in response to market inputs (Input Series).

Formally, the system can be represented as:

[1]The information loss can be quantified through mutual information: $I(X; \hat{y}) < I(X; X)$, indicating that the prediction $\hat{y}$ contains less information about market conditions than the original feature set X.

$$f : X \to a$$

where: - $X \in \mathbb{R}^n$ represents the input feature space - $a \in A$ denotes the action space (buy, sell, hold)

The optimization objective can be expressed as:

$$\min_{\theta} \mathcal{L}(\theta) = \sum_{i=1}^{N} \ell(f_\theta(X_i), a_i^*)$$

where: - θ represents the model parameters - a_i^* denotes the labeled optimal action for input X_i - ℓ is a loss function measuring deviation from labeled actions

The system's operational efficacy is contingent upon the Trading Module's capacity to extract and utilize information from both Input Series and Labeled Trades $\theta^{'}$. A fundamental limitation exists in this approach: since optimization targets the Trading System rather than the utility function $U(\theta, \theta^{'})$, the resultant solution typically exhibits sub-optimal characteristics in terms of financial performance.

3.4.0.2. Direct Optimization of Performance

In this approach, there is no intermediate step and labeled data is not given. The environment is observed, X_t, the system carries out an action, and receives a scalar reward for its past activities, representing the trading performance in some form (e.g. rate of return). Based on this reward, the system alters the way it behaves in subsequent episodes and steps.

For example, consider a reinforcement learning agent trading Microsoft (MSFT) stock. At time t, the agent observes market state X_t (e.g., MSFT price, volume, technical indicators) and selects action $a_t \in \{-1, 0, 1\}$ representing short, neutral, or long positions. The agent receives reward r_t based on position value:

$$r_t = a_{t-1} \cdot \frac{P_t^{\text{MSFT}} - P_{t-1}^{\text{MSFT}}}{P_{t-1}^{\text{MSFT}}}$$

The agent optimizes policy $\pi_\theta(a|X_t)$ to maximize expected cumulative reward:

$$\max_{\theta} \mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} r_t\right]$$

where $\gamma \in [0, 1]$ is a discount factor prioritizing near-term returns.

3.4.1. Selected aspects of reinforcement learning

This section examines pertinent aspects and challenges within the reinforcement learning paradigm.

3.4.1.1. Components of a reinforcement learning system

Reinforcement learning systems are designed to address sequential decision-making problems by selecting actions that maximize cumulative discounted future rewards. The following section explains the components of reinforcement learning using chess and trading as illustrative examples. This subsection draws partial inspiration from Sutton and Barto (2017).

3.4.1.1.1. Environment (E)

It defines the possible states and actions. In chess, this encompasses the rules and potential configurations of pieces on the board. It is important to note that some states will never be reached.

In trading contexts, environmental rules might stipulate that an agent can only take positions of 0 or 1, or that portfolio asset weights must sum to 1.

3.4.1.1.2. State (s)

It represents a snapshot of the environment at time t, containing information that guides the agent's next action selection. States can be terminal, indicating the agent cannot choose further actions, thus ending an episode—a sequence of state-action pairs from start to finish. In trading applications, a state at time t might comprise various financial metrics: returns, implied/realized volatility, moving averages, economic indicators, technical signals, and market sentiment measures.

3.4.1.1.3. Action (a)

Given the current state, the agent selects an action that transitions to a new state, either deterministically or stochastically. The action selection process itself may be deterministic or probability-based. In chess, an action involves moving a piece according to the rules. In trading, actions might include establishing long or short positions, maintaining neutral exposure, or adjusting portfolio weights.

3.4.1.1.4. Policy (π)

It maps environmental states to corresponding actions. In psychological terms, this resembles stimulus-response associations. A policy may take various forms—from lookup tables to functions (linear or otherwise). Trading applications often involve continuous variables requiring extensive computation to determine optimal outcomes. As the core component of a reinforcement learning agent, policies fundamentally determine behavior. Policies can be stochastic rather than deterministic. Even after numerous episodes, an efficient algorithm may continue exploring alternative states instead of exclusively exploiting currently optimal actions.

3.4.1.1.5. Reward (r)

Rewards are the essence of reinforcement learning predictions. Value function, as previously stated, is a sum of, often discounted, rewards. Without them, as components of value function, an agent would not be able to spot (or optimal) better policies actions are based on. Hence, it is assumed rewards are the central point, required element, of every RL algorithm. Rewards are always given as a scalar, single value that is retrieved from the environment, that is easy to observe and interpret. With value function it is much harder since it can be obtained only by calculating a sequence of observations a RL agent makes over its lifetime.

3.4.1.1.6. Value Function

It predicts future, typically discounted rewards to help the agent determine the desirability of states. Value functions depend on initial states (S_0) and the agent's selected policy. Every state should have an associated value, defaulting to zero for unexplored paths. The general formula for a value function is:

$$V^{\pi} = \mathbb{E}_{\pi}[\sum_{k=1}^{\infty} \gamma^k r_{t+k} | s_t = s]$$

where:

- γ is a discount factor from the range $[0;1]$. It measures how much more instant rewards are valued. The smaller it is the more immediate values are relatively more relevant and cause algorithm to be more greedy. Sometimes γ is equal to 1 if it is justified by the design of the whole agent.

Value estimation, as a area of research in RL is probably the most vital one in the last decade. The most important distinction between different RL algorithms lies as to how it is calculated, in what form, and what variables it incorporates.

3.4.1.1.7. Model (m)

A model shows the dynamics of environment, how it will evolve from S_{t-1} to S_t. The model helps in predicting what the next state and next reward will be. They are used for planning, i.e. trial-and-error approach is not needed in order to achieving the optimal policy. Formally, it is a set of transition matrices:

$$\mathbb{P}^a_{ss^{'}} = \mathbb{P}[s^{'} | s, a]$$
$$\mathbb{R}^a_s = \mathbb{E}[r | s, a]$$

where:

- $\mathbb{P}_s s^{'} a$ is a matrix of probability of transitions from state s to state $s^{'}$ when taking action a. Analogously, $\mathbb{R}^a_s$ is an expected value of reward when an agent is in state s and taking action a

3.4.1.2. Exploration/exploitation

A fundamental challenge in reinforcement learning concerns the balance between exploration and exploitation. To optimize cumulative rewards, an agent must execute actions that previously yielded substantial payoffs (exploitation). However, during the initial learning phase, the agent lacks knowledge regarding effective strategies. Consequently, it must investigate potentially beneficial actions for its current state (exploration). This dilemma remains unresolved in the field, although several methodological approaches have been developed to address it.

3.4.1.2.1. ϵ-greedy policy

The most straightforward approach involves predominantly greedy behavior, wherein an agent selects the action (A_t) that maximizes the utilized value function (e.g., $Q_t(a)$). However, with a probability of ϵ, the agent randomly selects an available action, independent of action value estimates. This algorithm ensures comprehensive exploration of all actions across all states, ultimately leading to $Q_t(a) = q_*(a)$. Consequently, the probability of selecting the optimal action converges to greater than $1 - \epsilon$, approaching certainty. The limitation of this method lies in its minimal indication of practical efficacy. Asymptotic guarantees may require excessive time in authentic environments. Research demonstrates that small ϵ values facilitate greater initial rewards but typically underperform compared to larger ϵ values as the number of steps increases.

3.4.1.2.2. Optimistic initial values

One of the techniques to improve agent's choices is based on the idea of encouraging the agent to explore. Why is that? If the actual reward is smaller than initially set up action-value methods, an agent is more likely to pick up actions that potentially can stop getting rewards that constantly worsen value function $q(a)$. Eventually, the system does a lot more exploration even if greedy actions are selected all the time.

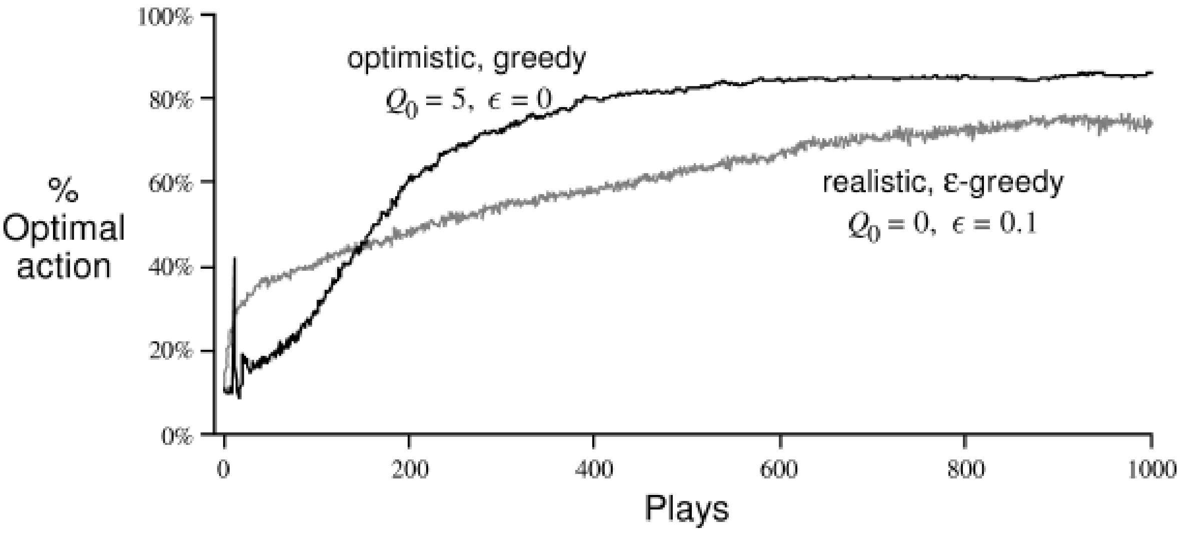

Figure 2: The effect of optimistic initial action-value estimates on the 10-armed testbed

3.4.1.2.3. Upper-Confidence-Bound Action Selection

The other method for handling the exploration/exploitation problem is by using the special bounds that narrow with the number of steps taken. The formula is as follows:

$$A_t = \underset{a}{\text{argmax}}[Q_t(a) + c\sqrt{\frac{ln_t}{N_t(a)}}],$$

where:

- ln_t is the natural logarithm of \$t%
- $N_t(a)$ - the number of times that action a has been selected prior to time t

- c - the exploration rate

The idea of this soltuion is that the square-root part is an uncertainty measure or variance in the a estimation. The use of the natural logarithm implies that overtime square-root term, so does the confidence interval, is getting smaller. All actions will be selected at some point, but the ones with non-optimal values for $Q(a)$ are going to be selected much less frequently over time. UCB performs well, but it is harder to apply (generalize) for a broader amount of problems than ϵ-greedy algorithm. Especially, when one is dealing with nonstationary problems. In such situations, algorithms more complex than those presented in this subsection should be selected.

3.4.2. Limitations

Reinforcement learning, while powerful, does not constitute a universal solution for all machine learning challenges. Its applicability is primarily restricted to problems characterized by distinct states, determinable policies, and well-defined value functions. A state merely represents a signal received by the agent—a momentary representation of the environment at a specific point in time.

The majority of traditional reinforcement learning methodologies focus predominantly on states and their associated values. Although this approach proves adequate for relatively simple environments, it encounters significant limitations when applied to more complex scenarios. Notably, tabular representations become inadequate for storing expected values of states or state-action pairs, particularly when variables exhibit continuous properties. Such representations would exceed available memory constraints.

Consequently, alternative methodologies must be employed to address these challenges and achieve computational efficiency in solving complex reinforcement learning problems.

3.5. Timeline of Reinforcement Learning Algorithms

Reinforcement learning has evolved significantly over the decades, progressing from theoretical foundations to sophisticated algorithms capable of algorithmic trading problems. This section presents an overview of key developments in RL, with particular attention to their applicability in algorithmic trading.

It must be note

3.5.1. 1950s–1980s: Theoretical Foundations

- 1952-1954: Bellman established Temporal Difference (TD) methodology and Markov Decision Process (MDP) formalism as mathematical foundations for RL.
- 1979: Watkins conceptualized Q-learning principles, though formal articulation came later.
- Financial application: Primarily theoretical, establishing frameworks for sequential investment decision-making.

3.5.2. 1989-1992: Fundamental Algorithms

- 1989: Q-learning (Watkins)
  - Off-policy, model-free algorithm for determining state-action value functions.
  - Financial relevance: Suitable for discrete trading decisions.

    - Limitation: Poor scalability for extensive state/action spaces.
- 1992: SARSA
    - On-policy variant of Q-learning evaluating the implemented policy.
    - Financial relevance: More robust in non-stationary financial environments.

3.5.3. Mid-1990s: Approximation Methods

- Implementation of function approximators for value function estimation.
- Financial application: Enabled management of larger state spaces approaching financial data complexity.

3.5.4. 2013–2015: Deep Reinforcement Learning

- 2013 (DeepMind): Integration of Q-learning with deep neural networks.
    - Experience replay and target networks enhanced learning stability.
- 2015: DQN achieved human-comparable performance in Atari environments.
    - Financial implementation: First significant market applications using neural networks for feature extraction from financial data.

3.5.5. 2015–2016: Policy Gradient Methodologies

- REINFORCE (Williams, 1992): Initial stochastic policy gradient formulation.
- Actor-Critic: Separated policy representation from value function estimation.
    - Financial advantage: Direct return optimization and accommodation of continuous action spaces for position sizing.

3.5.6. 2016–2018: Advanced Algorithmic Frameworks

- A3C (2016): Asynchronous Advantage Actor-Critic
    - Parallel learning across multiple agents for greater efficiency.
- DDPG (2015): Deep Deterministic Policy Gradient
    - Specialized for deterministic continuous policy learning.
- PPO (2017): Proximal Policy Optimization
    - Conservative policy updates with improved reliability.
- SAC (2018): Soft Actor-Critic
    - Off-policy approach with entropy maximization for exploration-exploitation balance.
- Financial applications: Execution optimization, portfolio construction, and market-making strategies.

3.5.7. 2018–Present: Multi-Agent Systems and Meta-Learning

- Multi-Agent RL: Models market participant interactions for realistic impact simulation.
- Meta-RL: Develops systems capable of rapid adaptation to new market regimes.

3.5.8. 2020s Onwards: Practical Implementation

- Hierarchical RL: Integrates strategic planning with tactical execution.
- Safe RL: Implements safeguards against catastrophic market losses.
- Explainable RL: Enhances decision transparency for regulatory compliance.

- Current priorities: Sample efficiency, risk-sensitive objectives, and regulatory constraint integration.

3.5.9. Model-free vs Model-based

Reinforcement learning algorithms can be categorized into three principal classifications:

- Model-Based approaches - These methodologies function with the prerequisite that the environmental model is known in advance. The agent selects actions through deliberate planning and systematic exploration within this predefined model structure. The Markov Decision Process (MDP) represents a quintessential example of this paradigm, requiring explicit knowledge of both the Markov transition probability matrix and the associated reward function.
- Model-Free approaches - These methodologies acquire knowledge directly from state-action values or policies through experiential learning. They can achieve comparable behavioral outcomes without prior knowledge of the environmental model in which the agent operates. In practical applications, reinforcement learning is predominantly employed in environments where transition matrices remain unknown. Within a given policy framework, each state possesses a value defined as the cumulative utility (reward) commencing from that state. Model-free methods typically demonstrate lower efficiency compared to model-based approaches, as environmental information is integrated with potentially inaccurate state value estimations

1.

3.6.

3.6.1. Value-Based vs Policy-Based vs Actor-Critic

Reinforcement learning algorithms can be further categorized based on their optimization approach:

- Value iteration - a model-based algorithm that computes the optimal state value function by improving the estimate of $V(s)$. It starts with initializing arbitrary values and then updates $Q(s,a)$ and $V(s)$ values until they converge. The pseudocode is as follows:

Algorithm 1 Value Iteration Algorithm. Based on @Alpaydin2013

```
1: Initialize $V(s)$ to arbitrary values
2: repeat
3:     for all $s \in S$ do
4:         for all $a \in A$ do
5:             $Q(s,a) \leftarrow E[r|s,a] + \gamma \sum_{s' \in S} P(s'|s,a)V(s')$
6:         end for
7:         $V(s) \leftarrow \max_a Q(s,a)$
8:     end for
9: until $V(s)$ converge
```

- Policy iteration - while in the previous bullet the algorithm is improving value function, policy iteration is based on the different approach. Concretely, there are two functions to be optimized $V^{\pi}(s)$ and $\pi^{'}(s)$. This method is based on the premise that a RL agent cares about finding out the right policy. Sometimes, it is more convenient to directly use policies as a function of states. Below is the pseudocode:

Algorithm 2 Policy Iteration Algorithm. Based on @Alpaydin2013

```
1: Initialize $\pi^{'}$ to arbitrary values
2: repeat
3:     $\pi \leftarrow \pi^{'}$
4:     Compute the values using $\pi$ by solving the linear equations
5:     $V^{\pi}(s) = E[r|s, \pi(s)] + \gamma \sum_{s' \in S} P(s'|s, \pi(s)) V^{\pi}(s')$
6:     for all $s \in S$ do
7:         $\pi^{'}(s) \leftarrow \text{argmax}_a (E[r|s, a] + \gamma \sum_{s' \in S} P(s'|s, a) V^{\pi}(s'))$
8:     end for
9: until $\pi = \pi^{'}$
```

- Actor-Critic methods represent a hybrid approach that combines elements of both value-based and policy-based learning. Such algorithms maintain two structures: an “actor” that determines the policy (action selection), and a “critic” that evaluates actions through value function approximation. This synthesis often yields improved stability and sample efficiency compared to pure policy-based methods while retaining their capability to handle continuous action spaces.

  The actor-critic architecture can be formalized as follows: Actor (Policy): $\pi_\theta(a|s)$ parameterized by θ

  Critic (Value): $V_w(s)$ parameterized by w

Algorithm 3 Actor-Critic Algorithm

```
1: Initialize actor parameters $\theta$ and critic parameters $w$
2: repeat
3:     Observe state $s_t$
4:     Select action $a_t \sim \pi_\theta(a|s_t)$
5:     Execute action $a_t$, observe reward $r_t$ and next state $s_{t+1}$
6:     Compute TD error: $\delta_t = r_t + \gamma V_w(s_{t+1}) - V_w(s_t)$
7:     Update critic: $w \leftarrow w + \alpha_w \delta_t \nabla_w V_w(s_t)$
8:     Update actor: $\theta \leftarrow \theta + \alpha_\theta \delta_t \nabla_\theta \ln \pi_\theta(a_t|s_t)$
9: until convergence
```

where: α_w and α_θ are learning rates for the critic and actor, respectively.

3.6.1.1. Model Free Learning

Model Free learning constitutes a subcategory of reinforcement learning algorithms employed when the underlying model remains unknown. In this approach, the agent enhances its decision-making accuracy through environmental interactions without possessing explicit knowledge of the transition matrix. This methodology is particularly suitable for trading environments, as financial markets inherently lack a definable model and exhibit non-stationary transition probabilities. Consequently, direct application of value or policy iteration algorithms becomes infeasible.

Despite the opacity of the Markov Decision Process and its components, the agent can accumulate experience from sampled states. The theoretical foundation suggests that the distribution of sampled states will eventually converge to that of the transition matrix. Similarly, $Q(s, a)$ values converge to Q^* and the policy π^* approaches optimality. This convergence requires that all state-action pairs be visited infinitely often, and that the agent adopts a greedy strategy once it identifies the optimal action for each state.

3.6.1.2. On-Policy vs Off-Policy

3.6.1.3. On-Policy vs Off-Policy

Reinforcement learning algorithms can be categorized based on their policy evaluation and improvement mechanisms. The distinction between on-policy and off-policy methods lies in how they utilize experience for learning:

On-policy methods learn and improve the same policy that is used for action selection during environmental interaction. These algorithms evaluate and refine the behavioral policy directly from the experience it generates. SARSA exemplifies this approach, updating Q-values using:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

where a_{t+1} is selected according to the current policy.

Off-policy methods, conversely, learn a target policy different from the behavioral policy used for exploration. This separation enables learning from historical data or experiences generated by alternative strategies. Q-learning represents this category, updating values using:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

The fundamental advantage of off-policy learning in financial applications is its ability to learn optimal strategies while following exploratory or risk- averse policies, facilitating more efficient learning from historical market data without requiring direct market interaction during training.

3.6.1.4. Single-Agent vs Multi-Agent

Reinforcement learning frameworks can be distinguished based on the number of decision-making entities within the environment:

Single-agent reinforcement learning involves a solitary agent interacting with a stationary or quasi-stationary environment. The agent optimizes its policy to maximize expected cumulative rewards without considering the actions of other decision-makers. This paradigm assumes that environmental state transitions are influenced solely by the agent's actions and stochastic processes inherent to the environment.

Multi-agent reinforcement learning encompasses scenarios where multiple agents operate simultaneously within a shared environment. Each agent's actions influence not only their individual rewards but potentially the state transitions and rewards experienced by other agents. This introduces strategic considerations analogous to game theory, where optimal policies depend on the behavior of other participants.

The distinction becomes particularly relevant in market microstructure analysis, where the collective behavior of numerous market participants influences price formation processes. However, most practical trading applications adopt a single-agent perspective, treating market dynamics as an exogenous environment rather than explicitly modeling other participants.

3.6.1.5. Discrete vs Continuous

Reinforcement learning methodologies can be categorized based on the nature of their state and action spaces:

Discrete reinforcement learning encompasses systems with finite, enumerable state and action spaces. Such frameworks characterize the environment via a bounded set of distinct states, while the agent chooses from a limited repertoire of possible actions. This formulation permits tabular representations of value functions and policies, thus enabling precise solutions through dynamic programming when environmental dynamics are fully specified. Within financial contexts, discrete models may conceptualize market conditions as categorical regimes (e.g., bullish, bearish, consolidating) and actions as specific portfolio allocations (e.g., long position, short position, market neutrality).

Continuous reinforcement learning pertains to systems with uncountable state and/or action spaces. Such methodologies facilitate the processing of real-valued observations and actions, thereby requiring function approximation techniques for the representation of value functions and policies. The mathematical framework can be expressed as:

$$\pi_\theta : \mathcal{S} \to \mathcal{P}(\mathcal{A})$$

where:

- $\mathcal{S} \subset \mathbb{R}^n$ represents a continuous state space,
- $\mathcal{A} \subset \mathbb{R}^m$ denotes a continuous action space.

Financial markets inherently exhibit continuous characteristics in both state representations (e.g., prices, volatility measures, economic indicators) and potential actions (e.g., position sizing, risk parameters). Consequently, continuous reinforcement learning frameworks, particularly those employing neural network function approximators, have demonstrated superior efficacy in capturing the complex, non-linear relationships present in financial data.

4. Design of the Trading Agent

4.1. Action Space

4.1.1. Book Pressure-Related Variables

4.1.2. Volume-Related Variables

4.1.3. Last Trade Price-Related Variables

4.1.4. Time-Related Variables

4.1.5. Technical Indicators

4.1.6. Autoregressive Variables

4.2. Reward Function

4.2.1. Differential Sharpe Ratio

Differential Sharpe Ratio (DSR) is a dynamic extension of the Sharpe ratio. This measure captures the marginal impact of returns at time t on the Sharpe Ratio. The calculation begins with the following formulations:

$$A_n = \frac{1}{n} R_n + \frac{n-1}{n} A_{n-1}$$

$$B_n = \frac{1}{n} R_n^2 + \frac{n-1}{n} B_{n-1}$$

At $t = 0$, both values equal zero. These serve as the foundation for calculating the exponentially moving Sharpe ratio on an η time scale:

$$S_t = \frac{A_t}{K_\eta \sqrt{B_t - A_t^2}}$$

where:

- $A_t = \eta R_t + (1-\eta) A_{t_1}$
- $B_t = \eta R_t^2 + (1-\eta) B_{t_1}$
- $K_\eta = (\frac{1-\frac{\eta}{2}}{1-\eta})$

The differential Sharpe ratio offers several advantages in algorithmic systems (Moody & Wu, 1997):

- Recursive updating - eliminating the need to recalculate mean and standard deviation of returns each time. The formulations for A_t and B_t allow simple calculation through updates for R_t and R_t^2.
- Efficient on-line optimization - the formula structure enables quick computation through updates of the most recent values.
- Interpretability - the measure is easily explained, as it quantifies how the latest return affects the Sharpe ratio (risk and reward).

4.2.2. Other Reward Functions

4.3. Value Function

4.4. Policy

4.5. Monte Carlo Control

Monte Carlo Control is an RL algorithm that leverages complete episode experiences to estimate optimal policies. Unlike temporal difference methods, Monte Carlo techniques do not rely on bootstrapping, instead deriving value estimates directly from completed trajectories.

Formally, it follows the following steps:

1. **Policy Evaluation**: For each state-action pair (s, a) encountered in an episode, the action-value function is updated according to:

   $$Q(s,a) \leftarrow Q(s,a) + \alpha[G_t - Q(s,a)]$$

   where:

   - $G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}$ represents the return following time t
   - α denotes the learning rate
   - T signifies the terminal time step

2. **Policy Improvement**: The policy is updated to be ϵ-greedy with respect to the current action-value function:

   $$\pi(a|s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A(s)|}, & \text{if } a = \text{argmax}_{a'} Q(s,a') \\ \frac{\epsilon}{|A(s)|}, & \text{otherwise} \end{cases}$$

The algorithmic implementation is presented below:

Algorithm 4 Monte Carlo Control with ϵ-greedy Policy

```
 1: Initialize $Q(s,a)$ arbitrarily for all $s \in S, a \in A(s)$
 2: Initialize $\pi$ to be $\epsilon$-greedy with respect to $Q$
 3: Initialize Returns$(s,a)$ as empty list for all $s \in S, a \in A(s)$
 4: for each episode do
 5:     Generate an episode following $\pi$: $S_0, A_0, R_1, S_1, ..., S_{T-1}, A_{T-1}, R_T$
 6:     $G \leftarrow 0$
 7:     for $t = T-1, T-2, ..., 0$ do
 8:         $G \leftarrow \gamma G + R_{t+1}$
 9:         Unless the pair $S_t, A_t$ appears in $S_0, A_0, ..., S_{t-1}, A_{t-1}$:
10:             Append $G$ to Returns$(S_t, A_t)$
11:             $Q(S_t, A_t) \leftarrow$ average(Returns$(S_t, A_t)$)
12:             Update $\pi$ to be $\epsilon$-greedy with respect to $Q$
13:     end for
14: end for
```

4.5.1. Advantages for Trading Applications

Monte Carlo Control offers several distinct advantages in financial trading contexts that complement other policy optimization approaches:

- **Model-Free Learning**: The algorithm requires no prior knowledge of market dynamics or transition probabilities
- **Reduced Bias in Value Estimation**: By utilizing complete episode returns rather than bootstrapped estimates, Monte Carlo methods eliminate the bias introduced by function approximation errors
- **Effective Learning from Episodic Experience**: Trading naturally decomposes into episodes (e.g., daily sessions, trade lifecycles), aligning well with Monte Carlo's episodic learning paradigm and enabling direct optimization of terminal performance metrics.
- **Robustness to Partial Observability**: Financial markets often exhibit partially observable characteristics; Monte Carlo methods demonstrate greater resilience to such conditions compared to one-step temporal difference approaches.
- **Simplified Credit Assignment**: The algorithm effectively addresses the temporal credit assignment problem by directly attributing rewards to state-action pairs, facilitating more accurate evaluation of trading decisions that may have delayed consequences.

In this research, Monte Carlo Control serves as a complementary approach to policy optimization, particularly valuable for initial policy exploration and establishing baseline performance metrics against which more sophisticated algorithms can be evaluated.

4.5.2. Proximal Policy Optimization (PPO)

PPO advances policy gradient methods with enhanced sample efficiency and stability. It resolves step size determination challenges through a clipped objective function that effectively constrains policy updates.

The core innovation of PPO lies in its objective function:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$$

where:

- $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ represents the probability ratio between the new and old policies
- $\hat{A}_t$ is the estimated advantage function, which quantifies how much better an action is compared to the average action in a given state; defined as $\hat{A}_t = Q(s_t, a_t) - V(s_t)$, measuring the relative benefit of selecting action a_t in state s_t
- ϵ is a hyperparameter that constrains the policy update

The algorithm can be formalized as follows:

4.5.3. Advantages for Trading Applications

The empirical analysis of PPO implementation in financial trading contexts reveals several distinctive advantages that substantiate its selection for this research:

Algorithm 5 Proximal Policy Optimization (PPO)

```
1: Initialize policy parameters $\theta$ and value function parameters $\phi$
2: for iteration = 1, 2, ... do
3:     Collect set of trajectories $\mathcal{D}_t = \{\tau_i\}$ by running policy $\pi_\theta$ in the environment
4:     Compute rewards-to-go $\hat{R}_t$
5:     Compute advantage estimates $\hat{A}_t$ using GAE or other methods
6:     Compute policy ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$
7:     Optimize the PPO-Clip objective:
8:     $\theta \leftarrow \text{argmax}_\theta \frac{1}{|\mathcal{D}_t|T} \sum_{\tau \in \mathcal{D}_t} \sum_{t=0}^{T} \min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)$
9:     Update value function by regression on mean-squared error:
10:    $\phi \leftarrow \text{argmin}_\phi \frac{1}{|\mathcal{D}_t|T} \sum_{\tau \in \mathcal{D}_t} \sum_{t=0}^{T} (V_\phi(s_t) - \hat{R}_t)^2$
11: end for
```

- **Algorithmic Stability**: The policy clipping mechanism functions as a constraint on the magnitude of policy updates, effectively mitigating the risk of catastrophic divergence during the training process. This characteristic is particularly valuable in financial markets where volatility clusters can induce erratic learning trajectories.
- **Enhanced Sample Efficiency**: Comparative analyses demonstrate that PPO exhibits superior learning efficiency, requiring substantially fewer environmental interactions to achieve convergence relative to conventional policy gradient methodologies. This property is especially advantageous when utilizing finite historical market data or when computational resources impose constraints on simulation iterations.
- **Diminished Hyperparameter Sensitivity**: Experimental evidence indicates that PPO maintains robust performance across diverse hyperparameter configurations, thereby reducing the dimensionality of the optimization problem associated with algorithm calibration. This characteristic facilitates more efficient experimental design and validation procedures.
- **Native Support for Continuous Action Parameterization**: The mathematical formulation of PPO inherently accommodates continuous action spaces without discretization requirements, aligning precisely with the continuous nature of trading decisions such as position sizing, entry timing, and risk allocation parameters.

In the context of this research, PPO has been implemented as the foundational policy optimization algorithm, facilitating the development of an adaptive trading agent capable of extracting complex, non-linear relationships from market data while maintaining learning stability across diverse market regimes.

4.6. Step Size

5. Implementation of the Trading Agent

The following assumptions are established for the trading mechanisms and algorithmic simulation framework: (b) Zero latency exists between market data reception and trade execution when a favorable signal is present. (a) The absence of market competition enables execution at counterparty prices (selling at bid price and purchasing at ask price). (c) Position constraints limit holdings to x contract(s) at any given moment, with transactions restricted to whole contract units. (d) A transaction fee of x% of the execution price is applied to all trades.

5.1. Design of the research

The whole system can be divided into three main parts:

- Data preprocessing - taking FX data from Bloomberg with use of the dedicated API, parsing the data and adjusting it for the further analysis. The system is dedicated for currency trading, however with little adjustments it could fit in other asset classes as well.
- Variable extraction - not all preprocessed currency pairs are relevant and worth adding. For instance, if USD/CNH is highly correlated with USD/CNY it is senseless to add the latter to the portfolio. #TO DO
- State-action space - the extracted variables, based on time series for currency pairs, are merged into state space

5.1.1. Assumption

In the work, the author has assumed that:

- Zero slippage - the FX market is liquidity is good enough that there the execution price is equal to the price shown by the venue (Bloomberg)
- Zero market impact - trades executed by the agent are not big enough that they can move the market and cause significant market impact

5.1.2. Experimental Environment

The experimental environment was implemented using Python 3.12 with the PyTorch deep learning framework. The system was developed and tested on a Linux-based computational environment with CUDA support for efficient neural network training.

5.1.3. Experimental Procedure

The experimental methodology followed a structured sequence of operations:

1. Data acquisition and preprocessing were conducted initially. This involved:
    - Elimination of invalid or anomalous data points
    - Partitioning of the dataset into training and testing segments using a 4:6 ratio
    - Configuration of appropriate experimental parameters
2. For the trading simulation phase:
    - A selection of ten currency pairs was made according to specific criteria
    - Three distinct reinforcement learning algorithms were implemented to simulate trading activities
    - Price data and performance metrics were systematically collected and compared
3. Comprehensive analysis was performed, incorporating:
    - Examination of individual currency pair characteristics
    - Integration of findings from the algorithmic trading simulations
    - Identification of patterns across different market conditions
4. The experimental results underwent rigorous evaluation, including:
    - Statistical assessment of performance metrics

- Comparative analysis against benchmark strategies
- Critical discussion of implications for algorithmic trading applications

5.2. Data Preparation

5.2.1. Data Collection

5.3. Data Preparation

5.4. Raw Data Schema

Field	Description
ts_recv	The capture-server-received timestamp expressed as the number of nanoseconds since the UNIX epoch.
size	The order quantity.
ts_event	The matching-engine-received timestamp expressed as the number of nanoseconds since the UNIX epoch.
channel_id	The channel ID assigned by Databento as an incrementing integer starting at zero.
rtype	The record type. Each schema corresponds with a single rtype value.
order_id	The order ID assigned at the venue.
publisher_id	The publisher ID assigned by Databento, which denotes dataset and venue.
flags	A bit field indicating event end, message characteristics, and data quality.
instrument_id	The numeric instrument ID.
ts_in_delta	The matching-engine-sending timestamp expressed as the number of nanoseconds before ts_recv.
action	The event action. Can be Add, Cancel, Modify, cleaR book, Trade, Fill, or None.
sequence	The message sequence number assigned at the venue.
side	The side that initiates the event. Can be Ask for a sell order (or sell aggressor in a trade), Bid for a buy order (or buy aggressor in a trade), or None where no side is specified by the original source.
symbol	The requested symbol for the instrument.
price	The order price expressed as a signed integer where every 1 unit corresponds to 1e-9, i.e. 1/1,000,000,000 or 0.000000001.

List of Figures

Bibliography

Fama, E. F.;Malkiel, B. G. 1970. "Efficient capital markets: A review of theory and empirical work." *Journal of Finance* 25 (2): 383–417.

Fama, Eugene F. 1965. "The Behavior of Stock-Market Prices." *The Journal of Business* 38 (1): 34. https://doi.org/10.1086/294743.

Haug, Mark, and Mark Hirschey. 2006. "The january effect." https://doi.org/10.2469/faj.v62.n5.4284.

Jason, Brownlee. 2013. "What is Machine Learning?" https://machinelearningmastery.com/what-is-machine-learning/.

Lintner, John. 1965. "Security prices, risk, and maximal gains from diversifivation." *Jf* 20 (4): 587–615. https://doi.org/10.1111/j.1540-6261.1965.tb02930.x.

Malkiel, Burton G. 2003. "The Efficient Market Hypothesis and Its Critics." *Journal of Economic Perspectives* 17 (1): 59–82. https://doi.org/10.1257/089533003321164958.

Marsland, Stephen. 2009. *Machine Learning: An Algorithmic Perspective.* https://doi.org/10.1111/j.1751-5823.2010.00118_11.x.

Mitchell, Tom M. 1997. *Machine Learning.* 1. https://doi.org/10.1145/242224.242229.

Mosic, Ranko. 2017. "Deep Reinforcement Learning Based Trading Application at JP Morgan Chase." *Medium.* https://medium.com/@ranko.mosic/reinforcement-learning-based-trading-application-at-jp-morgan-chase-f829b8ec54f2.

Mossin, Jan. 1966. "Equilibrium in a capital asset market." *Econometrica* 34 (4): 768–83. https://doi.org/10.2307/1910098.

Samuel, Arthur L. 1959. "Some Studies in Machine Learning Using the Game of Checkers Some Studies in Machine Learning Using the Game of Checkers." *IBM Journal* 3. https://www.cs.virginia.edu/%7B~%7Devans/greatworks/samuel1959.pdf.

Sharpe, William F. 1964. "Capital Asset Prices: A Theory of Market Equilibrium under Conditions of Risk." *The Journal of Finance* 19 (3): 425–42. https://doi.org/10.2307/2329297.

Shen, Lucinda. 2017. "Here's How Much the Top Hedge-Fund Manager Made Last Year." http://fortune.com/2017/05/16/hedge-fund-james-simons-renaissance-technologies/.

Stulz, Rene M. 1995. "American Finance Association, Report of the Managing Editor of the Journal of Finance for the Year 1994." *The Journal of Finance* 50 (3): 1013. https://doi.org/10.2307/2329297.

Sutton, R S, and A G Barto. 2017. "Reinforcement Learning: An Introduction." *Neural Networks IEEE Transactions on* 9 (2): 1054. https://doi.org/10.1109/TNN.1998.712192.

Turner, Matt. 2015. "The robot revolution is coming for Wall Street traders." http://www.businessinsider.com/robots-to-replace-wall-street-traders-2015-8?IR=T.

Weber, Thomas A. 2012. "Price Theory in Economics." In *The Oxford Handbook of Pricing Management.* https://doi.org/10.1093/oxfordhb/9780199543175.013.0017.