Ensemble Reinforcement Learning to Forecast Time Series Data: A Performance Analysis

Zara Andreia Albino Madureira

Dissertation written under the supervision of professor Pedro Afonso Fernandes.

Dissertation submitted in partial fulfilment of requirements for the MSc in Business Analytics, at the Universidade Católica Portuguesa, June 2023.

Abstract

This study aims to investigate the feasibility of applying Temporal Difference models to forecasting time-series data. Initially, this research offers an overview of the key concepts of Approximate Dynamic Programming, covering the basic principles of Dynamic Programming and the structure and architecture of Reinforcement Learning. Subsequently, a concise outline of related literature is presented, highlighting valuable and interesting contributions regarding related topics. The methods used in this analysis encompass Temporal Difference algorithms, in particular TD(0), TD(λ), and GTD2. To assess the suitability of these models in predicting time-series data, their performance is compared to that of benchmark models, including a Hodrick-Prescott filter and Auto-Regressive models, when applied to economic indicators such as Gross Domestic Product, Private Consumption, Investment, and Exports in Portugal. The model's performance was assessed through the analysis of three main indicators, the Mean Absolute Error, the Mean Square Error, and the Root Mean Square Error. By comparing the performance of benchmark and proposed models, the study suggests that temporal difference models indulge in higher quality predictions, proving themselves to be reliable tools to forecast time-series data.

Keywords: Approximate Dynamic Programming, Reinforcement Learning, Dynamic Programming, Markov Decision Processes, Economics Forecasting, Temporal Differences Methods.

Title: Ensemble Reinforcement Learning to Forecast Time Series Data: A performance Analysis

Author: Zara Andreia Albino Madureira

Sumário

O presente estudo tem como objetivo explorar a aplicabilidade de modelos de diferenças temporais na previsão de dados de séries temporais. Em primeiro lugar, é fornecida uma visão geral dos principais conceitos da Programação Dinâmica Aproximada, a começar pelos conceitos fundamentais da programação dinâmica e indo até à arquitetura da aprendizagem de reforço. Segue-se uma breve descrição de trabalhos complementares que abordam temas relacionados, onde são revelados resultados valiosos e interessantes de outros autores. Os métodos utilizados nesta análise são algoritmos de diferenças temporais, em particular TD(0), TD(λ) e GTD2. A fim de avaliar a viabilidade destes modelos aplicados a séries temporais, o seu desempenho quando aplicados a métricas económicas - produto interno bruto, consumo privado, investimento e exportações em Portugal - são diretamente comparados com a performance de um filtro Hodrick-Prescott e um modelo Auto-Regressivo, que foram desenvolvidos para fins de *benchmarking*. A performance dos modelos foi avaliada através da análise de três métricas, o erro médio absoluto, o erro quadrático médio e a raíz do erro quadrático médio. Ao comparar o desempenho dos modelos de referência e dos modelos propostos, observa-se que os modelos de diferenças temporais fornecem previsões de maior qualidade, provando ser ferramentas eficazes na previsão de séries temporais.

Palavras-chave: Programação Dinâmica Aproximada, Aprendizagem de Reforço, Programação Dinâmica, Processos de Decisão de Markov, Previsão Económica, Métodos de Diferenças Temporais.

Título: Aprendizagem de Reforço por Agrupamento para Previsão de Dados de Séries Temporais: Uma Análise de Desempenho

Autor: Zara Andreia Albino Madureira

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

Artificial Intelligence (AI) has progressed significantly in the past decades, involving the development and refinement of its algorithms. These are designed to address various real-world problems, covering a broad spectrum of complexity - from simple control and reaction systems to sophisticated human-like robots (Hunt et al., 2014).

As a sub-field of AI, Machine Learning contributes with algorithms that are able to learn from the data provided with the main goal of generating experience-based predictions. Several approaches have been explored by researchers to improve their applicability as well as the accuracy of their results (Zhou and Liu, 2021). The learning process of the machines can be supervised, unsupervised, or neither, which is the case for Reinforcement Learning (RL) (Sutton and Barto, 2018).

The algorithms built under the RL field focus on solving multi-decision problems, i.e., problems in which several actions must be taken to achieve the solution. Positions are mapped to actions and this schema is used by an agent to interact with a set and unknown environment. The agent attempts several actions and learns from their outcomes, which consists of a learning structure similar to the one seen when humans learn; for instance, when playing games - the player learns through trial and error until it recognizes successful patterns in its actions and acts accordingly. This leads to the discovery of optimal behavior, thus leading to the solution of the problem (Szepesvári, 2010) (Sutton and Barto, 2018) (Kober et al., 2013).

Given the nature of RL, it is crucial to consider the exploration-exploitation trade-off (Sutton and Barto, 2018). Ideally, the agent would take a wider range of actions to learn from all of them, meaning that the model would promote exploration. However, it is similarly relevant to promote the selection of the best action based on the knowledge it acquired previously, i.e., exploitation. The uncertain, dynamic, and on-growing environment accentuates the importance of balancing the trade-off between the two to maximize long-term rewards. This concept leads to the Approximate Dynamic Programming (ADP) approach.

The learning process developed within ADP is based on the ability to find the optimal solution by attempting to solve the same problem multiple times, with the main goal being to solve decision-making problems in dynamic environments. The algorithms focus on learning and establishing a value function, representing the expected reward for each

state, as well as an optimal policy, which specifies the action to be taken in each state. The exploration-exploitation trade-off is optimized since the agent is exploring new actions while simultaneously storing and using information previously acquired in past actions by constantly updating the value function and policy (Bertsekas and Tsitsiklis, 1996) (Szepesvári, 2010) (Bertsekas, 2011).

ADP encompasses various techniques, among which we highlight Temporal Difference (TD) learning, a popular approach in RL that involves updating the value function based on the difference between the expected and actual rewards received by the agent (Sutton, 1988). TD methods can be used as a component of the ADP algorithm to update the value function, improving this process in terms of computational efficiency (Szepesvári, 2010).

This research is developed under the following research questions: Is RL relevant to forecast economic/business series? Are RL algorithms effective and efficient to forecast time series? Which TD method withholds the best results for time series forecasting? By answering these questions, we aim to investigate the potential application of Temporal Difference models for time-series data analysis.

To evaluate the effectiveness of these models, their performances were compared to the ones of a Hodrick-Prescott filter and Auto-Regressive (AR) models, which served as benchmark models. The research considers the Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) as performance metrics.

Corporate performance is directly linked to economic developments (Skare and Hasić, 2016) and on that account, the focus will be on the following four economic aggregates: Gross Domestic Product, Exports, Imports, and Private Consumption. The dataset regards official quarterly data retrieved from Statistics Portugal (in Portuguese, Instituto Nacional de Estatística) combined with data from the Bank of Portugal (in Portuguese, Banco de Portugal).

This thesis is structured as follows. The foundations and relevant concepts concerning dynamic programming and reinforcement learning are presented to provide proper basic knowledge to understand the models used in the research process. Related work is then acknowledged by exploring previous successful applications of similar structured models to forecast time series data. The section that follows consists of a dataset analysis and a thorough description of the models used. The findings are then presented, followed by a

discussion regarding the same, and complemented by interesting future work possibilities related to this research. Finally, the conclusion of the study is shown. The R code used for this investigation can be found in the appendices.

2 Fundamentals

2.1 Problem Formulation

Before diving deep into the details of the theories covered in this research, it is of the most relevance to comprehending how the problems at hand are structured and which components to take into consideration in the solution-seeking process.

Many real-world challenges can be classified as multi-step decision problems characterized by an established environment, an agent that interacts with it, a set of states, and a range of possible actions. Within this structure, the agent interacts with the environment by performing actions that will change their current state, and these actions are associated with a reward or a penalty. The goal is for the agent to find the sequence of actions that leads to the highest possible total reward, i.e., the optimal solution of the problem, and, for that purpose, it is consistently encouraged to act in a way that will converge to the optimal solution, through the rewards associated with each action (Bellman and Dreyfus, 1962), (Hayes et al., 2022), (Charpentier et al., 2021).

As a result of this framework, three other concepts become relevant - policy, value, and reward functions. Policy is the function that maps states into actions, and it can be deterministic or stochastic, meaning that it can depend only on the state itself, or it can be based on the probability distribution of the actions when the current state is known. The purpose of the value function is to reflect the quality of a state-action pair in the long run, in other words, to demonstrate how good an action is in a given state (Hayes et al., 2022), (Gattami et al., 2021),(Charpentier et al., 2021). And finally, the reward function defines the rewards and penalties, which will be crucial to guide the agent's learning process. Contrary to the value function, the reward function expresses what action is good considering the momentary condition in which the agent is in (Bellman and Dreyfus, 1962). The interaction between these components is displayed below.

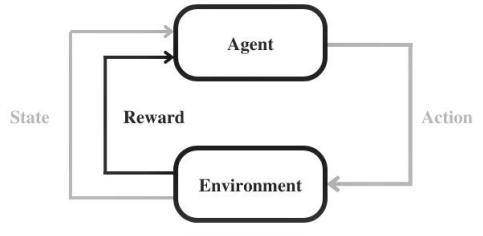

Figure 1: Illustration of the structure of a multi-dimension problem.

2.2 Markov Decision Processes

Markov Decision Processes (MDP) are one of the fundamental components used to assemble RL algorithms. These processes are a result of describing the sequential decision-making process of the problems mathematically, considering the constant interaction between the agent and the environment. The details of these processes can be found thoroughly explained in the work of Szepesvári (2010) as well as in the work of Ljungqvist and Sargent (2004), and this section is based on these two historical references.

An MDP considers four main components:

- X: A countable non-empty set of all possible states of the environment;
- A: A countable non-empty set of all possible actions;
- r: An immediate reward function, which provides the expected immediate reward associated with taking action a while in state x;
- T: A transition model describing the probability of moving from state x to another state y knowing the last chosen action a taken in state x.

As noted in Ljungqvist and Sargent (2004), for a random stochastic process $M = (X, A, T)$, for all $k \geq 1$ and all t, a Markov process satisfies:

$$P(x_{t+1}|x_t, x_{t-1}, ..., x_{t-k}) = P(x_{t+1}|x_t). \tag{1}$$

This condition is known as the Markov property and it's a mathematical way to set and describe the memory loss of the given stochastic process. This means that the process only takes into account the most recent state while disregarding past ones. MDP is a valuable tool for modeling problems in which an agent learns by interacting with a system to attain the optimal solution through a sequence of decisions, regardless of how the process began (Szepesvári, 2010).

2.3 Dynamic Programming

2.3.1 Definition and Key Components

In the early days of optimization, the primary method for tackling such problems as linear programming, which is effective when there are only a few decision variables and a simple objective function. However, as decision problems grew in complexity, linear programming became less reliable in finding optimal solutions. Needing a differentiable function whose derivative wouldn't always be null, having binary variables, retrieving problematic or even non-existent solutions, and getting misleading results through assessing relative extrema instead of absolute extrema are just some of the many difficulties these algorithms face. These and more are explored in greater detail in (Bellman and Dreyfus, 1962) (Bellman, 1952).

To overcome these limitations, dynamic programming theory was introduced by Richard Bellman in the 1950s, a technique focused on dividing problems into sub-problems that takes advantage of functional equations to represent the optimal solution mathematically. This technique behaves correctly under more complex scenarios, providing reliable optimal solutions (Bellman and Dreyfus, 1962) (Bellman, 1952).

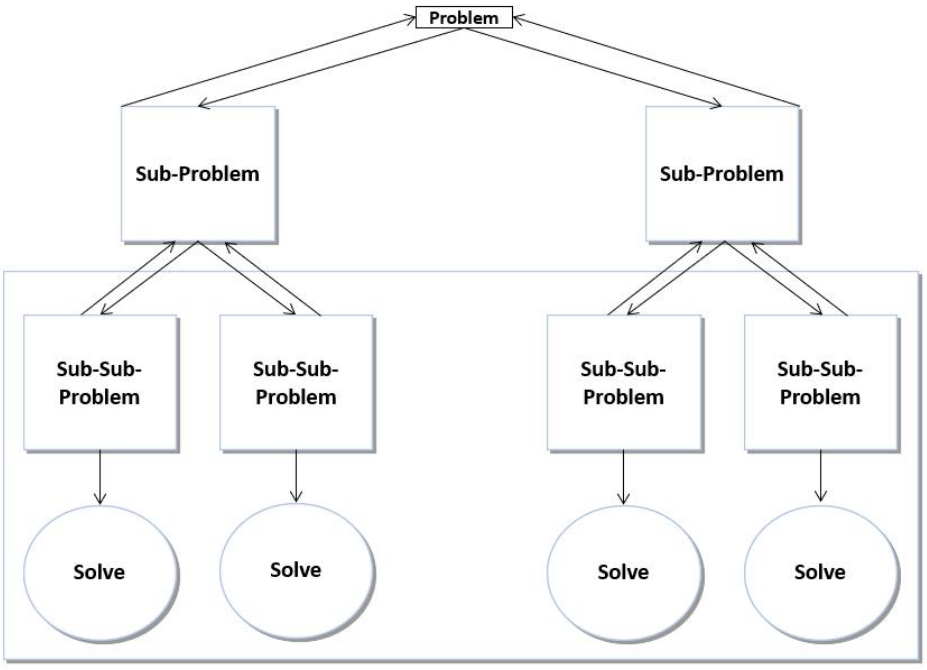

Figure 2: Illustration of the structure of dynamic programming techniques.

2.3.2 Deterministic and Stochastic Approaches

Dynamic programming can be applied to problems of deterministic and stochastic nature, as well as problems that are a combination of both. The algorithms start with the last stages of the problem and work to the very first ones, thus being characterized by a backward recursive approach (Bertsekas, 2011).

In deterministic problem-solving, dynamic programming algorithms work based on the assumption that the states are predictable without uncertainty or random components. The goal is to find the solution within specified rules and constraints regarding the environment of the problem. Thus by breaking it down into smaller and simpler sub-problems, the algorithms then solve them sequentially, allowing for an assessment of all possible actions at each step or stage (Bertsekas, 2011). Some applications of deterministic dynamic programming are finding the shortest or cheapest path, effectively allocating resources, and inventory and production planning, for example.

The deterministic approach works wonderfully in contexts where the rules and constraints are easily specified. However, that is not always the case in real-world problems, hence the need for stochastic dynamic programming techniques.

Stochastic dynamic programming, referred to simply as dynamic programming, has the significant advantage of being able to deal with large-scale problems that consider many decision variables. The decision-making process is assessed through probabilities distributions that represent uncertainty in each step (Bertsekas, 2011) (Ross, 1995). It has been generally contemplated as the most viable approach for solving stochastic optimal control problems, hence its substantial contributions towards machine learning applications (Sutton and Barto, 2018). Examples of these algorithms' applications are the optimization of portfolios, transportation systems and agricultural production.

In summary, in dynamic programming, an optimization problem is broken down into simpler and smaller sub-problems which are solved by the machine, and the storage of their immediate results not only promotes efficiency but also improves the process by avoiding repetition of computational steps (Bellman and Dreyfus, 1962). As described by Sutton and Barto (2018), the focus is on "(...) the use of value functions to organize and structure the search for good policies (...)". The work developed by Bellman is explored below, as it was the pillar for developing the recursive algorithms incorporated in dynamic programming.

2.3.3 Bellman's Principle of Optimality

Bellman's Principle of Optimality led to Bellman's Equations, which are the ground of dynamic programming. As it is exquisitely explained by Bellman and Dreyfus (1962), this principle states that "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy about the state resulting from the first decision".

Bellman's principle of optimality provides a framework for finding an optimal solution in a multi-step decision-making problem, with the assurance that such a solution exists and is unique. This principle involves breaking down a complex decision-making problem into sub-problems and recursively solving them, thereby enabling the optimal solution by exploring the development path process and controlling the feedback obtained (Bellman and Dreyfus, 1962), (Bellman, 1952). By using Bellman's principle of optimality, sub-optimal solutions can be deleted, improving the efficiency and effectiveness of their decision-making process. This principle has found widespread application in fields such as engineering, economics, and computer science. It has been used to solve real-world problems, from designing efficient algorithms to developing optimal control systems. It is expressed mathematically in Bellman's Equations, explained in the following section.

2.3.4 Bellman's Equations

The Bellman's equations, as the name suggests, were introduced by Richard Bellman, and they consist of a solid and highly relevant foundation for stochastic dynamic programming theory. Dynamic programming problems are often characterized by the uncertainty associated with future outcomes, and this mathematical framework allows their resolution while taking that into consideration (Bellman and Dreyfus, 1962), (Bellman, 1952).

The optimal solution to a problem is the sequence of actions corresponding to the highest expected reward. The Bellman Equations formulate the search for this solution considering the probabilistic outcomes that could arise from each decision at each state. The probabilities of transitioning between states and the expected associated rewards are known; therefore, the Bellman optimality equations can be approached in an iterative manner (Bellman and Dreyfus, 1962), (Bellman, 1952). The optimization problem is divided into smaller subproblems, facilitating its resolution.

Below, we can find a simplified version of Bellman's Equation, also known as the

optimality equation, retrieved from Ljungqvist and Sargent (2004):

$$V(x) = max\{r(x,a) + \beta \cdot V(T(x,a))\} \tag{2}$$

Where:

- V is a value function, for all $x \in X$;
- x is a state of the environment, $x \in X$;
- a is an action possible in the state x, $a \in A$;
- β is a discount factor with values between 0 and 1;
- $r(x,a)$ is the expected immediate reward associated with taking action a while in state x;
- T is a transition model describing the probability of moving from state x to another state y knowing the last chosen action a taken in state x.

The iterations of this equation result in a series that is known to converge uniformly, thus guaranteeing that the value function will converge to the optimal value. Before presenting the resolution process, it is important to understand under which conditions the optimal solution exists and how it is unique.

Existence and Uniqueness of an Optimal Solution

For the equation to converge, the value function V needs to be considered a contraction map, i.e., a function that, when applied to different points in a complete space, consistently reduces the distances between them through the use of a constant discount factor, β, while simultaneously maintaining the relative distance between them (Stokey, 1989).

Starting with the fact that the space in which the value function is defined must be complete, there are two possible cases to consider, discrete and continuous action and state spaces, and this condition can be verified in both. These are known to be complete for discrete spaces since every Cauchy sequence is convergent within said space. Regarding the latter, the action and state spaces must comply with some additional conditions, and the distance between points is measured with different metrics, such as the supremum

norm. A more detailed explanation can be found in Rudin (1976) since it falls outside this research's scope.

For the value function to be considered a contraction map of modulus β, it must comply with Blackwell's pair of sufficient conditions:

- **Monotonicity**: V is monotone, i.e, $f(x) \leq g(x)$ for all x that implies $(Vf)(x) \leq (Vg)(x)$ for all x;
- **Discounting**: For any nonnegative constant a and some $\beta \in (0, 1)$, $Vf(x) + a \leq Vf(x) + \beta a$;

The monotony condition symbolizes that if the value function is always greater or equal to another function, then when applying the Bellman operator, β, this order is maintained, which guarantees that the sequence of value functions is convergent. In the problem context, it means that as the agent learns through its previous actions, it will always perform the actions that provide equal or higher rewards, thus getting closer and closer to the optimal solution. As for the discounting condition, guaranteeing that the value function is bounded and the discount factor scales it down prevents it from diverging to infinity. This means that as the agent explores future actions, it associates lower rewards with future long-distance rewards compared to immediate rewards. These conditions, retrieved from Ljungqvist and Sargent (2004) and Stokey (1989), are met, and therefore the value function V is known to be a contraction map.

The Bellman operator is a transformation of the value function, and it can be defined as a continuous bounded function v into a continuous bounded function $Tv = \max\{F(x, a) + \beta \cdot v(y)\}$, where $y \leq g(x, a)$ and $x \in X$. By using the metric $d_\infty(v, w) = \sup_{x \in X} |v(x) - w(x)|$ we have a complete metric space. Supposing that $v(x) \geq w(x)$ for all $x \in X$, we have

$$Tv = \max_{a \in \mathbb{R}^k} \{r(x, a) + \beta \cdot V(v(y))\} \tag{3}$$

Below we can see that the operator is monotone and that it is discounting. Meaning that it forms a contraction since the Blackwell sufficient conditions are met.

$$
\begin{aligned}
Tv &= \max_{a\in\mathbb{R}^k}\{r(x,a)+\beta\cdot v(y)\}, \quad y\le g(x,a)\\
&\ge \max_{a\in\mathbb{R}^k}\{r(x,a)+\beta\cdot w(y)\} \quad y\le g(x,a)\\
&= Tw.
\end{aligned}
$$

$$
\begin{aligned}
T(v+c) &= \max_{a\in\mathbb{R}^k}\{r(x,a)+\beta\cdot [x(y)+c]\}, \quad y\le g(x,a)\\
&\ge \max_{a\in\mathbb{R}^k}\{r(x,a)+\beta\cdot v(y)+\beta\cdot c\} \quad y\le g(x,a)\\
&= Tv+Tc.
\end{aligned}
$$

T is a contraction present in a complete space and therefore results in the existence of a fixed convergence point, which is the unique solution of the Bellman equation. Now that it is clear that Bellman's equations converge to a unique optimal solution, we can look further into the process of iterating them.

Approaches to Solving the Bellman Equations

Based on the work of Ljungqvist and Sargent (2004), here we solve the Bellman equation (2) through the iteration of the value function.

Firstly, a sequence of value functions and a corresponding sequence of policy functions are created. Initiating the value function in the first point in time $V_0(x)=0$, we iterate the following equation until it has converged in V_j for a given $\hat{x}=g(x,a)$:

$$
V_{j+1}(x) = max\{F(x,a)+\beta\cdot V_j(T(x,a))\} \tag{4}
$$

When iterating Bellman's equation, only the action with the maximal value will be considered in the decision-making process. Below, there are some iterations of the equation.

$$
\begin{aligned}
V_1(x) &= max\{F(x,a)+\beta\cdot V_0(T(x,a))\} \qquad t=0\\
V_2(x) &= max\{F(x,a)+\beta\cdot V_1(T(x,a))\} \qquad t=1\\
V_3(x) &= max\{F(x,a)+\beta\cdot V_2(T(x,a))\} \qquad t=2
\end{aligned}
$$

...

$$V_j(x) = max\{F(x,a) + \beta \cdot V_{j-1}(T(x,a))\}$$

Through the iteration process, the difference between consecutive iterations becomes increasingly smaller as $|V_{j+1}(x) - V_j(x)| < \varepsilon$, where ε represents the desired accuracy of the estimates. The equation converges to the optimal value function, and this convergence is guaranteed by the properties mentioned above.

This method is a reliable tool to solve the Bellman equation and is explored in more depth in the work of Ljungqvist and Sargent (2004). This source also presents two alternative approaches to solving this equation, the policy function iteration and Howard's improvement algorithm, each with advantages and disadvantages.

2.3.5 The Curse of Dimensionality

As the number of dimensions of a problem increases, the iterative process described above becomes troublesome and more expensive to compute, ultimately becoming inconceivable. This phenomenon is known as the curse of dimensionality and constitutes a major limitation of dynamic programming. It was originally diagnosed by Richard Bellman and is a severe limitation in many real-life applications (Verleysen and François, 2005) (Sutton and Barto, 2018).

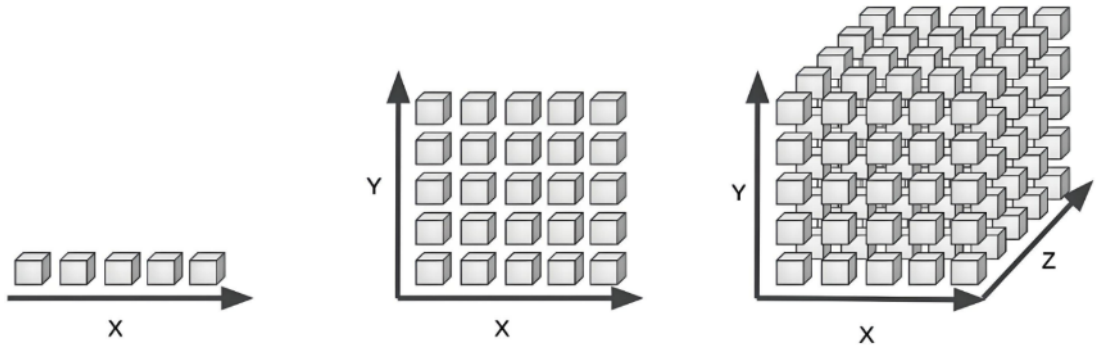

Figure 3: Illustration of the exponential growth of a multi-dimensional problem.

This limitation arises when iteratively solving Bellman's equation by discretizing the state space. By not approximating the value and/or policy functions, the number of discrete points increases substantially, increasing the computational complexity Ljungqvist and Sargent (2004).

A new set of methodologies has been carried out to overcome the curse of dimensionality. Approximate dynamic programming theory uses dynamic programming techniques combined with machine learning, recurring to approximated functions to represent the value function. This results in the reduction of the size of the state space and thus enables the solution of large-scale problems (Bertsekas and Tsitsiklis, 1996). The following section will cover the primary concepts of this theory.

2.4 Reinforcement Learning

2.4.1 Main Concepts and Architecture

RL distinguishes itself from other ML methods because it makes decisions based on the results of previous ones. It starts by acting on a trial and error approach, and through this process, the models collect information that allows for future decisions to be made from it to find the optimal solution to the problem. Every set of decisions is associated with a reward or a penalty, which will be used to maximize the final reward. Given the approach taken by RL and the nature of the problems it tackles, six main concepts are relevant to understand: agent, environment, state, action, policy, and value (Kober et al., 2013) (Sutton and Barto, 2018) and they have been introduced previously in section 2.1.

RL takes advantage of the policy and value functions described above, allowing either of them to be the main focus of the algorithms, being subject to iteration. There are two ways to define the path towards the optimal solution (Kober et al., 2013). One can view the solution as the one achieved through the policy associated with the highest reward - policy-based RL, or the agent can aim toward the optimal value achieved by any policy - value-based RL. This research is based on the latter. The solution can be achieved through optimizing the policy function to maximize the expected reward - policy-based RL, or through estimating the value function and then deriving the policy function - value-based RL. This research is based on the latter.

The goal is to continuously improve the sequence of actions the agent takes, thus increasing the total reward and eventually achieving the solution to the problem. The optimal solution consists of the one associated with the highest rewards, and the agent is consistently encouraged to act in a way that will converge to the optimal value through the reward associated with each action (Sutton and Barto, 2018).

Although RL algorithms can be adjusted to be applied to problems that do not consist of MDPs, this thesis is focused on problems that satisfy these conditions, taking advantage of the characteristics that naturally follow, such as the guarantee of convergence and the simplification of the decision-making processes (Szepesvári, 2010).

Understanding Value Functions

This section provides a concise overview of the definition of value functions, which are extensively discussed in the work of Sutton and Barto (2018).

Rewards that lead the agent to the optimal solution are measured by value functions, which can map the value of moving from a state through an action or by tackling the expected long-term rewards associated with every possible action.

Value functions are a central component of reinforcement learning (RL), measuring the expected long-term rewards an agent can achieve from different states. Among the types of value functions, the state-value function $V(s)$ holds particular significance. It represents the expected cumulative reward that an agent can obtain from a specific state, s, while following a given policy.

The state-value function, $V(s)$, captures the overall desirability of each state under a particular policy, with higher values indicating states that are more likely to result in higher long-term rewards. This function helps the agent assess the quality of different states and guides its decision-making process. Many dynamic programming algorithms, such as policy iteration and value iteration, utilize the state-value function to compute the optimal policy. By iteratively updating the state-value function based on the Bellman equation and refining the policy, these algorithms eventually converge to the optimal solution. Furthermore, the state-value function can also be employed in various model-free methods, such as Monte Carlo or temporal difference learning, where the agent learns directly from its interactions with the environment. In these scenarios, the state-value function enables the agent to adapt its policy and make better decisions as it gains more experience and knowledge about the environment.

Exploration-exploitation Trade-off

As covered in Sutton and Barto (2018), the exploration-exploitation trade-off is a fundamental dilemma in RL about choosing between exploring new actions and exploiting the knowledge it has already acquired. Exploring new actions is relevant to acquiring further information about the environment and might lead to discovering more optimal paths or sequences of actions. However, it is also relevant that the model considers the information gathered so far to choose which action to take next. This prevents the model from

taking less relevant actions to achieve the optimal solution. The balance in exploration and exploitation of the models will reflect on their performances, as excessive exploration can lead to the agent missing out on potential rewards and excessive exploitation might result in sub-optimal behavior.

Offline and Online Learning

As described in the work of Bertsekas (2022), there are two approaches to how the models learn from data - online and offline learning.

In an offline learning approach, the algorithms use a previously acquired, fixed dataset and thus learn separately from their decision-making process. With this framework, the agent does not interact directly with the environment or learn from new experiences. As an advantage, offline learning provides the security of having a model learn in a reliable and precise dataset; however, once the model is trained it might adapt poorly to a different environment or even different situations.

Online learning, on the other hand, provides an entirely different learning experience since the learning and decision-making processes are integrated with each other. The models acknowledge Real-time experiences, which learn directly and almost immediately from this newly acquired information. Dynamic environments are usually better described under online learning approaches as the models adapt their behavior instantaneously but at the cost of adapting their behavior to misleading or wrong information.

Both approaches have advantages and disadvantages and should be selected considering the context of the problem and the need for adaptability and flexibility. In this research, the learning approach is offline since the data is previously collected and fed into the models.

Model-based vs. Model-free Approaches

As studied in Sutton and Barto (2018), RL can be tackled through two different approaches, model-free and model-based. They operate differently and thus leading to very different outcomes.

In the model-based approach, the model is presented with a framework of the environment, which contains the transition and reward structures. After learning this framework,

the agent leverages it to learn from states and rewards by simulating actions on it. These models are adequate when planning for the actions to take is required, but they need a larger amount of data for proper training.

On the other hand, model-free models function without a specified model of their environment. These models tend to show better performances under environments that are more straightforward. They are more appropriate when planning the actions is unnecessary or when modeling the environment is highly complex or impossible. These methods are applied to problems in which the trial-and-error learning approach is viable and doesn't jeopardize the learning process, as the models can identify patterns and trends in the data.

The models used in this study are model-free approaches to RL, which are more appropriate considering the economic context of the problem. The complexity of the financial system and the changes in its dynamics over time prevent it from being modeled in a precise and accurate manner, which would then impact the reliability of the results (Ljungqvist and Sargent, 2004).

2.4.2 Approximate Dynamic Programming

As mentioned in 2.3.5, the state and action spaces grow larger and more complex, function approximation became convenient to reduce the computational costs of the models. Due to the many possible states and actions, exact dynamic programming methods such as value and policy iteration become computationally challenging and even infeasible. RL can be used under function approximation to overcome these limitations, which can be attained by different techniques such as a neural network, decision trees, or linear functions. These fall under a field known as Approximate Dynamic Programming (ADP), which combines dynamic programming principles with function approximation. ADP methods offer a scalable and efficient solution to complex reinforcement learning problems, providing good approximations to the optimal value function or policy despite the high dimensionality of the underlying problem. This technique is based on the same principles of RL, however, its historical developments remark from the RL literature. By solving problems in a sub-optimal way, ADP adds value by "(...) learning what to learn, and how to learn it (...)" to generate high-quality predictions (Powell, 2009).

The approximation method is important due to its impact on the computed predictions

of high-dimensional and continuous state-space problems.

The approximation approach should be chosen considering the problem and data characteristics, as it greatly impacts the results produced by the models. Linear function approximation will be taken as a foundation, building upon the work of Bertsekas (2011), which further develops the ideas introduced in Bertsekas and Tsitsiklis (1996) that cover both linear and nonlinear approaches.

To attain a sub-optimal solution to the problem at hand, the optimal value function $V(s)$ is replaced by an approximated version - scoring function or the approximate value function - $\tilde{V}(s, r)$ which is taken into consideration in every computational step. We can use nonlinear (neural networks) or linear architectures to attain this approximation. This research is based on linear approximation through polynomial functions.

A vector of parameters, r, is used to approximate the function's coefficients and this vector can be obtained through direct or indirect methods (Bertsekas, 2011). Direct methods minimize the approximation error, mostly using techniques such as gradient descent and least squares. The parameters are adjusted iteratively to reduce the difference between approximated and actual values. Indirect methods project the Bellman equation and focus on orthogonal characteristics that follow. They typically involve solving an associated problem that leads to a good approximation under certain conditions — for example, solving a related Bellman equation or applying a bootstrapping method like TD(λ) to adjust the parameters. These tend to be more efficient and robust since they take advantage of the specific structure of the problem. Both approaches attain to find a vector r that provides a good approximation of the value function under the chosen feature functions, with the parameters of the value function being updated on every iteration to bring it closer to the true value function.

Temporal difference methods consist of a subset of these ADP algorithms, and their primary concepts are described in the next section. A more detailed technical explanation of the variations used in this research can be found in the section 4.5.

2.4.3 Temporal Difference Algorithms

RL has had two main branches, one related to learning through trial and error and one focused on solving the problem of optimal control using value functions and dynamic programming. These two segments were, for a long time, pursued independently from one another, having more recently merged into what we call Temporal Difference (TD) methods. These methods learn straight from experience by updating the value function estimates for the next state. TD learning is a combination of Monte Carlo methods, which only update the value function based on the outcome of the action, and dynamic programming, which requires a complete and reliable model of the environment to update the value function continuously. They are widely used in RL since they are efficient and produce more accurate results. For this subject, this research is based on the work of Szepesvári (2010) and Sutton (1988).

TD methods have been verified to be a promising approach for most multi-step decision problems, standing out from other learning methods because they are driven by the difference between temporally successive predictions instead of focusing on the error between predicted and actual outcomes. The idea behind them is to apply bootstrapping to update the current state's value function estimates based on the current value and the predicted next value while simultaneously considering the next state's value function. The weights of the value function are updated according to the following equation:

$$\Delta w_t = \alpha(y_{t+1} - y_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w y_k \tag{5}$$

Applying TD methods comes with advantages compared to conventional methods, one being the ease of computing, as its incremental architecture saves in storage. Another advantage of these methods is that they use their experience more efficiently, converging faster and leading to more accurate predictions.

The temporal dependence between values for the economic metrics at hand indicates that the application of TD models is a good approach to forecasting the future values of these variables. This research will use three variants of TD methods to predict time series data - TD0, TD(λ), and GTD2. They were built based on the mentioned literature and are explained in more detail in the section 4.5.

3 Complementary Related Work

There has been growing interest in applying RL to forecast time series data, especially in the contexts of finance and economics. RL is less commonly used to forecast time series data when compared to other ML algorithms, such as auto-regressive models, moving average models, and recurrent neural networks. However, studies have indicated its possible success in this scenario, especially for problems in which the environment is dynamic and the relationships between the variables tend to be complex. This section aims to provide an overview of some of these papers and their main conclusions and challenges, which will provide ground and reasoning for the development of this research.

Yuwei Fu, Di Wu and Benoit Boulet (Fu et al., 2022) developed in the thirty-sixth AAAI Conference on Artificial Intelligence, a research paper exploring the results of applying a framework of combined RL models to forecast time series data. The framework chooses the best model to set the policy values at each step, improving the quality of the results. The conclusions of this research were positive as the models showed to be accurate and reliable when dealing with real-world time series data, even though the application of this framework on an imbalanced dataset is referenced as a limitation of the same.

The work of Seymour et al. (2004) reflects the proximity between temporal difference learning and the human learning processes. It states that the major advantage of temporal difference methods is the ability to learn from a given environment without being given any information about it apriori. Using previous data points to learn about the unknown environment, the models can withdraw relevant information, which is considered when attaining future predictions. These settings are relevant in the present study due to the economic context of data and the temporal connection between data points.

Last but not least, Steven Bradtke and Andrew Barto (Bradtke and Barto, 1996) performed a very insightful research on linear least squares function approximation, which consists of a solid foundation regarding the proposed models in the present study. It was shown that the algorithms converge faster and more accurately toward the optimal solution by approximating the adjustable parameters of temporal difference models.

4 Methods and Resources

4.1 Research Goal

The goal of this research is to explore the applications of RL models directed towards the prediction of time series, namely, the Portuguese quarterly gross domestic product (GDP), private consumption, investment, and exports. To achieve conclusions regarding this topic, the following research questions are considered:

- Is Reinforcement Learning relevant to forecast economic/business series?
- Are Reinforcement Learning algorithms effective and efficient to forecast time series?
- Which temporal differences method withholds the best results for time series forecasting?

4.2 Data Understanding and Preparation

4.2.1 Dataset Description

For the present study, official data from Statistics Portugal (in Portuguese, Instituto Nacional de Estatística) combined with data from the Bank of Portugal (in Portuguese, Banco de Portugal) is used to train, calibrate and test the models. This dataset has quarterly data from the beginning of 1977 until the end of 2022. It contains four economic metrics that we aim to predict: Gross Domestic Product (GDP), Private consumption, Investment, and Exports. The data is available in a quarterly manner, and it consists of a total of 184 observations and five attributes, as described below.

- OBS: Date of the record, in the shape YYYYQQ;
- GDP: Gross domestic product (2019Q4 = 100);
- PRIV_CONS: Private consumption (2019Q4 = 100);
- INVEST: Gross fixed capital investment (2019Q4 = 100);
- EXPORT: Exports of goods and services (2019Q4 = 100);
- IND_PROD: Industrial production (2019Q4 = 100);

As the data relates to the state of the economy over time, it is crucial to assess the impact of the Covid-19 pandemic, which had a significant economic impact in 2020 and early 2021. To adjust to this reality, all metrics in the dataset have been normalized to the fourth quarter of 2019, as this was the last quarter with typical economic conditions.

4.2.2 Outlier Treatment

Addressing outliers is highly relevant in the given context because RL requires smooth series to generate accurate predictions. The presence of outliers can induce errors in the pattern recognition process of the models, resulting in possibly incorrect or far from optimal solutions.

Below is a visual representation of the yearly median values of the four variables subjected to the study, which offers clear insights into the data's distribution and possible trends.

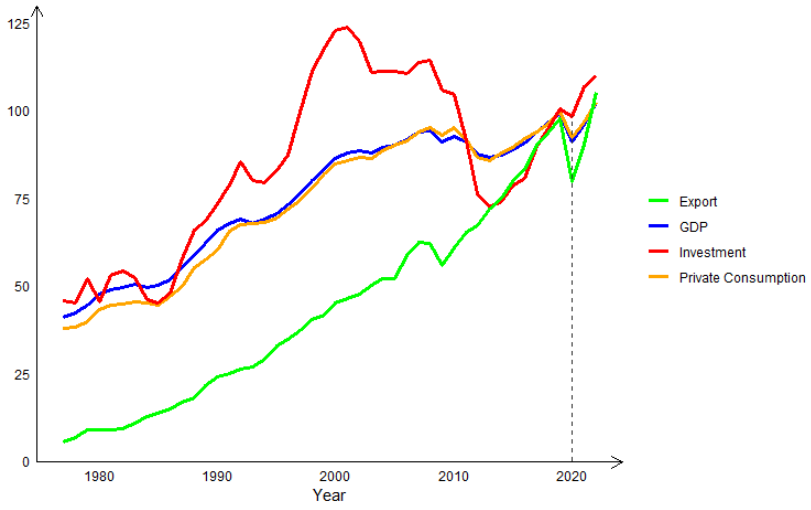

Figure 4: Overview of the four economic indicators over time.

It is noticeable that there is a significant drop in the year 2020. By taking a closer look at the quarterly values of these periods, we can assess if they impact the continuity of the series, thus constituting outliers.

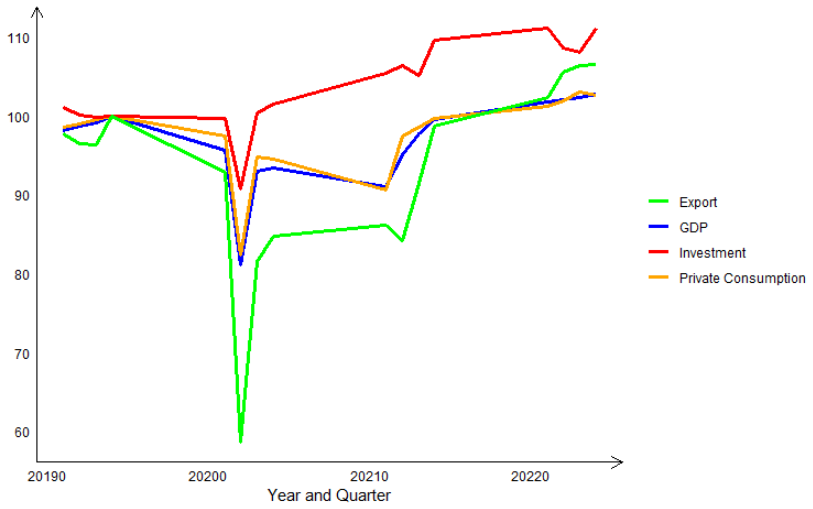

Figure 5: Overview of the four economic indicators over time between the last quarter of 2019 and the third quarter of 2022.

The second quarter of 2020 and the first quarter of 2021 come to attention, corresponding to the quarters highly impacted by the Covid-19 pandemic. The values corresponding to these quarters were replaced with the arithmetic mean of the immediately preceding and following quarter values. With this simple approach, the main characteristics of the dataset will be preserved, avoiding any possible impact on the quality of the attained predictions while guaranteeing the continuity of the series. As we can see in the plot below, the treatment was effective.

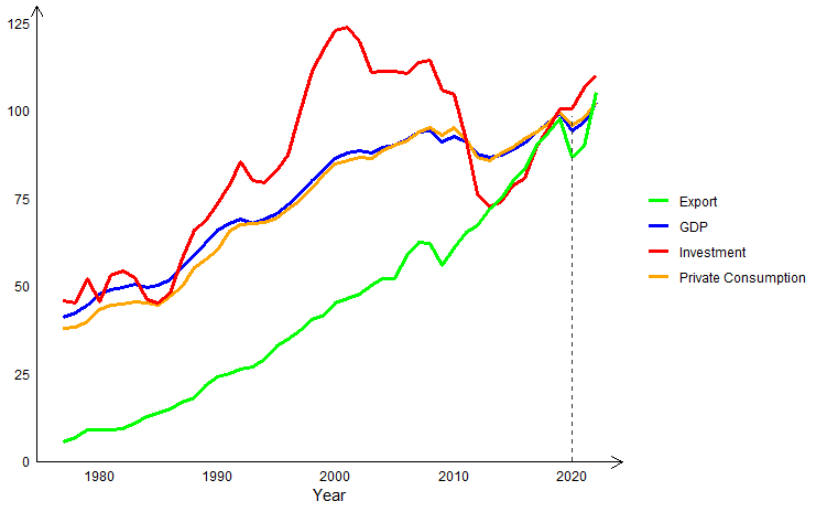

Figure 6: Overview of the four economic indicators over time between the last quarter of 2019 until the third quarter of 2022, after treating outliers.

The four target variables show no indication of abnormal behavior, indicating that the outlier treatment was effective. Below, there is the distribution of the original values side to side with the distribution of the treated values.

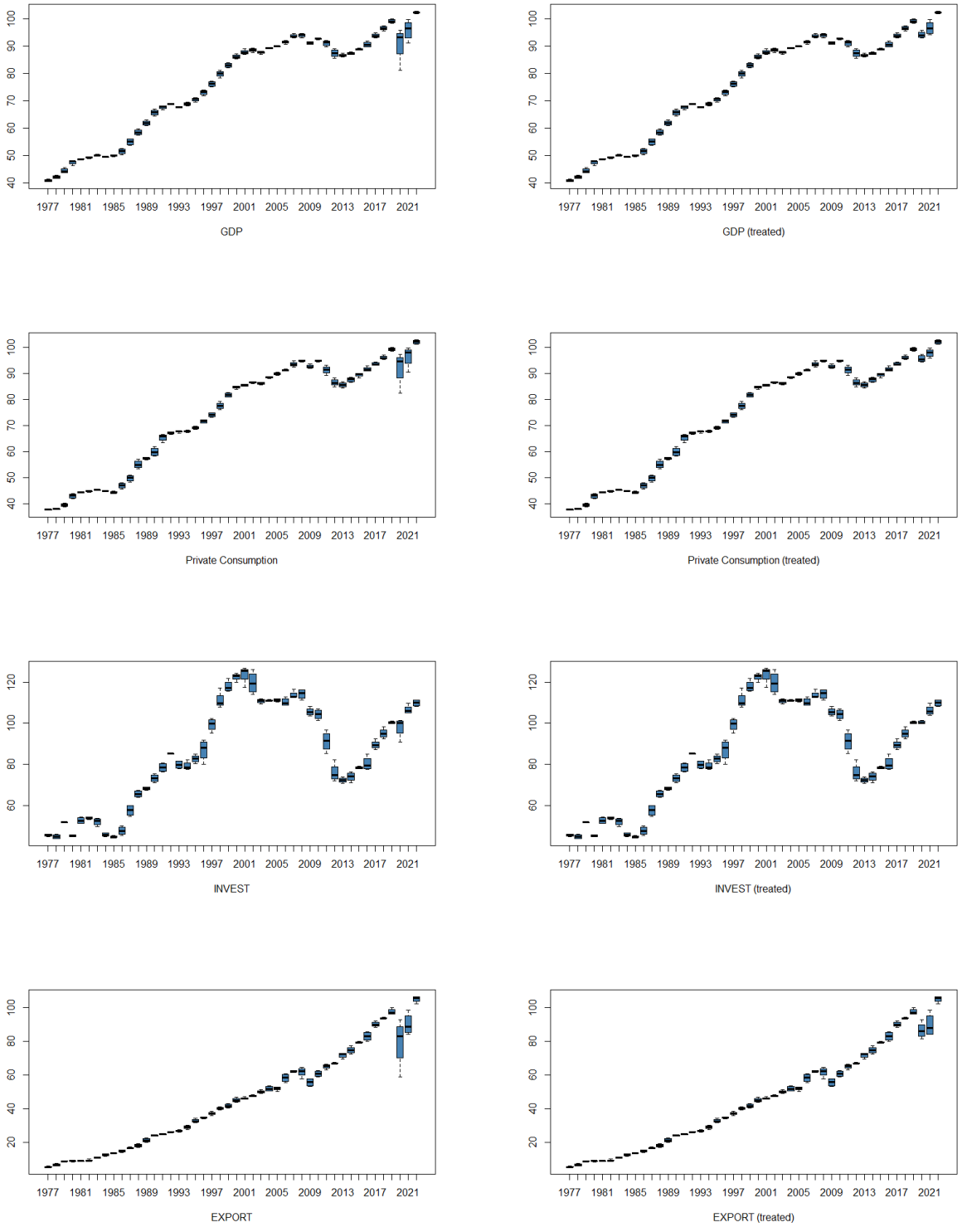

Figure 7: Variation of the four metrics over time.

4.2.3 Data Characteristics

The following visualizations and tables provide an overall view of the main characteristics and patterns in the behavior of the four economic metrics we aim to study.

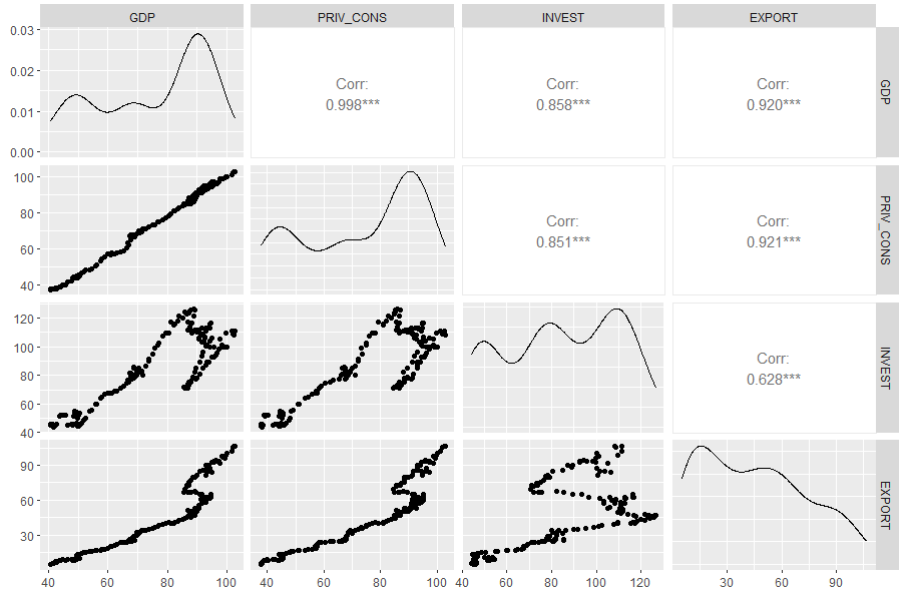

Figure 8: Scatter plot matrix showing correlations between all economic indicators.

The four variables present a certain degree of correlation among them, as expected due to their economic nature since Private consumption, exports, and investments are sub-components of GDP.

Below is a summary table for the four variables of interest, GDP, Private Consumption, Investment, and Exports. The table briefly overviews the main variables' distribution and variability. The four variables present similar distribution patterns, as evidenced by the similar mean, minimum, and maximum values. The standard deviation is lower for GDP than other variables, especially investment and exports.

Variable	Mean	St. Dev.	Min	Max
GDP	75.578	18.486	40.569	102.697
PRIV_CONS	73.915	20.448	37.596	103.000
INVEST	84.624	24.973	44.079	126.666
EXPORT	44.952	28.690	5.251	106.542

Table 1: Summary statistics of the main variables.

4.3 Cross-validation

Before diving into the details of each model's architecture, there is one concept that is important to understand - the concept of cross-validation. By dividing the data into subsets and using them for training and testing processes, cross-validation allows for the model's performance to be analyzed robustly and prevents the models from over-fitting, thus leading to more accurate results Berrar (2018). The way cross-validation is implemented varies according to the type of data at hand, mainly if it consists of a time series dataset. This subsection aims to understand the concept of classic cross-validation and how it differs from the cross-validation applied to time series data, which is the approach used in this research. The type of cross-validation this section refers to is known as k-fold and further detail on this technique and others within this topic can be found, for instance, in the work of Berrar (2018), Wong (2015) and Bergmeir et al. (2018).

Starting with a brief description of the classic cross-validation methods, the dataset is split into k folds, which are used selectively for training and testing purposes. Training is performed on all folds except one, which is used uniquely for the testing phase (Wong, 2015). As the following image suggests, this process is repeated until all folds have been used individually for testing.

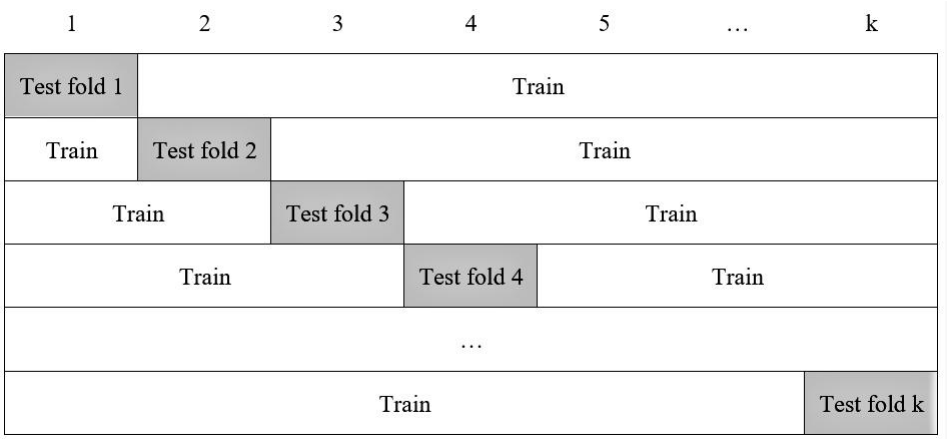

Figure 9: Classic K-fold cross-validation.

The folds are usually defined by randomly dividing a dataset into equal-sized folds or by defining stratified sampling, in which the folds preserve the general characteristics of the dataset by distributing the elements equally according to their features (Berrar, 2018). However, preserving the records' order is relevant in the presence of time series

data. This means that the folds should be established in a time-based approach to ensure that future-instanced data is not used to train for present data predictions (Bergmeir et al., 2018). Below is a visual representation of what cross-validation algorithms look like when adjusted and applied to time series data.

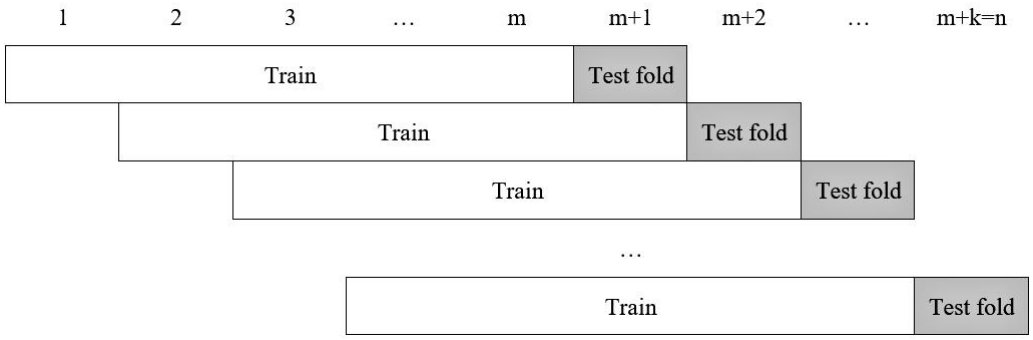

Figure 10: Cross validation adapted to time series data.

The implementations developed in this research use this kind of cross-validation scheme for time series. The total number of folds was established as 32, with this number being defined based on both the economic and mathematical perspectives of the method application (Lei et al., 2001). Given the economic context of the dataset, the economic cycles are an imperative aspect to take into account, and since their duration is between 6 to 8 years, i.e., between 24 and 32 trimesters, it is clear that the number of folds should consist of any number in this range. As for the mathematical context, the number of folds allows for more samples, producing better estimations of out-of-sample errors.

4.4 Benchmark Models

To evaluate the performance of the temporal difference models, these will be compared against two simple and well-established models - a Hodrick-Prescott filter model and an Auto-Regressive model. This section will provide a concise overview of these models used as a benchmark, elucidating their structure and key features.

4.4.1 Hodrick–Prescott Filter Model

The Hodrick-Prescott (HP) filter is considered a data-smothering method, and it dates back to the 1990s. To explain its design and relevance in the field, the work of Hodrick and Prescott (1997) is used as the primary reference. Further detail on this method and its economic applications can be found in the work of Baxter and King (1999), Kydland et al. (1990) and Backus and Kehoe (1992). This filter is recognized as the standard approach when studying cycles in the macroeconomic context, as it successfully favors long-term trends while diminishing the fluctuations that occur on a shorter-term basis. It separates the data into a trend component, representing the long-term behavior of the data, and a cyclical one that represents short-term fluctuations around the trend. It consists of a linear filter that applies a penalty for the roughness in the data, thus smothering it and consequently minimizing the sum of the smoothed data's squared second differences. This filter is often used to remove the cyclical component of the data, allowing for a deeper analysis of the long-term trends.

Its use under this particular scenario has been subject to debate. Some might argue that it can distort the behavior of the original data by considering false cycles. In contrast, others argue that it does not capture the most relevant features of the data. Regardless of this debate, it is still extensively used in economics.

The Hodrick-Prescott filter separates a time series y_t in trend z_t, cyclical c_t, seasonal s_t, holiday h_t and irregular ϵ_t components:

$$y_t = z_t + c_t + s_t + h_t + \epsilon_t, t = 1, 2, ..., n. \tag{6}$$

A prediction is attained by applying a one-period look ahead forecast at the end-of-sample ($t = n$):

$$\hat{y}_{n+1} = y_n + g_n + s_{n+1} - s_n \tag{7}$$

where $g_n = z_n - z_{n-1}$ is the first difference, i.e. growth, of the trend.

Estimating the trend growth is the main focus of this method, attained through minimizing the objective function displayed below.

$$\text{minimize } \sum_{t=1}^{T}(y_t - z_t)^2 + \lambda \sum_{t=2}^{T-1}[(z_{t+1} - z_t) - (z_t - z_{t-1})]^2 \tag{8}$$

where λ is a smoothing parameter that controls the trade-off between smoothness and fit to the data, and given that the data at hand is quarterly sectioned, it was set to 1600, as suggested in the literature. The first term measures the fit of the trend component to the data, while the second term measures the smoothness of the trend component. The trend component μ_t can be obtained by solving the above equation, and the cyclical component c_t can be obtained by subtracting the trend component from the original time series, i.e., $c_t = y_t - \mu_t$.

4.4.2 Auto-Regressive Model

Auto-Regressive Integrated Moving Average (ARIMA) models are usually applied to forecast time series data based on historical behavior. They consist of popular statistical models due to their remarkable capability of capturing complex patterns in the data, such as trends, seasonality, and cyclic fluctuations. This section is a summary of the work present in Hyndman and Athanasopoulos (2014).

ARIMA models try to describe the auto-correlations present in the data, and they are composed of a combination of Auto-Regressive (AR) and Moving Average (MA) models. In this research, the choice of parameters resulted in a first-order auto-regressive model.

This choice is driven by the model's ability to effectively capture the temporal dependency and the possible shocks often encountered in economics. Additionally, the easiness of interpretation and simplicity of the model also contributes to its selection.

AR models predict a variable based on its past values, which are manipulated linearly, as mathematically expressed below.

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \epsilon_t \tag{9}$$

Where ϵ_t represents white noise, the values of y_t are used as predictors, and this model is referred to as an AR(p) model of order p. These models can handle a significant variety of patterns, making them a reliable choice for economic forecasting applications.

By recurring to a first-order model, we have the following expression:

$$y_t = c + \phi y_{t-1} + \epsilon_t \tag{10}$$

4.5 Proposed Models

The purpose of this section is to go deeper into detail regarding the architecture of the Temporal Difference models suggested to predict time series data. Each of the instances of the models is presented below, with a brief explanation as to why they were used and which parameters they take. As mentioned previously, these methods are built and based on the work of Szepesvári (2010) and Sutton (1988).

Generally, the purpose of these models is to approximate the value function through a linear architecture, such as the one presented by Bertsekas (2011):

$$\tilde{V}(x, r) = r\phi(x) = \sum_{k=0}^{s} r_k \phi_k(x) \tag{11}$$

where $r0(r_0, r_1, ..., r_s)$ is the parameter vector and $\phi_r(x)$ are functions representing the features of state x, denominated *basis functions* by Szepesvári (2010). Defining $\phi_r(x) \equiv 1$, the linear architecture becomes

$$\tilde{V}(x, r) = r_0 + \sum_{k=0}^{s} r_k \phi_k(x). \tag{12}$$

Linear value approximation is mostly based on polynomial basis functions, $\phi_k(x) = x^k$ which results in the features $\phi(x) = (1, x, x^2, ..., x^s)^T$ (Szepesvári, 2010). Applying the quadratic case, the Bellman equation takes the form:

$$\tilde{V}(x, r) = \max \left\{ px + w - 0.5du^2 + \beta \left[r_0 + r_1(x + u) + 0.5r_2(x + u)^2 \right] \right\} \tag{13}$$

being $y = x + u$ the next state.

As explained in Ljungqvist and Sargent (2004) and in Bertsekas (2022), the Newton's one-step lookahead approximate policy function $\tilde{\mu}(x)$ for the parameters' values r_1 and r_2 respects the following first-order necessary condition:

$$-du + \beta r_1 + \beta r_2(x + u) = 0 \Rightarrow u = \frac{\beta r_1}{d - \beta r_2} + \frac{\beta r_2}{d - \beta r_2} x. \tag{14}$$

The parameter vector r can be estimated from data on transitions between states with TD algorithms.

4.5.1 TD(0)

TD(0) consists of a one-step learning algorithm and is the simplest form of TD learning. It makes predictions considering only the immediately preceding value. This method variation consists of a function that is called after each state transition, $TD(X, R, Y, V)$, described in algorithm 1. Given a finite Markov Process, the goal is to estimate the value function V considering a transition from a particular state X_t, and it's associated reward R_{t+1}.

Linear function approximation is applied for more efficient learning, integrated with a quadratic polynomial basis. This is a common choice since it is a solid approach to complex value functions recurring to accessible linear methods. This transforms every state variable into three features, $1, x, x^2$ which allows quadratic relationships between the state variables and the expected return to be considered. The respective weights are learned through interactions with the environment and are sequentially updated based on the TD error.

Algorithm 1 TD(0) function called after each transition.

1: **function** TD(0)(X, R, Y, V)
2: **Input:** X is the last state, Y is the next state, R is the immediate reward associated with this transition, V is the array storing the current value estimates
3: $\delta \leftarrow R + \beta V[Y] - V[X]$
4: $V[X] \leftarrow V[X] + \alpha\delta$
5: **return** V
6: **end function**

The parameter α represents the step-size sequence and takes small nonnegative numbers defined as c/t. The discount factor β was set to 0.96. The parameter c was calibrated in order to minimize the MAE value for each variable, the values taken are displayed in the table below.

Variable	c
GDP	0.00000001
PRIV_CONS	0.0000000005
INVEST	0.000000003
EXPORT	0.00000006

Table 2: Values taken by parameter c for TD(0).

4.5.2 TD(λ)

As mentioned above, both TD(0) and Monte-Carlo methods have their unique strengths, which can be achieved together in a new set of methods introduced by Richard Sutton, TD(λ). This algorithm differs from TD(0) since it considers all the past values to predict a future one, and their impact is measured by λ, which is designated as the trace-decay parameter. The algorithm attributes a higher weight to closer-in-time past values while considering that older ones have less impact, which promotes a balance between immediate and long-term values.

Regarding linear function approximation and the quadratic polynomial basis, these are applied similarly to TD(0), however, the weights calculated after each interaction with the environment are computed based on the TD error and the eligibility trace.

Algorithm 2 TD(λ) function called after each transition.

function TD(λ)(X, R, Y, V, z)
    Input: X is the last state, Y is the next state, R is the immediate reward associated with this transition, V is the array storing the current value function estimate, z is the array storing the eligibility traces
    $\delta \leftarrow R + \beta V[Y] - V[X]$
    for all x **do**
        $z[x] \leftarrow \beta\lambda z[x]$
        if $X = x$ **then**
            $z[x] \leftarrow 1$
        end if
        $V[x] \leftarrow V[x] + \alpha\delta z[x]$
    end for
    return (V, z)
end function

TD(λ) combines the generalization of Monte-Carlo methods while simultaneously allowing for bootstrapping. It is applicable for non-episodic problems, and by properly calibrating the trace-decay parameter, its convergence is achieved in a faster manner when compared to TD(0) and Monte Carlo methods. The parameters λ and β were set to 0.1 and 0.96 respectively. The parameter c was calibrated to minimize the MAE value for each variable; the values taken are displayed in the table below.

Variable	c
GDP	0.000000002
PRIV_CONS	0.000000005
INVEST	0.000000001
EXPORT	0.000000003

Table 3: Values taken by parameter c for TD(λ).

4.5.3 GTD2

Gradient Temporal Difference (GTD) learning algorithms converge to TD(λ solutions with a more robust and stable approach, while achieving a similarly good computational performance. In this research, we applied a GTD2 model with linear function approximation and quadratic polynomial basis.

This method updates two weight vectors, one by recurring to the gradient of the expected update and the other by approximating the expected feature vector. These vectors are then used and combined to compute the TD error and to update the primary weight vector, θ. This way it is ensured that the value function parameterized by θ converges towards a solution that minimizes the difference between the value of a state and the expected return from that state under a particular policy. The algorithm implemented in this research, proposed by Richard Sutton, is defined below

Algorithm 3 The function implementing the GTD2 algorithm

1: **function** GTD2(X, R, Y, θ, w)
2: **Input:** X is the last state, Y is the next state, R is the immediate reward associated with this transition, $\theta \in \mathbb{R}^d$ is the parameter vector of the linear function approximation, $w \in \mathbb{R}^d$ is the auxiliary weight
3: $f \leftarrow \phi[X]$
4: $f_J \leftarrow \phi[Y]$
5: $\delta \leftarrow R + \beta\theta^T f_J - \theta^T f$
6: $a \leftarrow f^T w$
7: $\theta \leftarrow \theta + \alpha(f - \beta f_J)a$
8: $w \leftarrow w + \beta(\delta - a)f$
9: **return** (θ, w)
10: **end function**

The discount factor β was set to 0.96. The parameter c was calibrated to minimize the MAE value for each variable, the values taken are displayed in the table below.

Variable	c
GDP	0.00000002
PRIV_CONS	0.00000004
INVEST	0.00000002
EXPORT	0.0.000003

Table 4: Values taken by parameter c for GTD2.

5 Findings

5.1 Performance Analysis

To assess the performance of the suggested models, the main performance indicators will be compared with the ones obtained through benchmark models. The performance indicators computed in this research are the Mean Absolute Error (MAE), the Mean Square Error (MSE), and the Root Mean Square Error (RMSE).

In the following tables, we find a direct comparison of these performance indicators with regard to the predictions of the four economic indicators.

Gross Domestic Product 1977 - 2022			
Model	MAE	MSE	RMSE
TD(0)	0.6082	1.3246	1.1509
TD(0.1)	0.6084	1.3206	1.1492
GTD2	0.6383	1.3104	1.1447
Median	0.6084	1.3206	1.1492
HP filter	0.7589	1.6073	1.2678
AR(1)	0.7675	1.4064	1.1859

Table 5: Models performance indicators for Gross Domestic Product forecast.

Private Consumption 1977 - 2022			
Model	MAE	MSE	RMSE
TD(0)	0.6207	0.8053	0.8973
TD(0.1)	0.6161	0.8164	0.9036
GTD2	0.6224	0.8039	0.8966
Median	0.6207	0.8053	0.8973
HP filter	0.7373	0.9943	0.9971
AR(1)	0.7237	0.8589	0.9267

Table 6: Models performance indicators for Private Consumption forecast.

Investment 1977 - 2022			
Model	MAE	MSE	RMSE
TD(0)	1.4021	2.8798	1.6970
TD(0.1)	1.4002	2.8775	1.6963
GTD2	1.4068	2.8803	1.6972
Median	1.4021	2.8798	1.6970
HP filter	1.6561	3.8815	1.9701
AR(1)	1.5021	3.5513	1.8845

Table 7: Models performance indicators for Investment forecast.

Exports 1977 - 2022			
Model	MAE	MSE	RMSE
TD(0)	2.2327	9.701	3.1147
TD(0.1)	2.2336	9.6113	3.1002
GTD2	2.2242	9.4830	3.0794
Median	2.2336	9.6113	3.1002
HP filter	2.3041	10.9260	3.3055
AR(1)	2.2175	9.8662	3.1410

Table 8: Models performance indicators for Exports forecast.

The MAE values allow us to determine how much difference there is between the actual and predicted values. A lower MAE indicates that the predicted value is closer to the actual value, and we can see that the three proposed models achieve a lower MAE compared to the baseline models.

The MSE is calculated as the average of the squared differences between the actual and predicted values, so the lower its value, the better the performance of the model. Similar to MAE, the proposed models show encouraging results by achieving lower MSE values.

By evaluating the RMSE, we can see how well the models fit the data. The RMSE

values obtained for the TD models are lower, so we can conclude that they fit the data better than the benchmark models.

The performance metrics presented emphasize different types of prediction errors since MAE does not place as much weight on larger errors compared to MSE and RMSE. Overall, the analysis of the results shows that the proposed TD models exhibit a better fit to the data and produce more accurate results for the economic variables.

6 Discussion and Future Developments

This research exhibits promising results in the application of TD models when forecasting smothered economic series, namely for key metrics such as GDP, private consumption, exports, and investment.

While Reinforcement Learning (RL) techniques have been applied to time series data in the past, these applications typically involved Neural Networks as an integral part of the process. Some noteworthy examples of these studies are Fu et al. (2022), Yu and Sun (2020) and Liu et al. (2005).

The application of temporal difference algorithms to forecast time series data represents a domain that is yet to be deeply analyzed and holds a lot of potential when it comes to exploring new applications of these models. This can be attributed to the architecture of the models and their limitations since they have been designed mainly for puzzle-solving and optimization problems. In this research, the quadratic case and the Newton method were leveraged to achieve the optimal policy and this simplification of the model's architecture might have had a positive impact on the results obtained.

Although this research has yielded positive findings regarding this application of temporal difference models, these models are also subject to some disadvantages that can impact their performances. For instance, the requirement of a smooth series for accurate predictions can be an impediment in some contexts. Additionally, the data's sequential nature might impact the exploration-exploitation balance since previous values influence future ones. This correlation between values might cause issues with the Markov property, meaning that the value of a state would be dependent not only on the last value taken but also on other previously taken values.

The major limitation of this application of TD models is the limited amount of data available, which is impossible to overcome due to the nature of time-series data. These models tend to converge rather slowly, which indicates that a higher number of records would reflect positively in it's success. The results of this research show that, with proper calibration of the parameters of the models and with cross-validation applied, TD models behave in a reliable manner when applied to real time-series data. Similarly to other mathematical algorithms, TD models might be suitable for this particular problem but would require refinement and adjustments for broader applicability. The investigation of such applications is a promising research topic for future papers.

7 Conclusion

Given the vast successful applications of reinforcement learning, this research sought to evaluate its performance when applied to time series data, specifically focusing on the forecast of four economic indicators - gross domestic product, private consumption, investment, and exports.

Temporal difference models, including TD(0), TD(λ), and GTD2, have been proven to be reliable when applied to multi-step decision-making problems, in which the algorithms learn from each step, eventually achieving an optimal solution. For benchmark purposes, basic models were implemented, particularly a Hodrick-Prescott filter and an Auto-Regressive model, which applications under similar circumstances have been previously studied and acknowledged as reliable. To assess the performance of these models, the main performance indicators - MAE, MSE, and RMSE - were compared between the proposed and the benchmark models.

Overall, we find that TD models have a better performance when compared to the benchmark ones, producing less erroneous predictions of the targeted variables. These results indicate that temporal difference methods can be successful when applied to time series data.

The forecast of economic indicators is highly relevant in a business context. The findings of this research provide ground for a new application of reinforcement learning methods with greater importance, making room for further analysis. These methods can be appropriately calibrated and applied to predict any other metrics of a time series nature, thus constituting a remarkable and powerful tool for business growth.

References

D. K. Backus and P. J. Kehoe. International evidence on the historical properties of business cycles. *The American Economic Review*, 82(4):864–888, 1992. ISSN 00028282. URL http://www.jstor.org/stable/2117348.

M. Baxter and R. G. King. Measuring Business Cycles: Approximate Band-Pass Filters for Economic Time Series. *The Review of Economics and Statistics*, 81(4):575–593, 1999. URL http://www.jstor.org/stable/2646708.

R. E. Bellman. On the Theory of Dynamic Programming. In *Proceedings of the National Academy of Sciences*, pages 716–719, 1952.

R. E. Bellman and S. E. Dreyfus. *Applied Dynamic Programming*. Princeton University Press, Princeton, N.J., 1962. URL https://www.rand.org/pubs/reports/R352.html.

C. Bergmeir, R. J. Hyndman, and B. Koo. A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics & Data Analysis*, 120:70–83, 4 2018. ISSN 01679473. doi: 10.1016/j.csda.2017.11.003.

D. Berrar. *Cross-Validation*. 01 2018. ISBN 9780128096338. doi: 10.1016/B978-0-12-809633-8.20349-X.

D. P. Bertsekas. Approximate Dynamic Programming (online chapter). In *Dynamic Programming and Optimal Control, Volume II*, chapter 6. Athena Scientific, Belmont, Massachusetts, 3rd ed. edition, 2011. URL https://www.mit.edu/~dimitrib/dpbook.html.

D. P. Bertsekas. *Lessons from AlphaZero for Optimal, Model Predictive, and Adaptative Control*. Athena Scientific, Belmont, Massachusetts, 2022. URL https://web.mit.edu/dimitrib/www/abstractdp_MIT.html.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, Massachusetts, 1996. URL https://www.researchgate.net/publication/216722122_Neuro-Dynamic_Programming.

S. J. Bradtke and A. G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22:33–57, 1996. doi: 10.1023/a:1018056104778.

A. Charpentier, R. Élie, and C. Remlinger. Reinforcement Learning in Economics and Finance. *Computational Economics*, 2021. ISSN 15729974. doi: 10.1007/s10614-021-10119-4.

Y. Fu, D. Wu, and B. Boulet. Reinforcement learning based dynamic model combination for time series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(6):6639–6647, Jun. 2022. doi: 10.1609/aaai.v36i6.20618. URL https://ojs.aaai.org/index.php/AAAI/article/view/20618.

A. Gattami, Q. Bai, and V. Aggarwal. Reinforcement learning for constrained markov decision processes. In A. Banerjee and K. Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 2656–2664. PMLR, 13–15 Apr 2021. URL https://proceedings.mlr.press/v130/gattami21a.html.

C. F. Hayes, R. Rădulescu, E. Bargiacchi, J. Källström, M. Macfarlane, M. Reymond, T. Verstraeten, L. M. Zintgraf, R. Dazeley, F. Heintz, E. Howley, A. A. Irissappane, P. Mannion, A. Nowé, G. Ramos, M. Restelli, P. Vamplew, and D. M. Roijers. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36, 04 2022. doi: 10.1007/s10458-022-09552-y.

R. J. Hodrick and E. C. Prescott. Postwar u.s. business cycles: An empirical investigation. *Journal of Money, Credit and Banking*, 29:1, 02 1997. doi: 10.2307/2953682.

B. Hunt, E. Carterette, and M. Friedman. *Artificial Intelligence*. Academic Press series in cognition and perception. Elsevier Science, 2014. ISBN 9781483263175. URL https://books.google.pt/books?id=9y2jBQAAQBAJ.

R. Hyndman and G. Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, 2014. ISBN 9780987507105. URL https://books.google.pt/books?id=nmTQwAEACAAJ.

J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32:1238–1274, 08 2013. doi: 10.1177/0278364913495721.

F. Kydland, E. Prescott, and R. Todd. Business cycles: Real facts and a monetary myth, 1990.

D. Lei, I. J. Anderson, and M. G. Cox. A robust algorithm for least absolute deviations. In *Algorithms for Approximation IV - Proceedings of the 2001 International Symposium*, 2001. URL dtic.mil/dtic/tr/fulltext/u2/p013759.pdf.

F. Liu, C. Quek, and G. S. Ng. Neural network model for time series prediction by reinforcement learning. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 809–814 vol. 2, 2005. doi: 10.1109/IJCNN.2005.1555956.

L. Ljungqvist and T. J. Sargent. *Recursive Macroeconomic Theory*. The MIT Press, Cambridge, MA, 2nd edition, 2004. URL http://englishonlineclub.com/pdf/RecursiveMacroeconomicTheory[EnglishOnlineClub.com].pdf.

W. B. Powell. What you should know about approximate dynamic programming. *Naval Research Logistics*, 56:239–249, 4 2009. ISSN 0894069X. doi: 10.1002/nav.20347.

S. Ross. *Introduction to Stochastic Dynamic Programming*. Elsevier Science, 1995. ISBN 9780125984218. URL https://books.google.pt/books?id=ofC9dQKXHqsC.

W. Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, Inc, 1976.

B. Seymour, J. P. O'Doherty, P. Dayan, M. Koltzenburg, A. K. Jones, R. J. Dolan, K. J. Friston, and R. S. Frackowiak. Temporal difference models describe higher-order learning in humans. *Nature*, 429:664–667, 06 2004. doi: 10.1038/nature02581.

M. Skare and T. Hasić. Corporate governance, firm performance, and economic growth – theoretical analysis. *Journal of Business Economics and Management*, 17:35–51, 01 2016. doi: 10.3846/16111699.2015.1071278.

N. Stokey. *Recursive Methods in Economic Dynamics*. Harvard University Press, 1989. ISBN 9780674735187. URL https://books.google.pt/books?id=BgQ3AwAAQBAJ.

R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 8 1988. ISSN 0885-6125. doi: 10.1007/BF00115009.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, Massachusetts and London, England, 2nd edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

C. Szepesvári. *Algorithms for Reinforcement Learning*. Morgan and Claypool Publishers, San Francisco, California, 2010. URL https://sites.ualberta.ca/~szepesva/rlbook.html.

M. Verleysen and D. François. The curse of dimensionality in data mining and time series prediction. volume 3512, pages 758–770, 06 2005. ISBN 978-3-540-26208-4. doi: 10.1007/11494669_93.

T.-T. Wong. Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation. *Pattern Recognition*, 48:2839–2846, 09 2015. doi: 10.1016/j.patcog.2015.03.009.

M. Yu and S. Sun. Policy-based reinforcement learning for time series anomaly detection. *Engineering Applications of Artificial Intelligence*, 95:103919, 2020. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2020.103919. URL https://www.sciencedirect.com/science/article/pii/S0952197620302499.

Z. Zhou and S. Liu. *Machine Learning*. Springer Nature Singapore, 2021. ISBN 9789811519673. URL https://books.google.pt/books?id=ctM-EAAAQBAJ.

8 Appendix

The R code used for this research is available in https://github.com/zaraandreia/Thesis.git.